SUPPORT VECTOR MACHINE PREDICTION OF HIV-1 DRUG RESISTANCE USING THE VIRAL NUCLEOTIDE PATTERNS

Seare Tesfamichael Araya

A dissertation submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science

Johannesburg, 2006

Declaration

I declare that this research report is my own, unaided work. It is being submitted for the degree of Master of Science in the University of the Witwatersrand, Johannesburg. It has not been submitted before any degree or examination in any other University.

Seare Tesfamichael Araya
20 th day of October 2006

Abstract

Drug resistance of the HI virus due to its fast replication and error-prone mutation is a key factor in the failure to combat the HIV epidemic. For this reason, performing pre-therapy drug resistance testing and administering appropriate drugs or combination of drugs accordingly is very useful. There are two approaches to HIV drug resistance testing: phenotypic (clinical) and genotypic (based on the particular virus's DNA). Genotyping tests HIV drug resistance by detecting specific mutations known to confer drug resistance. It is cheaper and can be computerised. However, it requires being able to know or learn what mutations confer drug resistance. Previous research using pattern recognition techniques has been promising, but the performance needs to be improved. It is also important for techniques that can quickly learn new rules when faced with new mutations or drugs.

A relatively recent addition to these techniques is the Support Vector Machines (SVMs). SVMs have proved very successful in many benchmark applications such as face recognition, text recognition, and have also performed well in many computational biology problems where the number of features targeted is large compared to the number of available samples. This paper explores the use of SVMs in predicting the drug resistance of an HIV strain extracted from a patient based on the genetic sequence of those parts of the viral DNA encoding for the two enzymes, Reverse Transcriptase or Protease, which are critical for the replication of the HIV virus. In particular, it is the aim of this reseach to design the model without incorporating the biological knowledge at hand to enable the resulting classifier accommodate new drugs and mutations.

To evaluate the performance of SVMs we used cross validation technique to measure the unbiased estimate on 2045 data points. The accuracy of classification and the area under the receiver operating characteristics curve (AUC) was used as a performance measure. Furthermore, to compare the performance of our SVMs model we also developed other prediction models based on popular classification algorithms, namely neural networks, decision trees and logistic regressions.

The results show that SVMs are a highly successful classifier and out-perform other techniques with performance ranging between (94.13%–96.33%) accuracy and (81.26% - 97.49%) AUC. Decision trees were rated second and logistic regression performed the worst.

Acknowledgements

The work on this thesis has been an inspiring, often exciting, sometimes challenging, but always interesting experience. It has been made possible by many other people, who have supported me.

I am very grateful to my supervisor Prof. Scott Hazelhurst who has supported me with his encouragement and many fruitful discussions. I would also like to express my sincere thanks to Lynn Morris from the NICD for her invaluable experience and advice regarding data and additional information.

I would also like to acknowledge the Carnegie Trust and WITS AIDS research institute for some financial help towards this research. I am also grateful for the support I got from the National Bioinformatics Network (NBN).

Finally, I wish to thank my parents for their continuous support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The Human Immunodeficiency virus (HIV) is a rapidly evolving virus resulting in the AIDS epidemic. The rapid evolution of this virus is due to fast replication and its mutant behaviour. This mutant behaviour of the virus gives it the advantage of acquiring drug resistance which is the main reason for the failure of much diagnostic treatment. One solution that has proved to give better results is pre-therapy drug resistance testing. This solution allows for administration of drugs that will prolong viral suppression and help reconstruct the patient's immunity. There are two approaches for testing HIV drug resistance, phenotypic and genotypic testing. Phenotypic testing is laboratory based and measures the relative drug susceptibility of an HIV strain directly [Beerenwinkel *et al.* 2003a]. Genotypic testing, on the other hand, considers the genetic information of the HIV strain extracted from the patient and the ability to interpret such data [Beerenwinkel *et al.* 2003a]. Due to the cost benefit and other advantages genotyping is more widely used than phenotyping.

Genotypic testing aims at identifying specific mutation points on the viral genetic makeup that are known to confer drug resistance based on the biological knowledge available about the virus and the drug/s. In genotyping, the genetic make up of the HIV strain extracted from a patient is examined for the existence of mutations (patterns) that are known to confer drug resistance to a drug or combination of drugs and then be classified as resistant or not resistant to the drug or combination of drugs accordingly. There are a number of ways of performing this task and interpreting the results of genotyping is not related to the testing process, hence this approach has been an ideal application for computerised expert systems [Lathrop *et al.* 1999]. Based on these characteristics, one can consider genotyping as a classification problem which is best solved using pattern recognition techniques than a traditional algorithmic approach. A number of applications addressing the HIV drug resistance problem as a pattern recognition problem are currently available. Draghici and Potter [2003] and Wang and Larder [2003] used neural networks as a tool for predicting the drug resistance profile of HIV strain based on genetic sequence of viral protease and amino acids on selected positions of protease where mutation is known to confer drug resistance respectively. Beerenwinkel *et al.* [2002] used decision tree

models to predict phenotypic resistance from genotypic information.

Pattern recognition techniques have been proved to be a better option when there are no known rules relating the input and output of a problem, and the interaction between different reactants is unknown. One such pattern recognition technique is the Support Vector Machine (SVMs), which is a statistical learning method proposed by Boser *et al.* [1992]. Although this technique is a relatively recent addition to pattern recognition techniques, it has shown superior performance as a tool for pattern recognition in numerous benchmark applications. Furthermore, its ability to give a better performance based on limited training samples makes it an ideal approach in computational biology problems, where generating data is costly or difficult. One such computational biology problem is HIV drug resistance. This research investigates the performance of SVMs in predicting drug resistance of an HIV strain based on the genetic sequence of two enzymes; namely Reverse Transcriptase (RT) and Protease (PRO), which are critical during viral replication and where most mutations are exhibited [Shafer 2002b]. The performance of SVMs will be compared to three traditional methods of pattern recognition: neural networks, decision trees and logistic regression.

The rest of the chapter is organised as follows. The next section gives an introduction to HIV and the drug resistance problem. This section also shows that this is an open area of research and indicates the importance of the research. Section 1.3 gives an introduction to Support Vector Machines, which is the pattern recognition technique used in this research. This section will highlight why SVMs are an ideal tool for this research and present some applications of SVMs in computational biology. Section 1.4 and 1.5 give the formal definition of the problem and an overview of the approach taken to assess the performance of the different algorithms respectively. Section 1.6 gives an overview of the results found in this work followed by a section highlighting the contribution of the research. Finally, section 1.8 gives the structure of this document.

1.2 HIV and the drug resistance problem

The Human Immunodeficiency Virus (HIV) and its infectious agents have resulted in the worldwide AIDS epidemic. World Health Organisation (WHO) statistics shows that as of December 2003, HIV had already infected between 34 and 46 million people around the world with the majority of these infection in sub-Saharan African countries [UNAIDS 2004]. Although it has been more than two decades since HIV was first discovered, there is no effective drug to fully stop the virus from replicating and stop the epidemic. At the present there are more than 18 antiretroviral drugs available and the best they achieved so far is prolonging viral suppression and helping with immunologic reconstruction using combinatorial therapy (combination of up to 3 or 4 drugs) [Shafer 2002a]. A major reason for the failure to halt the epidemic is HIV drug resistance.

HIV drug resistance refers to the loss in the ability of a drug or combination of drugs to suppress the replication of the virus. The two main reasons for drug resistance are fast replica-

tion and mutation [Klatt 2003]. Mutation refers to the change in the genetic make up (RNA) of the virus during replication. During replication, the viral RNA integrates with the human cell DNA in a number of steps with the aid of different enzymes such as RT and PRO [Klatt 2003]. HIV mutants result from the high viral replication rate and RT infidelity. RT infidelity is due to the lack of proof-reading mechanism that preserves the genetic composition of double stranded DNA genome within the viral RT [Shafer 2002a]. Although all mutations change the structure of the virus, not all of them have the same effect. Some mutations cause the virus to become extremely infectious while others make it weak. For example, if a mutation occurred that made the newly replicated virus resistant to a particular drug or combination of drugs, treating the patient with this drug or combination of drugs will have a negative effect. On the contrary, other mutations slow down the replication process of the virus. Therefore, one of the main concern during pharmaceutical therapy of an HIV patient is identifying particular mutations known to confer drug resistance and administering drug/s accordingly. Identification and prediction of HIV drug resistance is the problem this research is addressing. To address this problem a number of approaches have been proposed within the last decade. In this regard, one of the most effective approaches has been pre-therapy HIV drug resistance testing [Hoffmann and Kamps 2003].

There are two approaches of testing an HIV's strains drug resistance: namely phenotypic testing and genotypic testing [Klatt 2003]. Phenotypic testing directly measures the replication of the virus in the presence of a drug or combination of drugs. Genotypic testing on the other hand relies on the genetic sequence of the HIV virus and theoretical knowledge of specific mutation and related drug resistance. The primary objective of genotypic testing is to detect specific mutations known to confer resistance to antiretroviral drugs in well-defined regions of the RT and PRO, the two critical enzymes in viral replication [Klatt 2003]. Genotypic is the most widely used because of its simplicity, speed, cost benefit and unlike phenotypic testing it doen not require a specialised laboratory. Furthermore, interpreting the result of genotypic is independent of the testing process. The later advantage, has opened the door for successful technology from other disciplines to collaborate in solving the problem. Such an application is the computerised expert system [Lathrop *et al.* 1999].

The first research carried out to address this problem of perdicting the effect of a drug or combination of drugs based on the genetic sequence of the HIV strain extracted from the patient using computerised expert systems was conducted by Lathrop *et al.* [1999]. Lathrop *et al.* [1999] introduced the AI system, CTSHIV that uses the scientific knowledge about HIV drug resistance to customise treatment to an individual patient. Following this breakthrough a number of techniques such as neural networks [Draghici and Potter 2003; Wang and Larder 2003], decision trees [Beerenwinkel *et al.* 2002], and different regression methods have been applied to predict drug resistance of an HIV strain directly or indirectly. Although most of these applications have been successful, the performance obtained is still far from optimal. Furthermore, the design of these systems were highly dependent on the biological knowledge available about the particular drug or combination of drugs and corresponding mutations that

are known to confer resistance. This means that for these systems to work properly a continuous update (with possible redesigning) of the systems is required with the discovery of new mutation points and new drugs. Therefore, the search for new techniques that not only outperform the existing ones but also address the dynamic behaviour of the virus (mutation and drug resistance) is required. This research will investigate such a technique by taking no biological knowledge about the virus into consideration while designing the model. Furthermore, this research uses the position $1-250$ of RT and position $1-99$ of PRO genetic sequence. Hence, the resulting model bases its decision not only on mutations already reported to cause drug resistance but patterns yet not known by humans but reflected in the genetic sequence of the strain.

1.3 Support vector machines

Support Vector Machines (SVMs) are a statistical pattern recognition technique proposed by Boser *et al.* [1992] to perform a number of classification and regression tasks in a wide variety of application domains [Cristianini and Shawe-Taylor 2000]. Since their introduction, SVMs have been successfully applied to many pattern recognition and regression applications such as text categorisation [Joachims 1998], speech recognition [Ganapathiraju 2002], face detection [Osuna *et al.* 1997], object recognition [Schölkopf 1997], and handwritten text recognition [Cortes and Vapnik 1995] with remarkable performance. The achievements by SVMs are credited to the two basic principles behind SVMs: namely the principle of Structural Risk Minimisation (SRM) and the ability to project data into high dimensional feature space where complex data can be classified with ease [Vapnik 1995]. In this section we will see a high level introduction on how a simple maximum margin classifier, which is the simplest form of SVMs, is extended into complex classifiers that are complex enough to model real world problems, yet simple enough to be analysed mathematically. But before discussing the classification method, an introductory summary of risk minimisation and the principle of Structural Risk Minimisation will be presented. A more detailed explanation of risk minimisation and the mathematical formulation of SVMs will be presented in Chapter 3.

Like all pattern recognition techniques, SVMs are also aimed at obtaining the best classification based on a limited number of training samples. There are a number of optimisation criteria to estimate the performance of a classifier. One such criterion, that is most commonly used by traditional pattern recognition techniques, is empirical risk minimisation [Ganapathiraju 2002]. The empirical risk of a classifier is defined as the sum of the cost of misclassification of the training sample. Based on this definition, it is intuitive to say that there can be a number of configurations that can minimise the empirical risk or even achieve zero empirical risk. But the best configuration is the one that trades-off between the empirical risk and expected error (the error on an unseen validation set). To decide on the best configuration that gives the least expected error, Vapnik and Chervonenskis proposed the statistical learning theory. This theory suggests that it is necessary to minimise the capacity of the set of functions (for example the degree of a polynomial function) relative to the number of training samples along with the empirical risk

[Vapnik 1995]. This is called the principle of Structural Risk Minimisation (SRM). There are a number of possible approaches to minimise both the capacity and the empirical risk. Some pattern recognition techniques accomplish this by defining the capacity of the set of functions and minimising the empirical risk while others minimise the capacity of the set of functions for a predefined empirical risk. However, SVMs accomplished this by simultaneously minimising both the empirical risk and the capacity of the hypothesis space [Osuna *et al.* 1997].

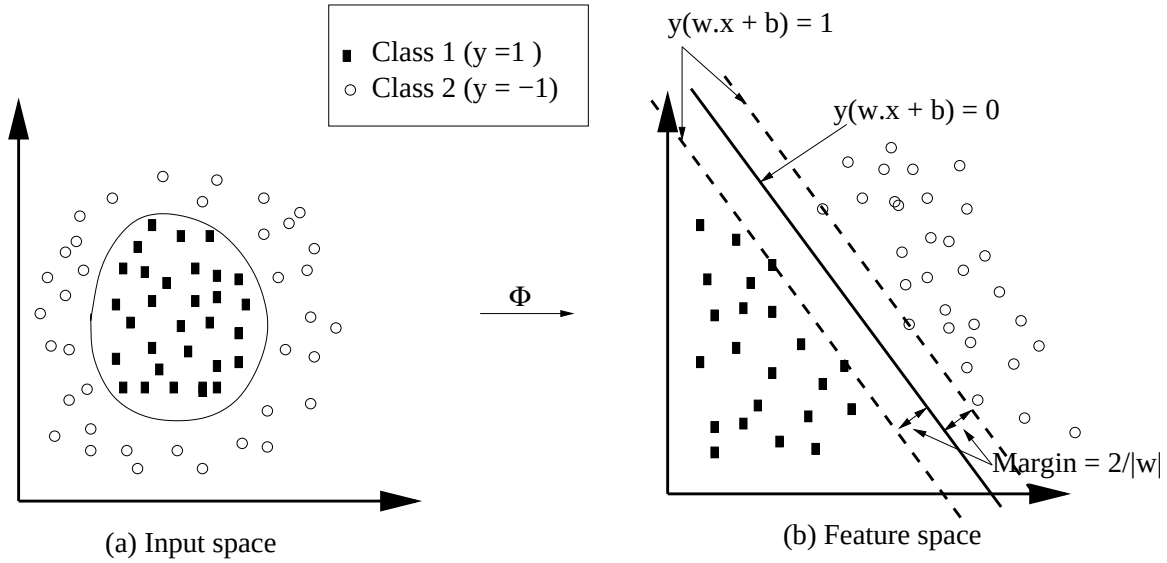

Figure 1.1: A toy example illustrating the SVMs training method. For a given non-linear classification problem shown in the input space, the SVMs map the training data nonlinearly into a possible higher dimensional feature space and construct the optimal hyperplane with maximum margin there. The square boxes and circles indicate positive and negative examples to be classified. The solid separating hyperplane in the feature space is the optimal hyperplane and the margin is defined as the distance between this hyperplane and any one of the hyperplane shown by broken line. The training samples that lie on the hyperplane indicated by a broken line are called support vectors

The simplest form of SVMs works well when the training sample is linearly separable by an optimal hyperplane, leaving all members of the same class on one side of the hyperplane and the rest on the other (see Figure 1.1.b). The optimal hyperplane is defined as the one that maximises the minimum distance between either of the two classes and itself. This distance is called the margin of the classifier. To find the optimal hyperplane, the constrained optimisation problem should be solved (i.e. maximising the margin constrained to the equalities given in figure 1.1.b). The solution of this optimisation problem is the orientation of the optimal hyperplane. Using the classical Lagrangian approach to solve this optimisation problem, the orientation of the hyperplane will be given in terms of the training samples and their corresponding non-negative Lagrangian multiplier [Cristianini and Shawe-Taylor 2000]. The classical Lagrangian approach also has the advantage of emphasising the importance of some training examples over the rest and this is reflected on the solution by showing that not all of the training samples have non-zero Lagrangian multipliers but rather a subset of the samples. Those training samples with non-zero Lagrangian multipliers are the ones that determine the optimal hyperplane and are called

Support Vectors (SV) [Vapnik 1995]. Note that for a classifier to perform well, the number of SVs should be relatively small otherwise over-fitting might occur [Vapnik 1995]. Given the orientation of the hyperplane and the threshold (which can also be determined mathematically in terms of the SVs, their corresponding label and Lagrangian multiplier) classification of an unseen sample will be according to the sign of the function:

$$f(x) = sign(\sum_i y_i \alpha_i (x.x_i) + b) \tag{1.1}$$

where x is the sample to be classified, α_i, the Lagrangian multiplier for the Support Vector x_i is found as a solution for the optimisation problem and the orientation of the optimal hyperplane w is given by $\sum_i \alpha_i x_i$ and b is the threshold..

So far we have seen the case where the training sample is perfectly separable by a linear hyperplane. But how do SVMs handle the case where the data is not linearly separable or not linear at all?

The case where the data is not linearly separable is handled by associating a misclassification cost whenever necessary. Hence the task is not only finding an optimal hyperplane that maximises the margin but also minimises the misclassification cost. The result of this optimisation problem also gives us the same decision function (equation 1.1), but the user will be required to freely select a parameter that trades off between the width of the margin and the misclassification error when performing the training (defining the model).

Most real life classification problems are not linearly separable and need a complex classifier. SVMs handle such complex classification problems by mapping the data from the input space (Figure 1.1.a) into a possibly high dimensional feature space (Figure 1.1.b), where the data can be separated by a simple maximum margin classifier. That means one needs to choose a mapping $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^m$ usually an Hilbert space $(\mathscr{H})$ where $m \geq d$ (usually the mapping is into a higher dimensional space) such that the data which was not linearly separable will become linearly separable (the two cases specified above). One question associated with this is how do we choose the mapping function $\Phi(x)$.

As it can be seen from the above equation, the decision function is based on the dot product between the sample to be classified and the support vectors. Hence the decision function in the feature space will have the form $f(x) = sign(\sum_i y_i \alpha_i (\Phi(x).\Phi(x_i)) + b)$. In addition to this, Mercer's theorem has shown that any dot product in a feature space can be computed by a function in the input space without the need for explicit mapping [Ganapathiraju 2002]. These functions are called kernels (see section 3.4).

There are a number of points to note here. Firstly, the use of kernel functions enables SVMs to implicitly perform classification on the feature space without the need to perform the mapping first and hence prevent the curse of dimensionality from occurring. Secondly, as is the case for the linearly separable and non-separable datasets, only support vectors are involved in determining the shape of the hyperplane.

Selection of a kernel map is problem specific and it is up to the user to choose one. Kernel

selection can be based on prior information about the problem (for example: mutation points), successful application of the kernel on similar problems or, in the absence of both of the above, kernels can be selected by empirically testing their performance in the given problem. There are a number of kernel functions known to give better performance depending on the characteristics of problem or the training patterns. However, the polynomial kernel of lower degree and the Radial Basis Function (RBF) are known to perform well on most datasets.

The polynomial kernel has the form:

$$k(x, y) = (x.y + 1)^d, \tag{1.2}$$

where d is the degree of the polynomial. A large value of d refers to a more complicated decision boundary. And for linear classifiers (linear hyperplane) the value of d is set to 1.

The radial basis function (RBF) kernel has the form:

$$k(x, y) = \exp(-\gamma|x - y|^2), \tag{1.3}$$

where γ is the width of the kernel. The smaller γ, the smoother the decision boundaries. This kernel is more favourable when one class is totally encircled by the other as shown in the Figure 1.1. The bigger γ gets, the tighter the closed boundaries (circles) become.

Summing up, SVMs are becoming popular because:

- Unlike most traditional pattern recognition techniques that use empirical risk minimisation, SVMs use structural risk minimisation, which minimises the error on yet-to-be-seen data and hence has good generalisation performance.

- Unlike other complex pattern recognition techniques, such as neural networks, which are very hard to analyse theoretically, SVMs are easy to theoretically analyse without losing the ability to solve complex pattern recognition problems [Hearst *et al.* 1998].

- Choosing different kernel functions gives different architectures which suit the problem at hand. Polynomial, RBF and Sigmoid kernels simulate polynomial classifiers, RBF classifiers and three layer (including feature and output layers) neural networks respectively [Hearst *et al.* 1998].

For these reasons and their incredible performance in many benchmark applications such as text, speech and image recognition, SVMs have been recently applied in many biological problems and have outperformed other pattern recognition techniques. Knowledge-based microarray analysis by Brown *et al.* [2000] and classification and validation of cancer tissues using microarray expression data by Furey *et al.* [2000] are two examples worth mentioning. Common problems among these and many other pattern recognition tasks in computational biology are the shortage of training samples compared to the large size of features per pattern, and the availability of noise in the patterns. These two problems have been the main reason for the

indifferent performance of other pattern recognition techniques. However, previous applications of SVMs in computational biology or bioinformatics have shown that these problems are not as critical for SVMs as they are for other pattern recognition techniques [Cristianini and Shawe-Taylor 2000].

1.4 Drug resistance as a pattern recognition problem

In the previous two sections, the HIV drug resistance problem, the methods of HIV drug resistance testing, and the advantages of the state-of-the-art pattern recognition technique, SVMs, were presented. This section will discuss how the HIV drug resistance problem can be considered as a pattern recognition problem and then exactly define the problem for this research.

In section 1.2, it was pointed out that certain mutations cause resistance and others do not. In genotype resistance testing, specific positions in the genetic sequence of the HIV strain will be checked for the existence of certain amino acids and depending on the result the relative susceptibility of the HIV strain to a drug is measured. For example, consider the mutation represented by the standard notation M184V, known to confer resistance to Eqivir (one of the 18+ antiretroviral drugs available). The biological interpretation of the above mutation is as follows. If mutation occurred and the amino acid methionine (M), which is found in position 184, is changed to Valine (V) the newly replicated virus becomes resistant to Eqivir [Hoffmann and Kamps 2003].

If we see this problem from a computational (pattern recognition) angle and represent each sequence as a vector in a Euclidean space, different sequences will be plotted to different points in this space depending on their genetic make-up (the amino acid sequence and its corresponding numeric value). Considering the above example, the virus before mutation (non-resistant virus) will have the numerical equivalent of M at its 184th vector component and is plotted as a point. However, the mutant (resistant) virus will have the the numerical equivalent of V at its 184th vector component and will definitely be plotted into a different point in our Euclidean space. Once the Euclidean space is divided into disjointed regions each representing a pattern class (Susceptible, Intermediate, Resistante) based on the training samples available, a pattern (HIV strain) can be labelled to its appropriate drug profile depending upon the region to which it is plotted.

The drug resistance problem this research is trying to address can be redefined based in the above argument as:

> Given a set of phenotypic data where each sequenced HIV strain is labelled as *Susceptible, Resistant* to a particular drug, can SVMs learn from these examples and predict the drug resistance profile of an unseen HIV strain?

which is the question for this research.

Drug profile of an HIV strain can be susceptible, intermediate or resistante. But very few of the pattens in the training data are given as labled intermediate. Hence we consider binary clas-

sification in the research. The research question is answered by investigating the performance of SVMs and comparing it to the performance of neural networks, decision trees and logistic regression models.

1.5 Overview of the research approach

To answer the research question a comparative approach is used. The performance of SVMs in predicting the drug resistance behaviour of HIV based on the viral nucleotide sequence was empirically tested. The SVMs' performance was then compared to that of neural networks, decision trees and logistic regression. These classification techniques were selected because of their popularity and reputation as a classification tool and have been already applied for predicting HIV drug resistance. The data used for the empirical testing was the same used by Ravela *et al.* [2003] which comprise sequences of isolates from 2045 individuals. Each sequence constitutes 297 nucleotides of the viral protease and 720+ (some are longer than 720) nucleotides of the viral reverse transcriptase.

One of the factors affecting performance of a classification algorithm is the input encoding techniques. When encoding the input data some biological knowledge about HIV drug resistance was incorporated. As mentioned previously, HIV drug resistance is related to mutation in specific positions of the viral genome (for example M184V). However not all mutations have the same effect. Some of these mutations are major and might cause resistance alone while others require the existence of the major mutations to cause resistance. To capture this property of the data, each sequence was converted into a vector form that highlighted both the global and local position of these mutations. Each sequence was grouped into non-overlapping triplets and each triplet was given a equivalent numeric value. With this scheme each sequence was converted to a vector in some higher dimensional space.

After encoding the data, the first set of experiments was done using SVMs. The experiment with SVMs started by further pre-processing the input data to make it suitable for the domain-restricted kernels and simplify the generalisation and error estimation. Then grid-search using cross-validation was conducted to select an appropriate kernel and tuning its diagonal factors. The two kernels used in this experiment were the polynomial (Equation 1.2) and radial-basis function (Equation 1.3). There are two parameters of interest for each kernel selected. For the polynomial kernel the parameters of interest are the degree of the polynomial d and a regularisation constant which indicates the trade-off between the training error and the separating margin. The parameters of interest for the RBF kernel are the width of the kernel γ and the regularisation constant. Kernel selection is done staring from a simple dot product kernel to a higher degree polynomials and then different RBF kernels.

The second set of experiments was done using neural networks. The network architecture used in this research is the popular feedforward multilayer perceptrons with back-propagation learning. There are different modifications to the standard back-propagation algorithms each with their advantage in terms of memory usage, convergence speed and training set size. Stan-

dard and Resilient backpropagation and Levenberg-Marquardt algorithms were tested. Levenberg-Marquardt was found to be more efficient in terms of performance, convergence and memory usage for our experiment. Hence, feedforward network with fully connected neurons, log-sigmoid activation function and Levenberg-Marquardt learning was used. The number of input neurons equalled the dimensionality of the data and there was only one output. Different numbers of neurons in the hidden layer were tested. The experiment with neural networks was repeated with reduced dimensionality. Dimensionality reduction was conducted using principal component analysis (PCA) where up to 90% of the information is retained.

Experiments with decision trees and regression were also carried out. The decision tree algorithm used in this experiment was C4.5 by Quinlan [1993]. For C4.5 this experiment used the default values for all parameters except for the confidence factor which determines how heavy the pruning is. The algorithm used for logistic regression is penalised logistic regression with ridge estimator. The default values of all parameters were also used for the logistic regression models.

1.6 Overview of the result

The performance of the different models was tested using 75% of the data for training and the remaining 25% for testing. The data was randomly split into training and testing set. The performance of a model was evaluated in terms of accuracy of classification and the area under the receivers operating characterstics curve (AUC). The AUC proposed by Bradley [1997] which reflect the trade-off between the classifiers' sensitivity and specificity, is a better single number performance indicator.

The results show that SVMs and decision trees performed the best followed by neural networks and logistic regression. The accuracy for SVMs ranged between 94% for a reverse transcriptase inhibitor to 96% for a protease inhibitor. The AUC for SVMs model ranged between 81% to 95% with the lowest AUC related to the lowest accuracy and the highest AUC with the highest accuracy. The accuracy of decision trees was between 91% and 97% and the AUC was between 85% and 94%. In addition to their performance, the decision tree models showed some other interesting results. For each tree, we compared the split criteria with previously reported mutation points. The result showed that most of these split criteria were positions associated with previously known mutations. The result from the decision tree model was also used as a confirmation for the input encoding technique used, and a reference point for the performance comparison. As specified in the previous section, the training data used for the model evaluation was already labelled using a rule-based algorithm. Hence, the internal structure of the patterns are expected to suit decision trees more than SVMs.

The accuracy for the neural network models ranged between 64% to 91%. The AUC for these models was between 69% and 95%. The performance of these models is below expectation. Dimensionality reduction using PCA was performed to enhance the performance. The performance of the models was reduced as a result of dimensionality reduction. The loss in the

performance of the neural network models are attributed to underlying properties of the data. Recall that drug resistance is caused by mutation but not all mutations have the same effect. Some cause resistance alone however, others depend on the occurrence of additional mutation to cause resistance. This primary-secondary relationship between mutations causing drug resistance results in some correlated attributes. Hence, might be a causes the loss of valuable information during dimensionality reduction using PCA.

The performance of the logistic regression models was on average the worst compared to the other models. The accuracy for these models was between 75% and 92% and the AUC for these models was between 46% and 85%.

The performance of SVMs and the decision tree models is almost equal, but, the input encoding technique and the characterstics of the training patterns might favour the decision tree models. Hence, the performace recorded by SVMs with the naive approach has shown a promising start. In general, despite the limitations this research has due to the some easily addressable and other more complicated issues, the result found answers the research question positively.

1.7 Contribution of the research

This research has a number of contributions. Due to the fast replication of the virus, new mutations are occurring almost at an order of billions per day inside an untreated patient. This high number of single point mutations, with the possibility of cross-mutations demands a dynamic system that can cope with the new mutations causing drug resistance and newly discovered drugs to address them. One such dynamic system is support vector machines. Furthermore, to our knowledge, no-one has applied SVMs as a prediction tool in the domain of HIV drug resistance based on the genetic sequence of the virus. Hence, application of such a dynamic system to this problem is the first contribution. Secondly, despite the fact that we used genetically classified data, the design of the SVMs models does not depend on the prior biological knowledge about the virus and the respective drugs. Hence, if trained with phenotypic data, such a model will not require redesigning with the discovery of new mutation points or drugs like most of the existing systems. In addition, the fact that the design of the system is not dependent on the biological data (mutations) makes the intended system capable of handling any kind of behavioural variation of the virus that might lead to drug resistance, provided that this behaviour is reflected on the genetic sequence of the virus. Thirdly, Salzberg [1999] has pointed out the importance of comparing classifiers on real data. HIV is one of the hard computational biology problems that is data-rich. Therefore, the results from this work further highlight the outstanding performance of SVMs as a tool for pattern recognition. Fourthly, we have seen how dimensionality reduction techniques affected the performance of the some of models negatively. This result together with the biological fact about the primary-secondary relationship between different mutations suggests that application of unsupervised dimensionality reduction is not a good idea for HIV drug resistance prediction tools. Finally, the recorded result for SVMs with the two

most popular kernels showed no necessity for the formulation of new kernel functions, however incorporating some general prior knowledge about the virus might boost the performance even further.

1.8 Structure of the document

Chapter 2 provides background information on HIV biology, general principles of pattern recognition and some of the different approaches available. Specifically, details of statistical pattern recognition, Bayesian decision theory and dimensionality reduction methods with particular emphasis on feature extraction methods are discussed. A brief introduction to neural networks, decision trees and logistic regression is also given in this chapter.

Chapter 3 gives a comprehensive background on the core pattern recognition technique investigated in this research (support vector machines). This chapter will start by presenting a detailed discussion on principles of risk minimisation. It then presents the basic formulation of SVMs starting from the simplest form of SVMs (linear SVMs) to the most complex non-linear SVMs. An overview of different implementations of SVMs, a comparison of SVMs to other pattern recognition techniques and previous benchmark applications of SVMs are also considered in this chapter. The discussion of SVMs is based on binary classification problems. However, an overview on multi-class extension of SVMs is covered. This chapter also reviews application of other pattern recognition techniques used in predicting HIV drug resistance.

Chapter 4 starts by motivating this research and emphasises the superior quality of SVMs for the job. The research question for the reported research is then formulated. The chapter further describes the details of the experiment: the input data and the input encoding technique used, the performance evaluation technique and why such a technique was selected and the experimental steps for the different classification algorithms.

Chapter 5 gives the results for the different models and highlights the findings of the different set of experiments. This chapter also compares the performace of the different models to each other and to previously published works. This chapter will also present limitations of this research.

Chapter 6 will present the conclusion for this document. This chapter will also give direction for future work.

The mathematical details of Chapter 3 and some part of Chapter 2 are presented in the Appendices.

Chapter 2

Background and Related Work

2.1 Introduction

In the previous chapter an overview of the research was presented. As presented in the previous chapter, the main objective of the reported work is investigating the performance of SVMs in predicting the drug resistance behaviour of an HIV strain extracted from a patient based on the genetic sequence of the viral RT or PRO mutants.

To understand the terms and methodologies used in this research enough background on the biology of the virus, the general principles in pattern recognition tasks and the pros and cons of different pattern recognition technique should be established. Hence this chapter is intended to give a brief background on HIV biology, different pattern recognition techniques with more emphasis on statistical pattern recognition and some diagonal factors on solving pattern recognition problems. Furthermore, this chapter is used as an introduction for Chapter 3 where SVMs are discussed in great detail.

This chapter is organised as follows. Section 2.3 covers the necessary biological background on the HIV and drug resistance problem. This section starts with general introductory biology and then describes the phylogenetic information, the genetic structure and viral replication cycle. This section also discusses how drug resistance occurs and the different approaches of predicting drug resistance. Section 2.4 gives a historical overview of pattern recognition techniques and describes the different techniques available. Section 2.5 gives a detailed background on statistical pattern recognition, Bayesian decision theory, dimensionality problem and techniques for dimensionality reduction. Section 2.6, 2.7 and 2.8 give detailed background on neural networks, decision trees and logistic regression respectively. Finally section 2.9 concludes the chapter.

2.2 Introductory Biology

The basic biological process at a cellular level is identical across organisms. The hereditary genetic information of a living organism is stored in the deoxyribonucleic acid (DNA) or ribonucleic acid (RNA). The units of information that are assembled together to form DNA or RNA are four nucleic acid units called nucleotides (bases). Adenine (A), guanine (G), cytosine (C) and thymine (T) make up DNA and the nucleotide thymine is replaced by uracil (U) in case of RNA. DNA and RNA, which are a polymer of these nucleotides have other common characteristics beside being a chain of nucleotides. One of these property is complementarity. This is the exclusive bonding rule between adenine and thymine/uracil (A-T/U) and guanine and cytosine (G-C) [Hunter 1993].

While RNA is a single stranded, DNA is double strand of bases in a double helix form. Each of these strands are often millions of bases long and its direction is determined by its head (5' end) and tail (3' end)[1]. The sequence of nucleotides in one of the helices is unrestricted. However, due to the exclusive bonding between nucleotides, the sequence on the complementary strand is completely deterministic. A strand of DNA or RNA which is an exact complement of another strand are called reverse complement to the second [Hunter 1993]. (For example, ATGCCA is the reverse complement of TACGGT).

The DNA contains genes of an organism, which are used as a template for manufacturing RNA which then will be used to manufacture protein. The primary role of the nucleic acids is to carry the encoding of the primary structure of protein. Proteins determine the shape and structure of a cell. Each non-overlapping triplet of nucleotides in the DNA strand are called codon. Each codon corresponds to a particular amino acid. Grouping of the four nucleotides accordingly results in $4^3 = 64$ possible triplets encoding 20 amino-acids and three special duty codons called stop codons [Hunter 1993] (see Table C.1 in Appendix C for complete list of amino acids and the codons encoding them). However, not all triplets in the genome encode for protein. In higher organisms approximately 97 – 98% of the genome is a non-coding sequence called *introns*. And the remaining 2 – 3% called *exons* codes the proteins [Watson *et al.* 1997].

The mechanism by which proteins are produced from DNA is a sequence of steps, which are known as the central dogma of molecular biology [Watson *et al.* 1997]. Generally, these steps can be summarised in two steps as transcription and translation. The process starts with unwinding the helix into a separate strand. From a single strand of DNA as a template the RNA is manufactured. However, not the entire DNA is used to manufacture the RNA. As we have stated in the above paragraph, only exons are genic. This process is called transcription, because RNA is transcribed from the DNA. The RNA is then translated into protein which determined the structure and function of an organism. Translating the RNA into protein starts from the codon which encodes the amino acid methionine (AUG). From then on each codon is

[1]DNA molecules are directional, due to the asymmetric structure of sugar which constitute the skeleton of the molecule. Each sugar is connected to the strand in its fifth carbon preceding it in the chain and in its third carbon following it in the chain

translated into the corresponding amino-acid and is added to the growing chain of amino-acids. This process is then stopped when one of the stop codons is encountered. However, there are some complications to the translation process. Depending on where we start encoding, we have three possible reading frames on each direction. (i.e. 6 open reading frames) [Hunter 1993]. For example, the following sequence ACTGAAGTCGCCA... can be read as ACT-GAA-GTC-GCC-A... or CTG-AAG-TCG-CCA-... or TGA-AGT-CGC-CA..., all of these making the different reading framed. Usually only one of these frames will produce a functional protein. However, this is not always true. Therefore, identifying the correct reading frame is the primary task in many computation biology problems.

2.3 HIV Biology

2.3.1 Introduction

Viruses are a group of submicroscopic infectious agents, unable to replicate outside a host cell. These submicroscopic organisms essentially contain their genetic material in terms of DNA or RNA surrounded by a protein coat. During replication, the virus integrates its DNA or RNA into the host's DNA and takes over the cell's biological mechanism to replicate [NIAID 2004]. If the virus contains its genetic material in the form of RNA, it should be first transcribed into DNA before integrating with the host's DNA. In most organisms, RNA is transcribed from DNA and hence, those viruses that contain their genetic material in terms of RNA are called retroviruses to indicate the reverse transcription of RNA to DNA.

This section is intended to give an overview on the biological background of HIV, which is a retrovirus. The section is organised as follows. Section 2.3.2 gives an overview on phylogenetic information of the virus and the structure and function of some of the major genes in the viral genome. It is followed by section 2.3.3 which describes the viral replication process at a very high level. Finally section 2.3.4 starts with the basic definition of drug resistance and gives the reason why HIV drug resistance occurs and describes the two general approaches of HIV drug resistance testing.

2.3.2 HIV phylogeny and genome

The HIV is a retrovirus belonging to the genus[2] *Lentivirus*, which shares many important characteristics with other retroviruses but also has some special features [Coffin 1999]. Besides HIV, the genus *Lentivirus* includes Simian Immunodeficiency Virus (SIV), which is phylogenetically closely related to HIV and the distant relatives: Vesina Virus and Feline Immunodeficiency Virus (FIV) [Hoffmann and Kamps 2003]. HIV is divided into two subtypes namely: HIV-1

[2]"A taxonomic category ranking below a family and above a species and generally consisting of a group of species exhibiting similar characteristics. In taxonomic nomenclature the genus name is used, either alone or followed by a Latin adjective or epithet, to form the name of a species".(The American Heritage Dictionary)

and HIV-2 based on the molecular weight of their protein and their subordinate genes [Hoffmann and Kamps 2003]. Despite the strict resemblance between HIV-1 and HIV-2, HIV-1 is the most common subtype and is the main infectious agent that has led to the worldwide AIDS epidemic. Even though HIV-2 infection is less common and less virulent, it also results in AIDS [Klatt 2003]. Furthermore, although HIV-1 and HIV-2 replicate in the same fashion the actual immune deficiency may be less severe in HIV-2 infected individuals [Hoffmann and Kamps 2003]. Some 99% of HIV patients are infected with HIV-1 with a growing number being infected by HIV-2 [Draghici *et al.* 2000].

HIV-1 is classified into two principal genetic groups designated M (main) and O (outliers). Genetic group M is highly prevalent and is further classified into 10 established subtypes, A through J. HIV-1 subtype B predominates in Europe and the Americas, whereas HIV-1 subtype C predominates sub-Saharan Africa [Klatt 2003]. An additional group N (non-M, non-O) has been discovered recently as a result of interaction between the two principal groups [Klatt 2003; Health Canada 2001]. In the remainder of this document HIV refers to HIV-1 unless otherwise specified.

Physically, an HIV viral particle has a diameter ranging approximately from 90 to 100 ηm and is surrounded by an envelope, which encapsulates the genome that encodes the major functional and structural components of the virus [Coffin 1999]. The HIV genome contains two single stranded RNA molecules each 9 kilobases in length. These RNA molecules contain 9 different genes encoding 15 different proteins [Greene and Peterlin 2003]. The HIV genome is classified into three major classes namely: structural genes (gag, pol and env), trans-activation genes (tat and rev) and accessory genes (nef, vif, vpr and vpu) [Hoffmann and Kamps 2003; Klatt 2003]. Like all retroviruses, the major genes *gag-pol-env* are contained in the genome in the conserved order *5'-gag-pol-env-3'* [Coffin 1999; Klatt 2003]. The schematic diagram in Figure 2.1 shows the viral genome of HIV-1 based on the diagram from `http://hiv-web.lanl.gov/immunology/pdf/2000/intro/GenomeMaps.pdf`.

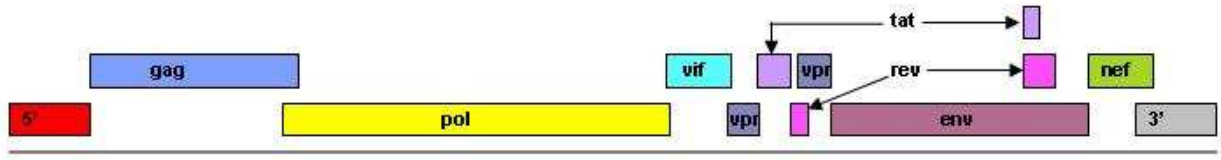

Figure 2.1: Schematic diagram of HIV-1 genome

The structural genes (*gag-pol-env*) are common to all retroviruses. The major components encoded by *gag* (group-antigen) include the nucleocapsid proteins capsid (CA), matrix (MA), and nucleocapsid (NC). The *env* (envelope) gene encodes the envelope glycoproteins, outer envelope glycoprotein and transmembrane glycoproteins. The *env* proteins of HIV have a number of distinct structural and functional features that enable the virus to replicate efficiently under the threat of the host's immune response and which not seen in other retroviruses. The *pol* (polymerase) gene encodes the enzymes reverse transcriptase (RT), protease (PRO) and integrase (IN). These enzymes are the main stakeholders in the viral replication process and hence

are major target of most antiretroviral drugs [Coffin 1999; Klatt 2003]. The other classes of genes are also critical in RNA transcription and viral release during the viral replication process [Klatt 2003; Hoffmann and Kamps 2003].

2.3.3 HIV replication

The replication of the virus within the host body is presented schematically in Figure 2.2. It is already stated above HIV virus is unable to replicate outside a living host cell. Therefore, viral replication starts with the virus entering the host cell [Coffin 1999]. This step of the viral replication is mediated by the *env* protein that interacts with a specific cell surface receptor. Once the viral core enters the cell, the genome RNA is reverse transcribed by the HIV reverse transcriptase into a double-stranded DNA molecule. The newly made HIV DNA is then moved to the cell's nucleus and integrates with the host cell's DNA aided by the HIV Integrase. At this step the viral DNA is called "provirus". This provirus uses the host cell's protein-making machinery to produce new copies of RNA called messenger RNA (mRNA). Once these mRNAs are processed in the cell nucleus, they are transported to the cytoplasm aided by proteins encoded by the *rev* gene. In the cytoplasm the HIV mRNAs are used to make long chains of viral proteins and enzymes with the help of the host protein making machinery and ribosomes. The newly made HIV core protein, enzyme and RNA will then gather inside the cell's membrane, while the viral envelope protein aggregates within the adjacent membrane. Just before the new virus exits the cell, the long chain of proteins and enzymes that make up the immature viral core are cleaved into smaller pieces by a viral enzyme called Protease. Finally, the virus will be assembled and detaches itself from the host cell. This results in a new infectious viral particle [Coffin 1999].

An animation of the HIV life-cycle is available at `http://www.hopkins-aids.edu/hiv_lifecycle/hivcycle_txt.html`

2.3.4 HIV drug resistance and assay of drug resistance testing

Once the HIV virus enters the human body it begins to replicate at a very high rate in the order of billions everyday. During replication, HIV produces perfect copies and copies containing errors (mutated virus). Mutation is very common in HIV because of the high rate of viral replication and RT infidelity which is largely due to the lack of 3'– to – 5' proofreading ability within viral RT [Whitney *et al.* 2002]. As a result of the changes in HIV's genetic structure (mutation), the ability of a drug or a combination of drugs to block HIV replication inside the body is reduced. This phenomenon is called drug resistance.

As it is stated in the previous section besides RT another enzyme which plays an important role in viral replication and the resulting drug resistant mutants is the HIV PRO. During viral replication, the cell produces a long strand of genetic material that must be cut up and put together correctly to form new copies of the virus. Cutting up this long strand is carried out by the enzyme PRO. Furthermore, PRO is responsible for processing the *gag* and *pol* genes,

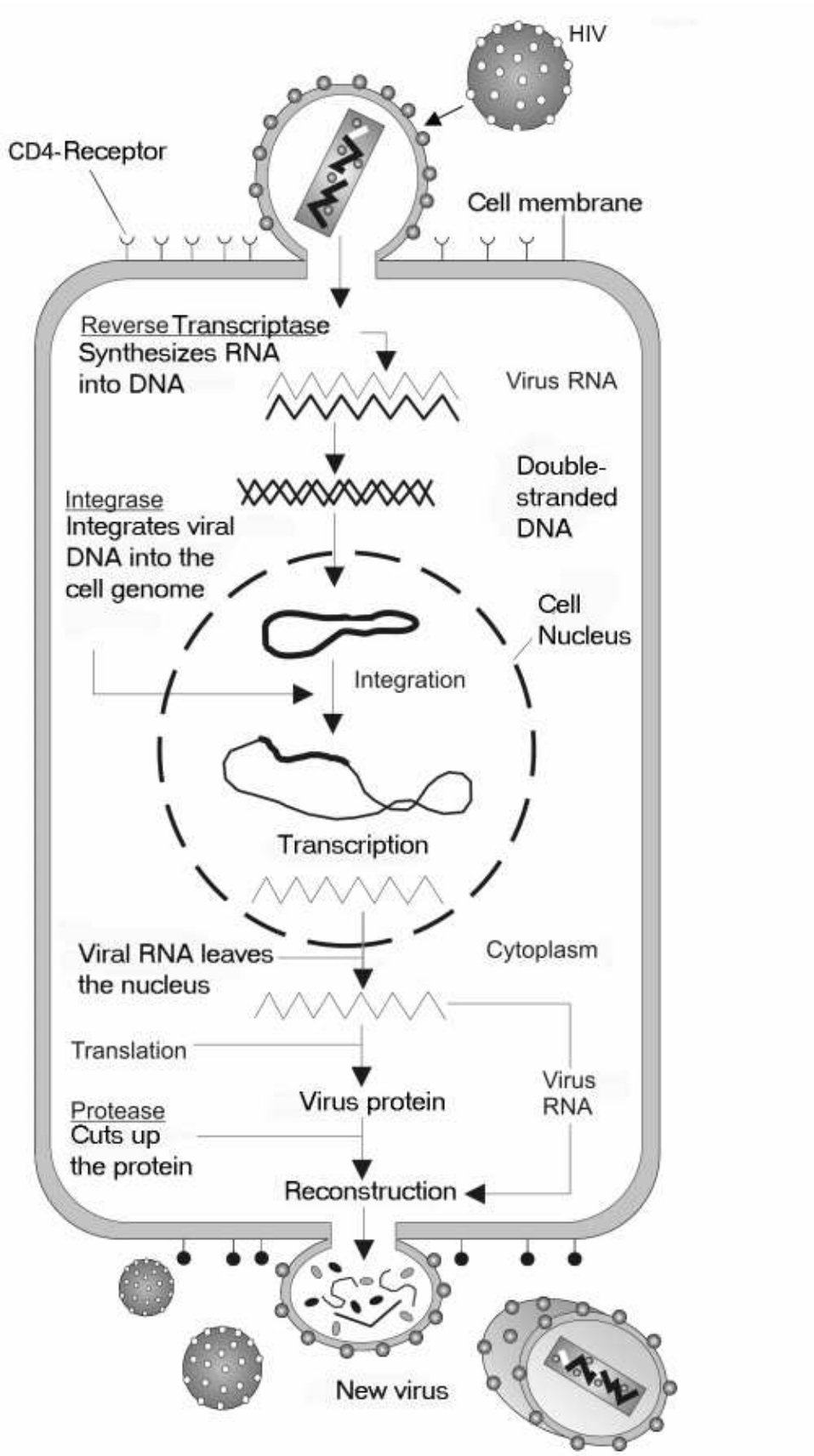

Figure 2.2: Viral replication cycle [Wikipedia 2004]

which are initially expressed as the precursor polyproteins *Gag* and *Gag-Pro-Pol*, into their mature stage immediately after budding [Shafer 2002a; Coffin 1999]. Due to their important role in the viral replication these two proteins are the targets of most of the existing HIV drugs. Furthermore, most mutations are exhibited in these proteins [Lathrop *et al.* 1999; Shafer 2002b]

To date, there are more than eighteen antiretroviral drugs available. These drugs are generally classified in to three major categories based on the enzyme targeted, the viral replication stage interfered and their chemical composition [Hoffmann and Kamps 2003]. The first class of drugs are Nucleoside/Nucleotide RT Inhibitors (NRTIs), which interfere with the viral replication stage by blocking the further elongation of the proviral DNA and interrupting the chain. There are seven NRTIs. The second class of drugs are Non-Nucleoside RT Inhibitors (NNRTIs). NNRTIs interfere with the viral replication state the same way as the NRTIs. There are three NNRTIs. The third major category are PRO Inhibitors (PIs). PIs function by interfering with the viral assembly stage of the replication. There are eight PIs. The fourth drug category are fusion inhibitors (FI). Fusion inhibitors are members of a broader class, the *entry inhibitors*, which stop the virus from entering the cell by preventing the final phase of attachment [Beerenwinkel *et al.* 2003b]. There are also some more experimental drugs in each class [Hoffmann and Kamps 2003]. A complete list of antiretroviral drugs is given in Table 2.1. For up to date information visit on-line HIV/AIDS information services such as `http://www.hopkins-aids.edu`.

Drug	**Abbreviation**	**Target**	**Class**
zidovudine	ZDV	RT	NRTI
didanosine	ddI	RT	NRTI
zalcitabine	ddC	RT	NRTI
stavudine	d4T	RT	NRTI
lamivudine	3TC	RT	NRTI
abacavir	ABC	RT	NRTI
tenofovir	TDF	RT	NRTI
nevirapine	NVP	RT	NNRTI
delavirdine	DLV	RT	NNRTI
efavirenz	EFV	RT	NNRTI
saquinavir	SQV	PRO	PI
indinavir	IDV	PRO	PI
ritonavir	RTV	PRO	PI
nelfinavir	NFV	PRO	PI
amprenavir	APV	PRO	PI
lopinavir	LPV	PRO	PI
atazanavir	ATV	PRO	PI
T-20		gp41	FI
T-1249		gp41	FI

Table 2.1: Antiretroviral Agents [Beerenwinkel *et al.* 2003b, page i19]

The extreme genetic and antigenic variability of HIV, which results from the development of drug resistant viral strains, is the most common reason for the failure of HIV drug therapy [Draghici *et al.* 2000]. Although there are many drugs available, none of these drugs have been

able to stop the viral replication totally. Despite the failure to produce an efficient drug to stop the viral replication, much has been done in combating the pandemic [Shafer 2002b]. One such achievement worth mentioning is the prolonging of the viral suppression and help with immunological reconstruction of the patient using combinational therapy. Furthermore, these successes have opened a new era of HIV therapy called highly active antiretroviral therapy (HAART) [Hoffmann and Kamps 2003]. HAART works based on pre-therapy drug resistance testing of the HIV strain extracted from the patient and giving the patient a combination of drugs containing one protease inhibitor. Due to the intolerable side-effects and toxicity, combination of drugs can not exceed four (combination of three drugs is the common one at least one of the three being a PI) [Lathrop and Pazzani 1999]. Hence a very careful measurement of drug resistance is required.

There are two ways of HIV drug resistance testing namely: phenotypic testing and genotypic testing [Bean 2000].

Phenotyping

Phenotypic testing directly measures the drug resistance behaviour of an HIV strain. The HIV strain from the patient is placed in a test-tube and the growth of the virus is closely studied under a treatment of the drug or combination of drugs by varying the concentration and strength. The measured viral replication is then compared to the wild type [Hoffmann and Kamps 2003].

Phenotypic testing has some disadvantages. Firstly, the process is time consuming and very expensive. Secondly, it requires a specialised laboratory. Furthermore, drug resistance of an HIV strain cannot be detected when the viral load is less than 20% [Bean 2000].

Genotyping

Genotypic testing is based on analysing certain mutations associated with drug resistance based on the genetic structure of the HIV strain extracted from the patient [Hoffmann and Kamps 2003; Bean 2000]. To conduct genotypic testing the contiguous PRO and RT genes which are extracted from the plasma are reverse transcribed to cDNA. The cDNA is then amplified using polymerase chain reaction (PCR) to generate sufficient DNA [Shafer 2002b]. This genetic sequence is then examined carefully for mutation. Depending on the number and type of mutations exhibited the test reveals whether the patient has developed resistance to a certain drug or combination of drugs. For example, if the mutation M184V/I is detected[3], the HIV strain will be resistant to the NRTI *Lamivudine* (trade name Epivir) [Hoffmann and Kamps 2003]. For explanation about M184V/I revisit Section 1.4.

Genotyping is more widely used than phenotyping. Some of reasons is that unlike phenotyping, genotyping does not require a specialised laboratory, it is cost effective, it is fast and

[3]Resistance mutation is described by using a number showing the position of the codon where the mutation occurred and two letters. The letters preceding the number represents the amino acid in the same position of the wild-type and the letter after the number shows the amino acid produced by the mutation. [Hoffmann and Kamps 2003]

interpreting the result does not depend on the testing process. Furthermore, the fact that it is totally based on the genetic sequence of the HIV strain has opened the door for leading technologies from other fields of study to contribute. One such technology is pattern recognition and artificial intelligence.

There are a number of commercially available tools to conduct genotypic resistance testing. But they have a number of limitations. Firstly, certain mutations cause resistance by themselves while others need to occur in the existence of others to cause resistance. And different genotypic tools treat these mutation points differently and hence leads to discordance between the different tools [Ravela *et al.* 2003]. Secondly, interpretation of the result of genotypic testing is difficult [Bean 2000]. Thirdly, most of these tools use knowledge-based approach and are highly dependent on the known biological facts about the virus and the drug. Hence, these tools will not be able to perform the required task without continuous update or even require redesigning with discovery of new drugs and mutation points known to confer resistance to the existing or newly discovered drugs. For the above mentioned reasons and more, genotypic testing is open for more research and collaboration for other disciplines.

2.4 Pattern recognition

Pattern recognition is an interdisciplinary field of study developed mainly in the 1960s covering developments from a wide variety of disciplines ranging from psychology and physiology to computer science and artificial intelligence [Webb 1999]. Since then scientists have been investigating ways to enable machines to recognise pattern the same way humans do to base their decision-making process on their daily life. According to Pavlidis [1977] pattern recognition is defined as understanding the building blocks of a given object. Tou and Gonazalez [1974] also defined pattern recognition as classification of an input data into a category called a pattern class, which is determined by some given common attributes based on exhibited patterns. A pattern is defined as the description of any member representing a pattern class. A pattern can be as basic as observation and measurements [Tou and Gonazalez 1974; Schalkoff 1991]. Examples of a pattern could be a DNA/Protein sequence, a text document, handwritten characters, or a signal waveform.

Pattern recognition is a computationally expensive task; therefore it has been a specialised subject in the past and applications were limited to certain domains. However recent advancements in computer hardware (processor speed and storage) have made pattern recognition widely applicable in range of application areas (see Table 2.2) [Jain *et al.* 2000]. Although the applications presented in table 2.2 are diverse, one can list a number of common properties between these applications [Jain *et al.* 2000]. One common property is that the data for each problem is represented by a very big set of parameters (features). Another common property is that the features that represent the data cannot usually be defined by the domain expert, but should be extracted from the exhibited patterns.

Independent of the application domain, a pattern recognition task consists of a number of

iterative steps. The first step is data collection and preprocessing. This step includes recording measurements and/or observations, extracting and/or selecting the most representative features and high level examination of the data to get some idea about the underlying distribution, probability, structure, etc., of the data. The second step is designing the pattern recognition system and performing the training. The pattern recognition system is usually designed based on the properties of the sample data and is dependent on the previous step. Sometimes it might be impossible to design the pattern recognition system and hence the intended system will be the result of the training process. Therefore careful representation and preprocessing is required for the performance of the system. The final step is testing the performance of the designed system and interpreting the result. Note that each step listed above can be further broken down into a number of steps.

Problem Domain	**Application**	**Input Pattern**	**Pattern classes**
Bio-informatics	Sequence analysis	DNA/Protein sequence	Known types of genes/patterns
Data Mining	Searching for meaningful pattern	Points in multidimensional space	Compact and well separated clusters
Document Classification	Internet search	Text document	Semantic categories (eg. business, sport, etc.)
Document image analysis	Reading machine for blind	Document image	Alphanumeric characters, words
Industrial automation	Printed circuit board inspection	Intensity or range image	Defective or non - defective nature of product
Multi media database retrieval	Internet search	Video clip	Video genres (eg. action, dialogue,etc.)
Biometric recognition	Person identification	Face, iris, fingerprint	Authorised users and access control
Remote sensing	Forecast crop yield	Multi spectral image	Land use categories, growth pattern of crops
Speech recognition	Telephone directory enquiry without operator	Speech waveform	Spoken words

Table 2.2: Examples of Pattern Recognition applications [Jain *et al.* 2000, page 5]

Based on the training approach, a pattern recognition task can be categorised as **supervised or unsupervised** pattern recognition. In supervised pattern recognition, the pattern recognition system is given a set of examples (input-output pairs) as a training set. This training set is usually of the form of attribute vectors, and is a subset of $\mathbb{R}^n$. Given the attribute vectors, we can synthesise the value of a mapping function for some samples in the training set and choose a set of hypotheses for the problem. On the contrary, in unsupervised pattern recognition, there is no output value associated with the inputs and the recognition task is to get some understanding of the process that generated the input so as to classify the training set's example into their respective classes according to their properties and also to generalise for unseen inputs. In the

remaining part of this document pattern recognition refers to supervised pattern recognition unless otherwise specified.

Another way of classifying pattern recognition approaches is based on some properties of the data and assumptions taken. The preprocessing step not only provides us with a concise representation of the data but also crucial information that helps us select the right pattern recognition approach to solve the problem at hand. This step sometimes provides the underlying and statistical basis of the patterns or the underlying structure, which is critical to pattern recognition. As in other cases none of the above information can be provided, all the necessary information for the desired system should therefore be investigated during the training process. Thus, based on the information available one approach might be better than another to solve a particular problem. In summary, based on the underlying principle, data representation and assumptions made, pattern recognition can be crudely categorised into four main practical approaches [Jain *et al.* 2000; Schalkoff 1991]. These categories are:

- Template matching,
- Syntactical/structural pattern recognition,
- Statistical pattern recognition,
- Neural pattern recognition (Neural Networks).

Each of these approaches are described below

There are other approaches that do not fit particularly into any of the above categories [Schalkoff 1991]. Such approaches include combinations of statistical and syntactical pattern recognition, reason-driven pattern recognition where artificial intelligence is used to infer some rules based on the training data, etc.

Template matching is conceptually the simplest and the earliest form of pattern recognition. In some applications the pattern under investigation is almost identical to some prototype of the pattern class. This prototype is called a template. A template might be a certain object in the problem domain or a string of patterns. Therefore, the pattern recognition problem is reduced to matching the unknown pattern with these templates and finding the best match. In other situations the templates may also be contained within the unknown pattern, therefore the pattern recognition task may also involve determining the relative positions of these templates. The performance of this pattern recognition approach depends on the quality of the similarity measure used. There are a number proposed similarity measurements to determine the best match between the known pattern (template) and the unknown pattern. Such measurements include edit distance, sum-of-squares difference and the maximum-likelihood formulation proposed by Olson [2000]. More on similarity measures can be found in Schalkoff [1991].

Algorithms based on template matching are easy to implement but suffer from low recognition performance due to distortion, view point change or large interclass variation among the

patterns [Jain *et al.* 2000]. This can often be addressed by incorporating some pre-processing techniques such as rotating the stimulus, scaling, etc. to make it upright, of a standard size, etc. respectively.

Syntactical/structural pattern recognition: In many real world problems,the structural and relational information contained in the patterns makes identification of quantifiable features that can be represented in a vector form difficult or sometimes impossible [Schalkoff 1991]. Some examples of pattern recognition problems satisfying these characteristics are picture recognition, time-series analysis, text recognition. One common characteristic of these patterns is that some kind of inheritance or identifiable organisation is usually exhibited [Olszewski 2001]. Syntactical pattern recognition is not only used for classification of patterns but also description of patterns [Fu 1974]. For example, in some pattern recognition problems, the structural information of the pattern is so important that classification of the pattern might not be enough. It might thus be necessary to describe the property of the pattern which makes it eligible to be classified in a certain way. As another example, some patterns like fingerprints are self-dependent and the number of possible descriptions are extremely large, hence classifying each pattern into its own class is impractical. Therefore, the task of such a pattern recognition system involves describing the pattern rather than classifying it.

The complexity, inheritance and identifiable organisation of the patterns, have motivated many syntactic methods to adopt a hierarchical perspective where these complex patterns are considered as a composition of simple sub-patterns and hence is hierarchically decomposed into simpler patterns [Fu 1974; Schalkoff 1991]. The simplest sub-patterns are usually referred as *primitives* and the relationship among them represents the structural features of the pattern [Olszewski 2001]. Primitives can then be quantified using formal grammar or relational descriptions (usually graphs) to facilitate recognition, classification or description of these patterns [Schalkoff 1991].

A **statistical approach** is based upon a statistical analysis of the data to be classified. The data are assigned to a particular class by computing class-conditional densities of the data, which is represented as a d-dimensional feature vector. The d-dimensional vector space is then divided into regions corresponding to the different class based on some criterion. Statistical pattern recognition will discussed in more detail in Section 2.5.

Neural pattern recognition is a computational system inspired by the learning characteristics and the structure of a biological neural network. The key element of this approach is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in complete harmony to solve specific problems. Neural pattern recognition will discussed in more detail in Section 2.6.

2.5 Statistical pattern recognition

In statistical pattern recognition, a pattern is represented by a set of features (say a set of *d* features) obtained through observations/measurements and conveniently viewed as a *d*-dimensional feature vector $X = (x_1, x_2, \ldots, x_d)$. The basic assumption in this approach is that there exists a multivariate class-probability distribution which can be inferred from this exhibited random patterns [Jain *et al.* 2000; Kanal 2000]. With this assumption, the problem of statistical pattern recognition can be formulated as follows: given a set of measurements representing the pattern as *d*-dimensional feature vectors, the purpose of the pattern recognition system is to classify these unknown patterns to one of the known (say *c*) pattern classes [Webb 1999; Jain *et al.* 2000]. Classification is done by subdividing the space spanned by these feature vectors into c disjoint (non-overlapping) regions, where each disjoint region represents a particular class [Schalkoff 1991]. The vector representation of a pattern also assures that a pattern can only be plotted to a point in the feature space and hence can be assigned to one class based on the region to which it is plotted.

Like all other pattern recognition approaches, statistical pattern recognition is also carried out in a number of sequential steps which can be divided into two major steps: training (learning) and classification (testing) [Jain *et al.* 2000]. Each of these are further broken down into a number of modules. The two major steps and the modules in each step are schematically presented in Figure 2.3.

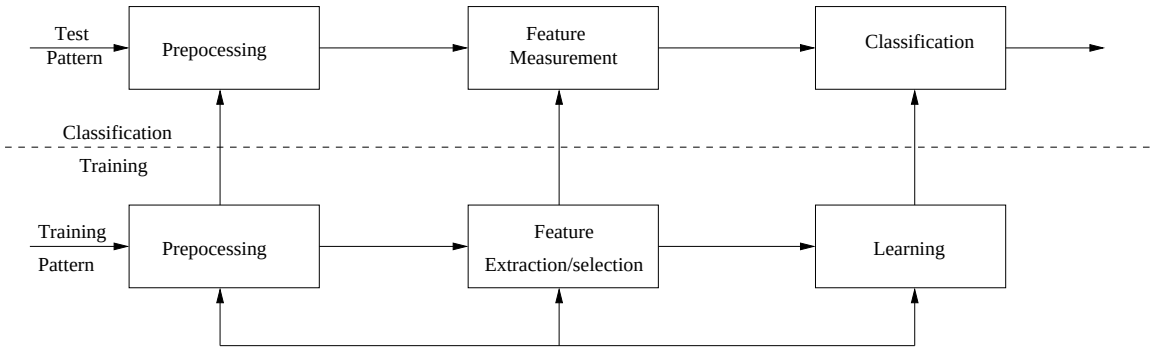

Figure 2.3: Model for statistical pattern recognition [Jain *et al.* 2000, page 8]

The first module is the preprocessing module. This module is responsible for the compact representation of each pattern. This includes removing noise, segmenting, normalising the pattern and other related operation. The preprocessing module is the same for both the training and the classification mode of the recognition system. Although this module gives a compact representation of the data, it does not necessarily give an effective one. Despite the intuition of considering a large number of parameters (features) to characterise a pattern in order to obtain a better recognition, in statistical pattern recognition having a large number of features to represent a pattern does not usually result in minimum classification error. To resolve this contradiction, it is necessary to find the representative features by taking into consideration the number of samples available and the correlation between the features. A module which is re-

sponsible for this sub-task and which is located in the training mode is feature extraction and/or selection module. This module is recursive and the feedback path shown in the figure (Figure 2.3) allows the designer to optimise the process.

The primary aim of building a pattern recognition system is to build a set of decision boundaries that can classify unseen patterns based on the finite set of sample patterns. Furthermore, Jain *et al.* [2000] have pointed that the performance of a classifier does not only depend on the complexity of the classifier but also on the interrelationship between the available sample size and the number of features targeted. For a classification task with arbitrarily large training sample which is representative of the underlying distribution of the pattern, increasing the number of features to characterise the pattern will not have a negative effect on the performance of the classifier [Jain *et al.* 2000]. However, in real world pattern recognition problems the number of available samples is limited. Moreover, Duda *et al.* [2000] have commented that increasing the number of features for a finite sample size does not result in small classification errors but may rather reduce the performance. This comment is further supported by Trunk [1979] with the help of examples. This property has put a constraint on the number of features that one can consider for a finite number of training samples. Although there is no defined rule or guide to solve this problem there have been a number of recommendations and guide lines given over the past couple of decades. For classifiers which are based on partitioning the feature space into regions, the number of training samples should be an exponential function of the number of features considered. However, Jain and Chandrasekaran [1982] have stated that it is generally acceptable to have the ratio between the number of training samples and the number of features per pattern class to be greater than ten. This ratio must be higher for more complex the classifiers. More comprehensive discussion on this topic is covered in Raudys and Jain [1991] and Jain and Chandrasekaran [1982].

There are a number of approaches to obtain a comparable ratio between the sample size and features targeted such as dimension reduction. The two techniques used in dimension reduction are feature selection and feature extraction. Feature selection is a process of selecting a subset of the features that effectively represent the pattern. Feature extraction on the other hand is the process of finding an arithmetic combination of the d-dimensional feature vector in relation to a lesser dimensional vector. Dimension reduction will be discussed in more detail in section 2.5.2.

The last step and the final module in statistical pattern recognition is the classification module. This module classifies patterns based on the class-conditional probability function which is obtained during the training process. As stated above, classification is achieved by subdividing the feature space into a number of disjoint regions. These decision boundaries are defined using Bayes decision rule [Jain *et al.* 2000]. Bayesian decision theory is a fundamental statistical approach to statistical pattern recognition problem. This approach formulates the classification problem in terms of the probability density function. In the next section an introduction to Bayesian decision theory is presented. A detailed presentation of this topic can be found in Appendix A.1. The presentation of these sections is based on the work by Duda *et al.* [2000]

and Webb [1999]. The notations are after Duda *et al.* [2000].

2.5.1 Bayesian decision theory

Bayesian decision theory is the basic concept behind statistical pattern recognition techniques. Given a random pattern x and a k class pattern recognition task, Bayes' theorem answers the question: what is the probability that the observation x belongs to the pattern class ω_j?

The Bayes rule describes this probability as:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \tag{2.1}$$

where $P(\omega_j|x)$ is the posterior probability, $P(\omega_j)$ is the prior distribution, $p(x|\omega_j)$ is the state-conditional probability and $p(x)$ is the probability density function for x.

Note that, the posterior probability in equation 2.1 is going to be calculated from other probability functions that are easy to calculate from the given training samples. Once this probability is estimated the pattern will be assigned to the pattern class with the highest probability. ie.

$$x \rightarrow \omega \quad \text{if} \quad P(\omega|x) = \max_{\omega_j} P(\omega_j|x) \tag{2.2}$$

(For compete presentation of Bayesian decision theory see Appendix A.1)

Bayes decision rule assumes that all the probability functions are defined which is not true in most real-life pattern recognition problems. However in many of these classification problems one can assume the form of the class-conditional density (eg. multivariate Gaussian). Based on this assumption statistical classifiers will be divided into parametric and non-parametric classifiers. Some examples of parametric classifiers are linear and non-linear discriminant functions. Some examples of non-linear classifiers are k nearest neighbourhood and multilayer perceptron.

So far we have seen that given the probabilistic densities or taking some assumption of about the probability density we will perform classification of unseen patterns based in equation 2.2. However, there are a number of attributes that can affect the performance of the classifier specially when probability density estimation is involved. One such attribute is the proportionality between the number of features and available training samples. If the number of features is too large relative to the number of training samples, the classifier is likely to perform badly. To avoid this problem there are a number of techniques. One of the most acknowledged approach is dimensionality reduction discussed below. Note that, even though this problem is most likely in pattern recognition techniques that rely on some probability estimation, it also occurs in other pattern recognition techniques.

2.5.2 Dimensionality reduction

The dimension of the data is the number of features that represent each observation or simply the size of the vector reperesenting the observation. High dimensional data has presented statistical pattern classification systems with many challenges and new perspectives to the problem. Based on a simple naive intuition one could argue that classification error can be decreased by increasing the number of features. However, this is not true for a number of reasons. Firstly, high dimensional data will have a great demand of computational resources. Secondly, with high dimensional data, it is very hard to understand the underlying structure of the data which thus degrades the classifier's performance. Besides the above mentioned reasons, the importance of maintaining the ratio between the dimensionality of the data and the number of samples to increase classifier performance is discussed in section 2.5. Therefore, dimension reduction is important due to measurement and computational cost and classification accuracy. To see further the relationship between the number of features and sample size and the importance of dimensionality reduction, consider the 3-class pattern recognition problem of classifying three different geometric shapes (circle, square and triangle) adopted from Gutierrez-Osuna given in Figure 2.4. A simple approach is to divide the feature space into a number of uniform bins and compute the ratio of examples for each class at each bin as shown in Figure 2.4 (a). When a new object is found, the object will be placed in the feature space and choose the predominant class (geometric shape) in the bin it is placed. It can be seen from this figure only one feature is considered. Moreover there is too much overlap and hence the performance will be low. To address this problem we need an extra feature to represent the pattern. So lets consider a two and three features as shown in Figure (2.4 (b)) and (2.4 (c)) respectively.

When the number of features increases from one to two, the number of bins increases from 3 to 9 (3^2). Here, we have to make a decision whether to maintain the density of example per bin or keep the number of examples constant. In the first case we need to increase the number of examples from 9 to 27 and the latter case results in sparse 2D scatter plot. Furthermore, when the dimension is increased to three the problem become worse. The number of bins grows to 27 and if the density needed to maintained 81 examples are required and on the other hand, if the number of examples are kept constant the 3D scatter plots are almost empty. Although this is a trivial example we have been able shown the problem of dimensionality that exists almost in all pattern recognition problem. In this example, the number of training sample (geometric shapes) should grow exponentially as the number of features considers. But increasing the features increasing continuously for a fixed sample size does not improve the performance, rather it starts to degrade the performance. This phenomenon is termed *Curse of Dimensionality* [Theodoridis and Koutroumbas 1999]. In practise, this means that, for a given sample size, there is a maximum number of features above which the performance of the classifier starts to degrade rather than improve. In the last decade much research have been carried out to solve this problem and come up with different approaches. One way is dimensionality reduction.

Dimensionality reduction can be described as determining a subset or combination of features that represent the pattern without losing the class discriminatory information. There are a

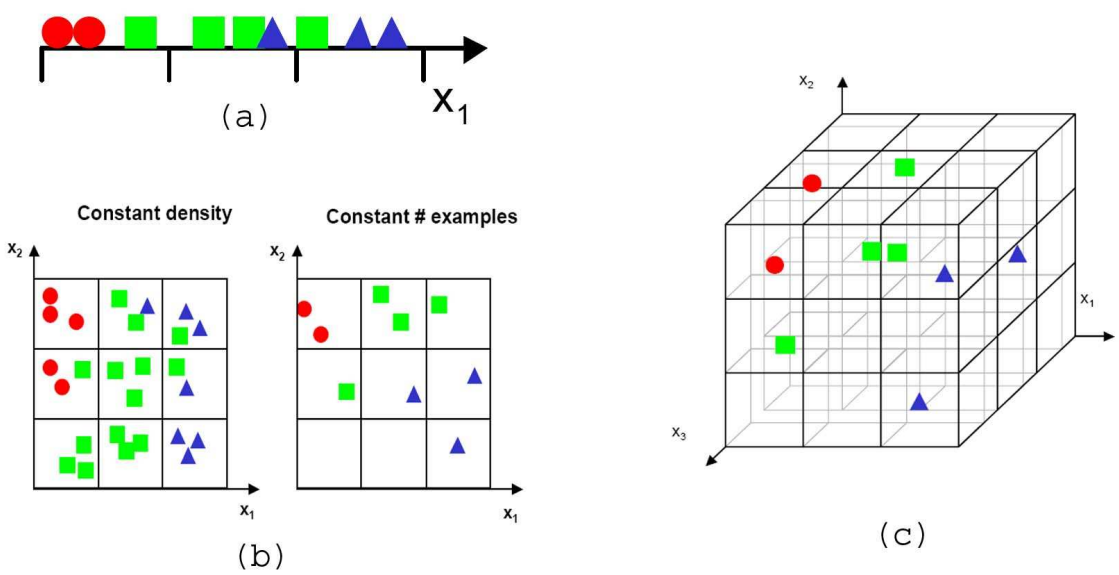

Figure 2.4: Number of bins required for different features (a) one feature (b) two features (c) three features [Gutierrez-Osuna]

number of methods of reducing the dimensionality that are grouped into feature selection and feature extraction approaches. Feature selection chooses a subset of all the features which are more informative while feature extraction creates a new set of less dimensionality by creating combination of the existing features. Both these approaches select features or create features, which are combination of the original features that have high discrimination power and give better between class difference and better within class similarity. This is done by choosing a optimal criterion function J that defines this similarity and/or difference [Webb 1999]. Although it is not reliable when the ratio of sample size and features is small, commonly used criterion function is the classification error [Jain *et al.* 2000]. Another issue that needs to be considered is the number of reduced features.

Feature extraction

The feature extraction method determines a feature subspace of small dimensionality which is a linear or non-linear combination of the ordinal feature space [Jain *et al.* 2000; Webb 1999]. Feature extraction can be formally defined as follows:

> Given a set of features $X = \{x_1, x_2, x_3, \ldots, x_d\}$, find a subset X' derived from X with $|X'| = m$ such that:
>
> $$J(X') = \max_{x \in X_m} J(X)$$
>
> where $J(.)$ is the optimal criterion function and $m \leq d$

Therefore the aim of feature extraction is to find a concise representation of the data with no or minimal loss of information. Feature extraction has been an important problem for decades and hence different methods have been proposed. These methods can be grouped in to linear or non-linear transformation of the original feature space. Linear and non-linear transformation may be distinguished further by supervised or unsupervised. The best known unsupervised linear transformation technique is the Principal Component Analysis [Jain *et al.* 2000; Webb 1999]. In these methods, the transformed features are ranked according to the value assigned by criterion function and the first $m \leq d$ features will be chosen [Jain *et al.* 2000]. Another linear transformation method which uses the same principle is projection pursuit [Jain *et al.* 2000]. Other methods like discriminant analysis use the within-class information to perform linear extraction. With the development of neural networks, new methods of feature extraction have been investigated. One such method is the non-linear feature extraction method known as Self-Organising Map (SOM).

The mathematical formulation and detailed discussion of Principal Component Analysis will be presented next.

Principal component analysis

The Principal Component Analysis (PCA) method originated in 1901 to derive a new set of features which are linear combination of the original set of features describing the data sorted in descending order of importance according to the criterion function [Webb 1999]. If the new feature space is said to have a dimensionality $m \leq d$, the first m newly derived features will be considered. This technique is widely used because of three important properties [Roweis 1998]:

- It is an optimal linear method for dimensionality reduction
- Model parameters are computed from the data itself
- Once model parameters are computed, compressing and decompressing data are trivial

There are a number of ways describing PCA mathematically or geometrically. Geometrically it can be described as finding a set of axes rotated from the original axes to better fit the data and magnify between class difference [Webb 1999; Norris 2002]. Norris [2002] also described PCA as "decomposing the original pattern into a set of distinct patterns over the sample and then recombine them to recreate the original data". PCA is based on the statistical representation of the pattern and suppose the pattern is represented as $\mathbf{X} = (x_1, x_2, \ldots, x_d)^T$ and the mean be denoted by:

$$\mu = E[x]$$

and the $d \times d$ covariance matrix given by:

$$\Sigma = E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T]$$

The problem of finding the principal components is then defined as finding the d dimensional normalised vector a_i^T such that for $y_i = x_i^T a$, the following properties holds:

1. Var$[y_1] \geq$ Var$[y_2] \geq \ldots \geq$ Var$[y_d]$
2. For all $i \neq j$ the covariance between y_i and y_j should be zero. i.e. y_i should be uncorrelated with y_j

Consider the first term y_1, the first principal component

$$y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1d}x_d$$

defined by choosing $a_1 = (a_{11}, a_{12}, \ldots a_{1d})$ to maximise the variance of y_1, constrained to $|a_1|^2 = 1$. The variance of y_1 can be written in terms of the covariance matrix and a_1 as:

$$\text{Var}[y_1] = a_1^T \Sigma a_1$$

Hence to find the first principal component one must solve the problem:

$$\text{Maximise} \quad a_1^T \Sigma a_1 \quad \text{subject to} \quad a_1^T a_1 = 1$$

this is equivalent to maximising

$$f(a_1) = a_1^T \Sigma a_1 - \nu a_1^T a_1$$

where ν is a Lagrangian multiplier. Setting the partial derivatives with respect to a_1 to zero and solving for a_1 will give us the vector value of a_1 that maximises the variance of y_1

$$\frac{\partial f(a_1)}{\partial a_1} = \Sigma a_1 - \nu a_1 = 0 \tag{2.3}$$

Comparing this expression with eigenstructure of the square matrix Σ, (2.3) tells us that choosing a_1 to be the eigenvector of Σ with eigenvalue ν solves the maximisation problem. Furthermore (2.3) implies:

$$\text{Var}[y_1] = a_1^t \Sigma a_1 = \nu$$

Ordering the eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_d$ of the covariance matrix Σ in such a way that

$$\lambda_1 \geq \lambda_1 \geq \ldots \geq \lambda_d$$

Because the aim is to maximise the variance we choose ν to be the largest eigenvalue λ_1. The second component, the second principal component

$$y_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2d}x_d$$

can also be defined by choosing $a_2 = (a_{21}, a_{22}, \ldots a_{2d})$ to maximise the variance of y_2, constrained to $|a_2|^2 = 1$ and the covariance between y_1 and y_2 is zero. The second constraint implies:

$$E[y_2 y_1] - E[y_1]E[y_2] = 0$$
$$a_1^T \Sigma a_1 = 0$$

Note that since a_1 is an eigenvector of Σ, and hence a_2 is orthogonal to a_1 (i.e. $a_1^T a_1 = 0$). Using Lagrangian multipliers (μ and η), the problem of maximising the variance of y_2 can be reformulated as

$$f(a_1, a_2) = a_2^T \Sigma a_2 - \mu a_2^T a_2 - \eta a_2^T a_1$$

Setting the partial derivatives with respect to a_2 to zero and multiplying it by a_1^t, tells us that $\eta = 0$ hence we have:

$$\Sigma a_2 = \mu a_2$$

This tells us that a_2 is also eigenvector of Σ, which is orthogonal to a_1. Remember that we are still looking to maximise the variance of y_2, hence a_2 will be the largest of the remaining eigenvectors of Σ (i.e. λ_2).

Following the same argument, the variance of the i^{th} principal component is

$$\text{Var}[y_i] = a_i^t \Sigma a_i = \lambda_i \tag{2.4}$$

And the total variance is:

$$\sum_{i=1}^{d} \text{Var}[x_i] = \lambda_1 + \lambda_2 + \ldots + \lambda_d = \sum_{i=1}^{d} \text{Var}[y_i]$$

Thus to compute the portion of the total variance if the data that is captured by the i^{th} principal component we can use the ration:

$$\text{proportion of variance} = \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j}$$

and the portion of the total variance captures by first k principal components is

$$p = \sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{d} \lambda_i$$

The value of the p is determined by the size of the reduced dimension and is chosen by the user. Although the value of p is problem specific choosing a value between 70% and 90% will retain the information of the original data well [Jolliffe 1986].

The effect of dimensionality reduction using PCA on the performance of some of the pattern recognition techniques used in this research will be assessed.

Feature selection

Another approach to dimensionality reduction is feature selection. Unlike feature extraction that finds a feature space will lesser dimensionality which is linearly or non-linearly combination of the input space, feature selection finds a subset of the input space that has a considerable information and lesser dimensionality. To decide which feature space out of the possible subsets, we can use classification performance of each subspace and select the one with the least classification error.

There are a number of approaches to do feature selection. In this research feature selection techniques are not applied but interested reader can read the work by Saeys [2004].

2.6 Neural networks

2.6.1 Introduction

The traditional von Neumann machine, which abstracts the human information processing was able to solve computational problems faster than a human brain, however it was inefficient when it comes to recognition, classification and description tasks that the human brain handle with ease. Real life classification/recognition problems are made complicated by a number of external factors such as orientation of the object, direction of vision, deformation etc. Aimed at addressing these limitations, a different paradigm of computing called neural networks was proposed way back in 1940's [Russell 1991].

Neural networks are systems inspired by the biological nervous system, which is composed of numerous inter-connected simple elements (neurons) operating in parallel. These small elements work as a unit to perform a given task (such are prediction, classification) by adjusting the value of the connections between them (termed as weight). The values of these connections are usually adjusted dynamically based on the performance of the network on the given training and validation sets [Russell 1991].

A neural network is characterised by three design decisions [Wu 1997]:

- pattern of connection between neurons (architecture)
- activation function
- method of determining the weight on the connection (training or learning algorithm)

Both supervised and unsupervised learning methods can be used for neural networks. However, the presentation in this section is based on supervised learning.

2.6.2 The Perceptron

The simplest form of a neural network used to classify linearly separable patterns is the perceptron created by Rosenblatt in 1961 [Rosenblatt 1962]. A perceptron has two layers namely the

input layer and the output layer (see Figure 2.5). The weights and bias can be adjusted from the input/output pair presented using the perceptron learning rules. The perceptron learning rule is described below. Perceptrons have gained repetition because of their ability to generalise well from a given training sample and a randomly distributed connection (random initial weights) [Demuth and Beale 2004].

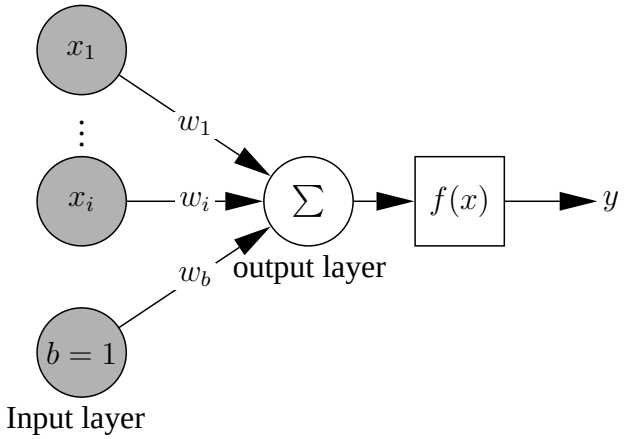

Figure 2.5: A perceptron. x_i and y are the input and output respectively. w_i is a weight associated with each node in the input layer and b is the bias.

The output of the network given in Figure 2.5 is:

$$y = f(\sum w_i.x_i + w_b) \tag{2.5}$$

The activation function $f(x)$ can have any form depending on the application of the network (for example: the sign function, log-Sigmoid function). The log-sigmoid function is a typical choice [Rumelhart *et al.* 1994].

The perceptron uses adaptive learning to adjust its weight in order to produce the correct output. The rule governing this, known as the perceptron learning rule, is as follows [Demuth and Beale 2004]: *If the network generated the correct output make no change, if a wrong output is generated, adjust the weights and the bias by an amount proportional to the difference between the correct output and the generated output.*

In a more generalised way, perceptron learning can be mathematically it can be summarised as:

$$w_{i+1} = w_i + \Delta w_i \tag{2.6}$$

$$\Delta w_i = \eta(T_i - y_i)x_i \tag{2.7}$$

where i corresponds to the current learning set, η is called the learning rate ($0 < \eta \leq 1$) , T_i refers to the target output and y_i is the actual output.

Equation 2.7 is known as the delta rule. Like any other error-correction learning rule, the ultimate aim of this rule is to increase the overall performance of the system or minimise the cost [Rumelhart *et al.* 1994]. The common cost function considered is the sum of squared error

(SSE) (sum of squared difference between the desired output and the actual output). SSE is defined as:

$$SSE = \frac{1}{2}\sum_i (T_i - y_i)^2 \tag{2.8}$$

With this definition of the cost, the learning goal will then be minimising this function (Equation 2.8) with respect to the weight (w_i).

$$\Delta w_i = \frac{\partial SSE}{\partial w_i} \tag{2.9}$$

$$\frac{\partial SSE}{\partial w_i} = \sum_i (T_i - y_i)\frac{\partial y_i}{\partial w_i}$$

The learning rule in Equation 2.7 will become:

$$\Delta w_i = \sum_i (T_i - y_i)\frac{\partial y_i}{\partial w_i} \tag{2.10}$$

Equation 2.10 is called the Gradient descent learning.

The perceptron learning rule is capable of solving any linearly separable classification problem. However, the gradient descent learning is capable of minimising the squared error to the hypothesis (some acceptable value defined by the experimenter) even when the problem is not linearly separable. Although these learning rules are capable of solving classification problems in finite time, some classification problems (for example: XOR function) are too complicated to be learned with acceptable accuracy. Most real-world problems are not linearly separable and can not be solved with the simplest architecture (single layer network). To address this problem a number of perceptrons can be combined to handle multiclass and complicated classification problem where there is more than one output and the problem needs a complicated hyperplane to classify the patterns.

2.6.3 Feedforward multilayer perceptrons

Multilayered perceptron has additional layer called a hidden layer and the neurons in these layers are called hidden neurons. (They are called hidden because the neurons in these layers have not external connection except, the input, output or another hidden neuron). The output from one layer serves as input to the next layer. Figure 2.6 shows multilayered perceptrons with one hidden layer.

The most popular form of multilayered perceptrons is feedforward topology. In this configuration neurons on a layer are only connected to neurons in the next layer. Connection between neurons in the same layer and loops are also not allowed. (for example: in Figure 2.6 the input neurons are connected to neurons from the hidden layer and neurons in the hidden layer are connected to neurons in the output layer).

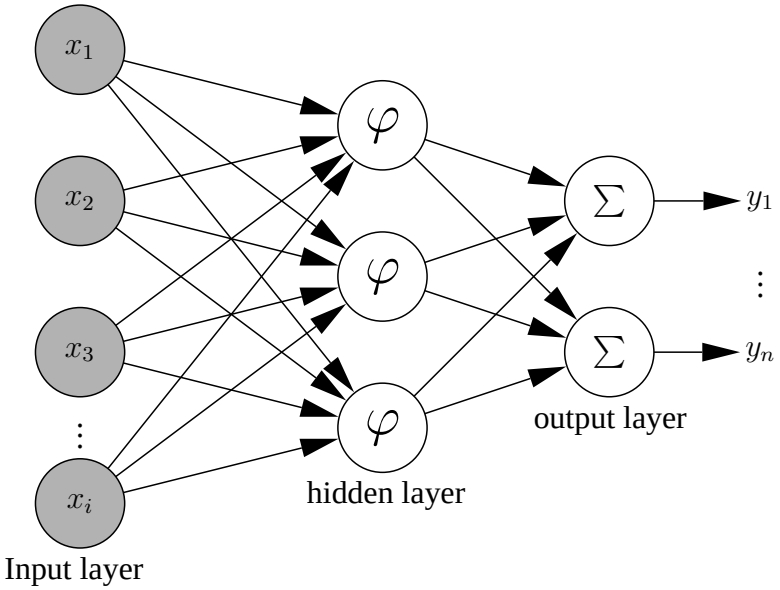

Figure 2.6: A multilayered perceptron with one hidden layer. Multilayered perceptron is called fully connected if each neuron in one layer are connected to the every neuron in the next layer. Otherwise it is called partially connected. Each connection between neurons have associated weight. The activation function (φ) on the hidden layers must be non-linear. Multilayered perceptron with linear activation in the hidden layers can be effectively represented by a perceptron with relevent activation function.

Similar to perceptrons the output is computed from the input and the weight of the respective connection. Each neuron in the network has an output which is the input for the neurons in the next layer:

$$y_j = f(\sum_i w_{ji} \times x_i + w_b) \tag{2.11}$$

Where w_{ji} refers to the weight of the i'th connection in the j'th layer. y_i is the output of the i'th neuron. x_i is the input for the i'th neuron. x_i's equals the network input for the neurons in the input layer and equals the output of the layer before for the subsequent layers. The final output(s) of the network is computed similarly.

The weights are adjusted during the training process to minimise the sum of squared error. For multilayered perceptrons SSE is defined as:

$$SSE = \frac{1}{2}\sum_i (y_{di} - y_{ai})^2 \tag{2.12}$$

where y_{di} refers to the desired output and y_{ai} is the actual output.

During the training the outputs of the network are calculated by calculating the output at each layer and continuing until final the output(s) is/are generated. The weights are then adjusted by an amount proportional to the SSE of the neuron feeding in to the weight at the given time during the training. Adjusting these weights is not direct as it is for perceptrons. The error for the hidden layer can only be known when the error at the output layer is known. Hence

there must be a mechanism to propagate the error through the network so that each subsequent weight is adjusted. There are a number of algorithms to handle this task. The most popular backpropagation algorithm is discussed below.

Backpropagation algorithm

Multilayer perceptron with backpropagation learning is the most popular network architecture [Delen and Kadam 2005]. During the training process of this architecture, the forward pass the activations propagate from the input layer to the output layer of the network. On the other hand the backward pass weights for the connection will be adjusted based on the difference between the desired and actual output of the layer after it (for example in Figure 2.6 the weights of the output layer are adjusted based in the error of the output of the network and the weight hidden layer will then be calculated based on these values). The backpropagation algorithm adjusts the weight of the connection between the neuron i and j at the $k+1$'s iteration as follows [Riedmiller and Braun 1993]:

$$w_{ji}(k+1) = w_{ji}(k) - \eta \frac{\partial SSE}{\partial w_{ji}}(k) \tag{2.13}$$

The learning rate η has an important effect on the convergence time of the learning. However, while a small value for the learning rate might leave the network to require a large number of iterations to converge, a big value will leave the network to jump between different values and might cause the network not to achieve the required bound on the error. To address this problem, one of the proposed solution is introducing a momentum term. The modified change in weight will be as shown in Equation 2.14. The introduction of these momentum term will give stability to the learning process (the parameter μ) scales the influence of the previous weight), however this is not always true in practice [Riedmiller 1994].

$$\Delta w_{ji}(k) = -\eta \frac{\partial SSE}{\partial w_{ji}}(k) + \mu \Delta w_{ji}(k-1) \tag{2.14}$$

The backpropagation algorithm has a number of limitation often making the training process too slow for real-world problems [Riedmiller and Braun 1993; Delen and Kadam 2005]. To address this problem there are a number of modifications to these algorithms. Some of these modifications took the heuristic approach while others perform the numerical optimisation to speed up the training process. Some of these algorithms are Resilient propagation algorithm proposed by Riedmiller and Braun [1993] which uses the heuristic approach and the Levenberg-Marquardt algorithm proposed by Hagan and Menhaj [1994] which uses numerical optimisation.

Resilient backpropagation algorithm

The Resilient backpropagation algorithm was proposed by Riedmiller and Braun [1993] to address the weight-update related limitations of standard backpropagation. This algorithm uses

the sign of the derivative to determine the increase or decrease in the value of the weight. The size of the weight change is determined with separate update value. The update value of the weights are as follows [Riedmiller and Braun 1993]:

$$\Delta w_{ji}(k) = \begin{cases} -U_{ji}(k) & \text{if } \frac{\partial SSE}{\partial w_{ji}}(k) > 0, \\ +U_{ji}(k) & \text{if } \frac{\partial SSE}{\partial w_{ji}}(k) < 0, \\ 0 & \text{else} \end{cases} \tag{2.15}$$

The update value is determined as follows:

$$U_{ji}(k) = \begin{cases} \epsilon^+ \times U_{ji}(k-1) & \text{if } \frac{\partial SSE}{\partial w_{ji}}(k-1) \times \frac{\partial SSE}{\partial w_{ji}}(k) > 0, \\ \epsilon^- \times U_{ji}(k-1) & \text{if } \frac{\partial SSE}{\partial w_{ji}}(k-1) \times \frac{\partial SSE}{\partial w_{ji}}(k) < 0, \\ U_{ji}(k-1) & \text{else} \end{cases} \tag{2.16}$$

$$\text{where } 0 < \epsilon^- < 1 < \epsilon^+$$

The update value is set to an initial value which is perferably proportional to the initial weight. 0.1 is usually a good initial value. The Resilient backpropagation algorithm is much faster than the standard backpropagation algorithm and has a modest memory requirement [Delen and Kadam 2005].

A thorough explanation of this algorithm can be found in Riedmiller and Braun [1993]; Riedmiller [1994].

Levenberg-Marquardt algorithm

Levenberg-Marquardt learning was introduced by Hagan and Menhaj [1994] to speed up the training process of a feedforward network. This algorithm generally gives a numerical solution to the problem of minimising a sum of squares (Equation 2.12) of a nonlinear function (nonlinear least squares) [Hagan and Menhaj 1994]. For the function $y = f(x, w)$, with x and y are the input and the output variables respectively, the Taylor series expansion gives us [Chan 1996]:

$$J\Delta w = e \tag{2.17}$$

where J is the Jacobian matrix that contains the first derivative of the network error with respect to weight and e is the error for a given input.

The Levenberg-Marquardt modification to the solution to the above equation (Equation 2.17) by Newton's method is given by [Hagan and Menhaj 1994; Chan 1996]:

$$\Delta w = (J^T J + \gamma I)^{-1} J e \tag{2.18}$$

where γ is a regularisation parameter introduced to prevent the ill-conditioned property of Hessian which is approximated by the square of the Jacobian $(H = J^T J)$. I is the unity matrix.

Based on the choice of value for γ, Equation (2.18) exhibits different property. When the Hessian γ equals zero, it will become the Newton's method with approximated Hessian matrix. With large value for γ the algorithm becomes gradient descent with step equals to $\frac{1}{\gamma}$. However a small value will change the algorithm to Gauss-Newton method [Hagan and Menhaj 1994; Delen and Kadam 2005]

A thorough explanation of this algorithm can be found in Marquardt [1963]; Hagan and Menhaj [1994].

2.7 Decision trees

2.7.1 Introduction

Decision trees are one of the popular classification techniques. This technique is an example of multistage decision process where a subset of the attributes (a single attribute for example) that make up the pattern rather than he whole pattern is used to examine the decision making process at different level of the decision tree [Webb 1999]. Decision trees are a tree-structured classifier (see Figure 2.7) where each node in the tree is either a leaf node, which specifies a class value or a decision node specifying a further test to be carried out on a single attribute value. The number of splits a decision node has depends on the outcome of splitting test carried on an attribute value on the node. For a discrete attribute P_1 with possible values $a_1, a_2, \ldots, a_n$, the possible outcomes of the split tests are $P_1 = a_1, P_1 = a_2, \ldots, P_1 = a_n$ (see Figure 2.7 for example). If the attribute P_1 has a continuous value, there are two possible outcomes, $P_1 \leq t$ and $P_1 > t$.

A decision tree can be used to classify a pattern by traversing the tree from root till we reach a leaf node which specify the class to which pattern belongs to. Travelling from the root to a leaf node is carried based on the outcome of the split test on each decision node in between. Application of decision tree to classify a pattern is similar to applying a series of *if-else-then* statements on the pattern.

This classification technique has a number of features that makes it popular such as speed of classification, the ability to interpret individual features separately and the ability to handle missing data [Jain *et al.* 2000]. Furthermore, the decision tree can easily be interpreted to decision rules which can be easily understood by the domain experts.

2.7.2 Constructing of decision tree

The main issue of decision tree construction is constructing the smallest possible decision tree based on a given learning set. This step is carried out in a recursive manner and is an NP-complete problem [Quinlan 1993]. The general method of constructing decision tree is summarised in Quinlan [1993, page 17-18] as follows:

> If there are k classes denoted $\{C_1, C_2, ..., C_k\}$, and a training set, T, then

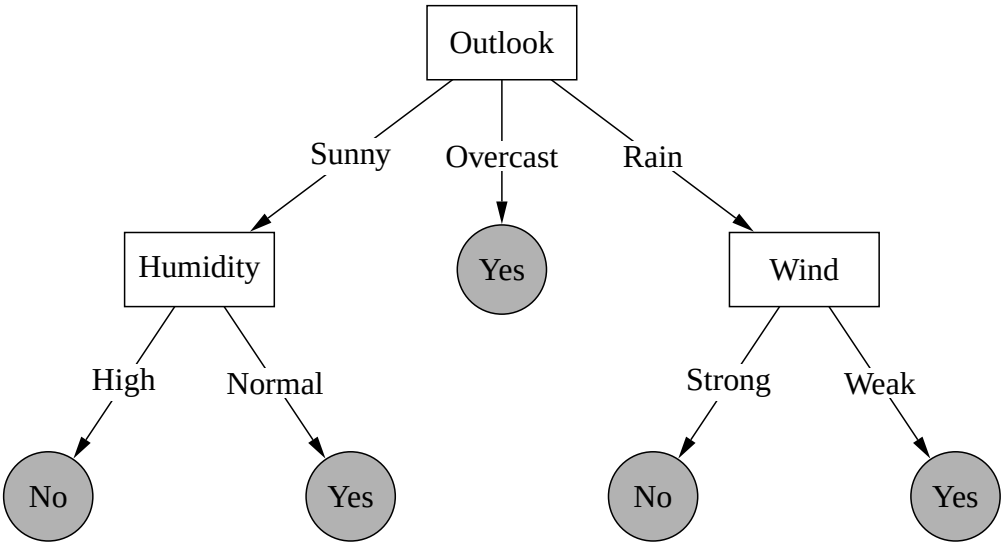

Figure 2.7: Decision tree for conditions to play tennis

- if T contains one or more patterns which all belong to a single class C_j, then the decision tree is a leaf identifying class C_j.
- if T contains no pattern, the decision tree is a leaf determined from information other than T.
- if T contains objects that belong to a mixture of classes, then a test is chosen, based on a single attribute, that has one or more mutually exclusive outcomes $\{O_1, O_2, ..., O_n\}$. T is partitioned into subsets $T_1, T_2, ..., T_n$, where T_i contains all the objects in T that have outcome O_i of the chosen test. The same method is applied recursively to each subset of training objects.

According to this outline, when constructing a decision tree there are certain questions one need to answer. Some of the questions are:

- how to choose the best split.
- when to stop growing the tree
- how to prune the tree
- how to handle missing attributes

Different variations of decision tree such as C4.5 [Quinlan 1993], CART (classification and regression tree) [Breiman *et al.* 1984], CHAID (chi-square automatic interaction detection) [Kass 1980], QUEST (quick unbiased efficient statistical tree) [Loh and Shih 1997] etc. address these questions differently. In this dissertation we will how these question are answered in the context of the induction of decision tree algorithm: C4.5.

2.7.3 Splitting test

Different decision tree algorithm use different criteria to split the training sample in to subsets on each split test. The split evaluation criteria used by C4.5 induction algorithm is information gain which measures the quality of given attribute as a split criteria based on the targeted classification. The presentation in section follows Quinlan [1993] unless specified.

For any subset S of set T of training examples, let $\mathbf{freq}(C_i, S)$ be the number of patterns belonging to class $C_i (i = 1, 2, \ldots, n)$. Selecting one example at random from S and declaring that it belongs to class C_i has a probability of:

$$\frac{\mathbf{freq}(C_i, S)}{|S|} \tag{2.19}$$

where $|S|$ is the number of examples in the subset S. And the information conveyed (in bits) by this 'message' is

$$-\log_2 \frac{\mathbf{freq}(C_i, S)}{|S|} \tag{2.20}$$

In general for the probability distribution $P = (= p_1, p_2, \ldots, p_n)$, where $p_i = \frac{\mathbf{freq}(C_i, S)}{|S|}$, the information conveyed by this distribution is:

$$\mathbf{Info}(P) = \sum_i^n p_i \times \log(p_i) \tag{2.21}$$

similarly the information conveyed by the set S will be given by:

$$\mathbf{Info}(P) = -\sum_i^n \frac{\mathbf{freq}(C_i, S)}{|S|} \times \log_2 \frac{\mathbf{freq}(C_i, S)}{|S|} \tag{2.22}$$

This amount is called entropy of the set S. (see Shannon [2001] for more on Entropy and information theory).

For example, if we have 15 patterns in S and 9 belong to one class and the rest belong to another class (two class problem), the average amount information needed to identify the class of a pattern in S equals:

$$\mathbf{Info}(S) = \mathbf{Info}(\frac{9}{14}, \frac{5}{14}) = 0.94$$

Now consider attribute X of a pattern in T. If X is a discrete value and have k possible values, and $T_1, T_2, \ldots, T_k$ are the subsets of T consisting of patterns with distinct values for this attribute (X), then the expected information requirement of T can be found as a weights sum over the T_i's:

$$\mathbf{Info_X}(T) = \sum_i^k \frac{|T_i|}{|T|} \times \mathbf{Info}(T_i) \tag{2.23}$$

The information gained by X induced partitioning can be calculated by evaluation the informa-

tion before and after partitioning as:

$$\mathbf{Gain}(X) = \mathbf{Info}(T) - \mathbf{Info_X}(T) \tag{2.24}$$

This value favours testing on attributes with large number of distinct value. For example, for attribute X with unique value for each patterns in T, $\mathbf{Info_X}(T)$ will be zero, maximising the $\mathbf{Gain}$. To avoid this bias Quinlan [1993] uses information gain ratio instead. By analogy to Equation (2.21), the information generated by dividing T into k subsets induced by X is given by:

$$\mathbf{Split}(X) = \sum_{i}^{k} \frac{|T_i|}{|T|} \log_2 \frac{|T_i|}{|T|} \tag{2.25}$$

This is *split information*. Information gain ratio is then defied as:

$$\mathbf{Gain\_Ratio}(X) = \frac{\mathbf{Gain}(X)}{\mathbf{Split}(X)} \tag{2.26}$$

The gain and gain ratio for an attribute which is already selected as a split criteria in an ancestor node will be zero. Therefore this attribute will not be selected again as we go down the tree.

In many real world problems attributes do not always have discrete values. Hence we need to address cases where attributes have continuous value. Quinlan [1993] and a later suggested improvement in Quinlan [1996] handles continuous attributes as follows.

Suppose the attribute X has a continuous range. Although X has a continuous range, there can only be finite number of these values in set T. Computing information gain of this attribute starts by sorting these values in ascending order. Say the ordered values for attribute X are $v_1, v_2, \ldots, v_m$. Then for each mid-point of intervals $v = \frac{v_i + v_{i+1}}{2}$ for $i \in [1, m-1]$ we partition T into two sets: the first subset contains patterns with attribute $X \leq v$ and the second subset containing attributes with $X > v$. For each of the m partitions the gain ratio (Equation 2.26) will be computed and the partition that maximise the gain will be used as a split criteria. Note that if all the attributes of the pattern have continuous range, we will have binary tree.

2.7.4 Stopping criteria

Based on the above specified criteria, the construction of the decision tree will be carried out and a further splitting will be stopped and a leaf node is declared based on a number of criteria. Allowing the tree to grow until every pattern in the training set is correctly classified will result in an over-fitted model, which will not have good generalisation ability. Alternatively, if we stop the construction process early will result in under-fitted model which will equally have poor generalisation ability. One suggested solution to address this problem is employing stopping criteria taking both over-fitting and under-fitting in to consideration. Some of these include:

- If all possible split test have zero gain. In this case all the patterns in the subset belong to

the same class.

- When the gain ratio is below some predetermined value.
- Number of patterns in the subset are fewer than some predetermined value or some proportion of the patterns belong to the same class.
- The depth of the node exceeds some predetermined value

Employing stopping criteria enables us to avoid over-fitting; however, the results are uneven due to the noise and incorrectly classified patterns in the training set. To avoid these anomalies, C4.5 uses pruning method which is based on estimating the error rates.

2.7.5 Pruning

The generalisation ability of a fully grown tree can be increased by pruning the subtrees that are not contributing positively towards the generalisation ability if the resulting model. Pruning of a decision tree refers to the process of replacing a subtree with a leaf node or with the most frequently used branch [Quinlan 1993]. Quinlan [1987] suggested three different techniques of pruning. These are:

- Cost-complexity pruning
- Reduced error pruning and
- pessimistic pruning

Cost-complexity pruning proposed by Breiman *et al.* [1984] performing pruning as follows: a large tree is created, and then a sub-tree is found starting from the leaves. Again from this sub-tree another sub-tree is found until we are left with a sub-tree containing the root node only. All this sub-trees will then be tested on a independent validation set and the sub-tree with the least cost will be selected.

Reduced error pruning also uses sequence of sub-trees to choose one with the least misclassification on an independent validation set. However, the way the sub-trees are generated is different from cost-complexity pruning. In this approach, for each non-leaf sub-tree, we replace it with the best possible leaf that will enable the resulting tree better the original on the validation set. This process in carried out on the resulting tree until no more gain in classification is found. Both cost-complexity pruning and reduced error pruning require a independent validation set and this might be a disadvantage in some classification problem where the number of training patterns are limited. There are a number of approaches in place to address such scenario. One of these approaches is cross-validation. However, the approach taken by C4.5 is a pruning method that does not require an independent validation set: pessimistic pruning.

Pessimistic pruning

Pessimistic pruning is an approach which increases the errors observed at each leaf pessimistically using continuous correction for binomial distribution to encourage pruning [Quinlan 1987]. Quinlan [1987] describes pessimistic pruning that effectively increases the number of observed error at each leaf by 0.5. However, C4.5 uses a far more pessimistic estimate as presented in [Kohavi and Quinlan 2002] as follows.

When a non-leaf covering N training patterns of which E are misclassified, $\frac{E}{N}$ is an estimate of the probability of p of misclassification. However, since the decision tree is constructed on the same N training patterns it tends to minimise the apparent error. To make our estimate more realistic we can derive a confidence limit for p. For a given confidence term CF, we can find an upper limit p_r such that $p \leq p_r$ with probability $1 - CF$. Following [Diem 1962, as cited in Kohavi and Quinlan [2002]], p_r satisfies:

$$CF = \begin{cases} (1-p_r)^N & \text{if } E = 0 \\ \sum_{i=0}^{E} \binom{N}{i} p_r^i (1-p_r)^{N-i} & \text{if } E > 0 \end{cases} \tag{2.27}$$

Let $U_{CF}(E, N)$ be the upper bound on error p_r.

For the non-leaf tree T (shown in Figure 2.8), produced from N training patterns, where the sub-trees T_i^* are already pruned: Let T_f^* be the subtree corresponding to the most frequent outcome of the split test B and let L be the leaf labelled with the most frequent class in the training pattern.

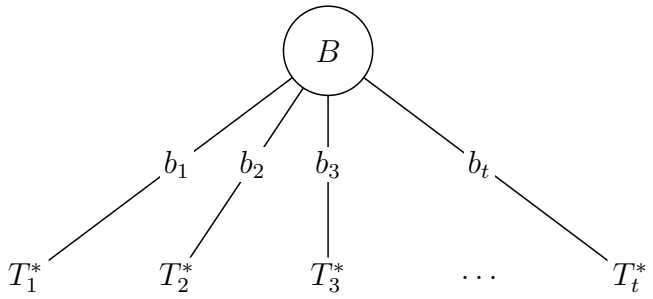

Figure 2.8: Non-leaf tree with already pruned subtrees T_i^*

If E_T, $E_{T_f^*}$ and E_L are the number of missclassified patterns by the tree T, sub-tree T_f^* and leaf L respectively, the corresponding estimated error rates are:

- $U_{CF}(E_T, N)$
- $U_{CF}(E_{T_f^*,N})$
- $U_{CF}(E_L, N)$

Depending on the lowest value of the above estimate, T will be left unchanged, is replaced by the subtree T_f^* or is replaced by the leaf L.

2.8 Logistic regression

Logistic regression is a regression method which is popular for modelling dichotomous data. In this model, the dependent variable Y has the value 1 with the probability $P(X_i)$ and the value 0 with probability $1 - P(X_i)$. The independent variables X_i can have any form. This model does not assume that the relationship between the independent variables and the dependent variable is a linear one, or that the dependent variable or the error terms are distributed normally.

Logistic regression is used to predict the effect of the independent variables on the binary response. The relationship between predictor and the response is determined by the *log odd* or **logit** transformation of $P(X_i)$. Given the values of $X_i = (X_{i1}, X_{i2}, \ldots, X_{id})$, the probability of the response being 1 (odds of observing 1 versus 0), is modelled using logistic regression model as:

$$\eta_i = \mathbf{logit}(P(X_i)) = log(\frac{P(X_i)}{1 - P(X_i)}) = \beta_0 + \sum_{j=1}^{d} \beta_j X_{ij} \tag{2.28}$$

where η_i is the linear predictor which is a combination of the independent variables. β_0 is the intercept and $\beta = (\beta_1, \beta_2, \ldots \beta_d)$ is the vector slope parameter [Cessie and van Houwelingen 1992; So 1995].

An alternative form of the logistic regression model is [Perlich *et al.* 2004]:

$$P(X_i) = \frac{\exp(\beta_0 + \sum_{j=1}^{d} \beta_j X_{ij})}{1 + \exp(\beta_0 + \sum_{j=1}^{d} \beta_j X_{ij})} \tag{2.29}$$

To find the best estimate for $P(X_i)$ the loss function used in this model is the log-likelihood. The parameter estimate $\hat{\beta}$ maximises the log-likelihood and gives estimate $P(X_i) = P(Y = 1)$. The log-likelihood for the data (X, Y) under the logistic model is given by:

$$l(\beta) = \sum_i [Y_i \log P(X_i) + (1 - Y_i) \log(1 - P(X_i))] \tag{2.30}$$

Maximisation of $l(\beta)$ yields the maximum likelihood estimator $\hat{\beta}$ for β. However, the model becomes unstable when the dimensionality of the pattern is big compared to the available sample [Perlich *et al.* 2004]. To obtain a more realistic estimates for the parameters and improve the predictive power of model, Cessie and van Houwelingen [1992] extended a method that adjusts the regression estimate by shrinking the correlation matrix for the predictor value to wards a fixed point by adding a constant λ to the diagonal element of the matrix.

The penalised log-likelihood is given by:

$$l^*(\beta) = l(\beta) - \lambda \sum_j \beta_j^2/2 \tag{2.31}$$

where $l(\beta)$ is the unrestricted log-likelihood function. The second term is the ridge penalty and λ is the ridge parameter. The ridge parameter regulates the penalty: when $\lambda = 0$ the solution

will be unrestricted maximum likelihood estimator, whereas $\lambda \rightarrow \infty$ the β_j all tend to zero [Cessie and van Houwelingen 1992; Eilers *et al.* 2001].

Choosing an optimal value for the smoothing or regularisation constant λ is crucial. The two dominant data-driven ways of selecting the ridge parameter are cross-validation and Akaik Information Criterion (AIC) [Schimek 2003]. Due to lack of empirical evidence in the later, cross -validation is the most popular and successful method[Schimek 2003]. The performance of cross-validation one may use either the fraction of misclassification or the strength of log-likelihood prediction.

A thorough explanation of penalised logistic regression with ridge estimator can be found in Cessie and van Houwelingen [1992]; Eilers *et al.* [2001]; Zhu and Hastie [2004].

2.9 Summary

In this chapter a detailed overview of the biological background of HIV, a brief overview of general pattern recognition principles and dimensionality problem and techniques of addressing it was presented. This chapter have also presented a detailed background on statistical pattern recognition techniques and Bayesian decision theory which is the tool broadly used to perform pattern classification based on some probabilistic knowledge about the random patterns and the pattern classes. A brief introduction to feedforward neural networks with backpropagation learning was given. Two algorithms proposed to address the problem with the standard backpropagation algorithm was also introduced. A brief introduction to decision trees based on the C4.5 algorithm was also given. Furthermore a brief introduction and direction for further reading for logistic regression was also give. This chapter have also covered a comprehensive discussion and the mathematical formulation of principal component analysis which is the best known unsupervised linear transformation technique.

Chapter 3

Support Vector Machine (SVM)

3.1 Introduction

SVMs are statistical learning technique developed by Boser *et al.* [1992] to perform a variety of supervised learning and function estimations task. Like any other machine learning technique, SVMs are aimed at minimising the risk (training or test error) based on some simple idea that provides a clear intuition of what supervised learning is all about. Most traditional machine learning techniques are based on the Empirical Risk Minimisation (ERM) principle, which approximates the estimation function based on minimising empirical risk (the error on the training data). However, SVMs use the principle of Structural Risk Minimisation (SRM), proposed by Vapnik [1979], which minimises the upper bound on the test error or risk [Schölkopf 1997]. The principle of SRM gives SVMs the advantage over other machine learning techniques, which are based on ERM. Machine learning techniques based on ERM are more likely suffer from over-fitting which is less likely to occur in those based on SRM. Over-fitting refers to the fact that the training data is perfectly learnt but the estimation function does not perform well on unseen data. SVMs do not suffer from over-fitting as much and hence give better generalisation.

Beside SRM, the ability of SVMs as a pattern recognition tool relies on their ability to transform the data into a higher dimensional feature space that is nonlinearly related to the input space. Transforming the data from the input space to feature space transforms a complex real-world classification into a simple classification problem where the classification task can be accomplished using a linear hyperplane [Ganapathiraju 2002]. Although the patterns are mapped from the input space into higher dimensional feature space, SVMs have a further advantage of performing all computations in the input space, using a specialised function called kernel functions. This trick further saves SVMs from suffering the curse of dimensionality.

As the result of the principle of SRM together with the kernel trick, SVMs are rapidly replacing neural networks, Radial Basis Functions (RBF) and polynomial classifiers which have been dominant pattern recognition tools [Hearst *et al.* 1998]. Furthermore, SVMs have already been applied in many benchmark applications including face detection [Osuna *et al.* 1997], text classification [Joachims 1998; Dumais *et al.* 1998; Sun *et al.* 2002], speech recognition [Ganapathiraju 2002] and in a number of other pattern recognition problems.

In this chapter a brief introduction to SVMs, implementation issues and some of the benchmark application of SVMs will be discussed. Section 3.2 will give a concise description on risk minimisation namely ERM and SRM. In section 3.3 we will give a deep insight into linear SVMs. In section 3.4, the non-linear Support vector machine is described in detail following a discussion on feature space and kernel mathematics. The material presents solution to two-class classification problem. Section 3.5 explains how the two-class classification will be extended to multi-class classification. Section 3.6 gives a brief overview of the approaches taken in implementation of SVM tools. The material presented in this section is based on Vapnik [1995]; Burges [1998]; Osuna *et al.* [1997]; Cristianini and Shawe-Taylor [2000]; Schölkopf [1997]. If any other reference is used it will be cited accordingly.

3.2 Risk minimisation

Given a set of training samples $(x_1, \omega_1), (x_2, \omega_2), \ldots, (x_l, \omega_l)$ generated independently and identically distributed from unknown probability with:

$$(x_1, \omega_1), (x_2, \omega_2), \ldots, (x_l, \omega_l) \in \mathbb{R}^d \times \Omega \tag{3.1}$$

where x_i is the input data, ω_i is the class x_i belongs to and l is the number of examples. The two-class pattern recognition problem can be described as choosing a function f_α given a set of decision functions:

$$\{f_\alpha : \alpha \in \Lambda\}, \qquad \text{where} \qquad f_\alpha : \mathbb{R}^d \to \{\pm 1\} \tag{3.2}$$

where $\{f_\alpha : \alpha \in \Lambda\}$ is called the hypothesis space[1] [Osuna *et al.* 1997] and will be denoted by $\mathcal{S}$. Λ is a set of abstract parameters introduced to enable f_α correctly classify unseen example (x, ω), which is generated with the same underlying probability distribution $P(x, \omega)$ as the training data set. However, for a given set of training samples there might be a number of such functions f_α which work on different subset of the training sample. So how do we know which one to choose?

Suppose we have two disjoint subsets of the training sample $X = \{x_1, x_2, \ldots, x_l\}$ and $\bar{X} = \{\bar{x_1}, \bar{x_2}, \ldots, \bar{x_{\bar{l}}}\}$ such that $X \cap \bar{X} = \emptyset$ and there exists a function f_α^* such that:

$$\begin{cases} f_\alpha^*(x_i) = \omega_i & i = 1, 2, \ldots, l, \\ f_\alpha^*(\bar{x_i}) \neq \bar{\omega_i} & i = 1, 2, \ldots, \bar{l} \end{cases}$$

If we use X as a training set and $\bar{X}$ the validation set, the decision function f_α^* will have a very poor performance on the validation set. Therefore, the aim of the pattern recognition system should be to find the function f_α which possibly performs better on both the training and validation set. In order to choose the function f_α^* that performs better on both the training and

[1]The hypothesis space is a set of all possible functions that can perform the classification and also where we choose the one that performs best

validation set, a way of quantifying the performance is needed. By defining of a loss function $L(\omega_i, f_\alpha(x_i))$ that measures the loss, the difference between the classification according to our pattern recognition system and the true class of the pattern x_i, the expected risk, which should be minimised over all classes of functions f_α will then be given as:

$$R(\alpha) = \int L(\omega_i, f_\alpha(x_i)) dP(x, \omega) \qquad \alpha \in \Lambda \tag{3.3}$$

Estimating the expected risk directly from the above equation is impossible because the distribution probability $P(x, \omega)$ is unknown. The only information available is the data, which is independently generated and identically distributed and can be considered a fair representation of the underlying distribution. Therefore, one can approximate this error by the measured mean error on the training set [Vapnik 1995].

3.2.1 Empirical Risk Minimisation (ERM)

The measured mean error (empirical risk) on the training set is given by:

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} L(\omega_i, f_\alpha(x_i)) \tag{3.4}$$

Note that there is no probability associated in equation 3.4 and $R_{emp}(\alpha)$ is a real number for a given function f_α. Once the estimation is done based on the available training sample, the function f_α that minimise the empirical risk will be considered. This is the principle of Empirical Risk Minimisation (ERM) [Vapnik 1995].

Consider the 0/1 loss function for two class pattern recognition problem defined as:

$$L(\omega_i, f_\alpha(x_i)) = \begin{cases} 0 & \omega_i = f_\alpha(x_i), \\ 1 & \omega_i \neq f_\alpha(x_i) \end{cases} \qquad i = 1, 2, \ldots, l$$

where ω_i takes the value $\{\pm 1\}$.

Using this loss function the empirical risk will have the following form.

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^{l} |f_\alpha(x_i) - \omega_i| \tag{3.5}$$

ERM is intuitive and easy to implement, but the question of consistency needs to be answered [Vapnik 1995; Osuna *et al.* 1997]. This means, does a non-trivial function f_α that minimises the empirical risk (3.4) also minimise the actual risk (3.3)?

Consistency of empirical risk depends on both the number of samples available (the law of large numbers) and the capacity of the set of functions (the Vapnik Chervonenkis dimension (VC dimension)) of the learning machine [Osuna *et al.* 1997; Vapnik 1995; Schölkopf and Smola 2001]. Furthermore, Vapnik and Chervonenkis have also shown that the finiteness of the

VC dimension of the hypothesis space is the necessary and sufficient condition for consistency of the ERM [Vapnik and Chervonenkis 1991, as cited in Osuna *et al.* [1997]].

When the number of available training samples is small, the empirical risk can easily be minimised to a small value close to zero, but the data hardly reflects the underlying probability distribution $P(x, \omega)$ and hence results in larger actual risk or inconsistent ERM. This phenomenon is termed over-fitting [Vapnik 1995]. Although it is not guaranteed, as the number of data points increases, the empirical risk will increase and at the same time the data will more likely represent the underlying probability distribution and the actual risk will converge to the empirical risk. In this case the ERM is said to be consistent and hence the pattern recognition system that performed well on the training set will also perform well on unseen problems generated with the same underlying probability distribution $P(x, \omega)$. Figure 3.1 shows a simplified description of the consistency or inconsistency of ERM.

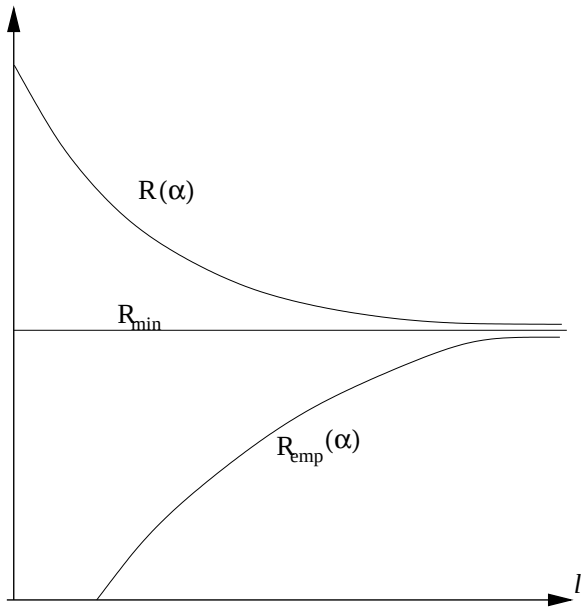

Figure 3.1: Asymptotic behaviour of a minimum empirical and corresponding expected risk for consistent ERM. The $x-$axis shows the the number of examples and the $y-$axis shows the corresponding risk

In most real life pattern recognition problems we are limited to a finite (usually small) training data and hence we need to make the most out of these data. To handle this situation a number of techniques have been proposed in statistical pattern recognition. One such approach is leaving room for a difference between the empirical risk and the actual risk. Following this, Vapnik [1995] has shown there exists a bound for the actual risk given the empirical risk and the measure for the capacity of the class of functions. This bound is given as:

$$R(\alpha) \leq R_{emp}(\alpha) + \phi(h) \tag{3.6}$$

The quantity h, which is the measure of the capacity of a set of functions, is called the VC dimension of a set functions and the function $\phi(h)$ is called the confidence term for the ERM

[Burges 1998; Schölkopf 1997]. The VC dimension of a set of function is defined as the maximum number of points that can be shattered by this class of functions. A given training sample of d dimension is said to be shattered by the function f_α, if the function can correctly classify the training sample into all the possible classes. For example, consider the two-class problem defined above. In this problem, a set of l points can have 2^l different possible classifications and if the set of functions f_α can accomplish this, we say the set of points is shattered by the set of functions f_α [Burges 1998] (see Figure 3.2). A comprehensive discussion on convergence, consistency and VC dimensions and other capacity measures can be found in Vapnik [1995].

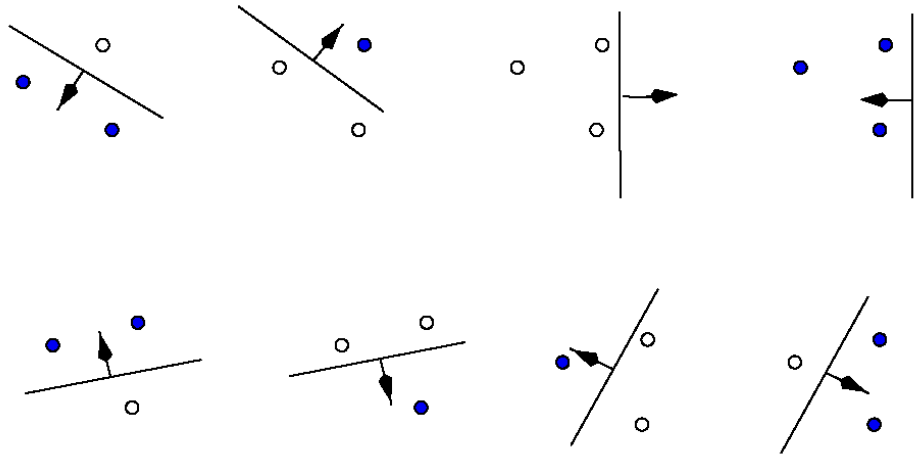

Figure 3.2: Three points in $\mathbb{R}^2$, shattered by oriented lines. The VC dimension of the set of oriented lines in $\mathbb{R}^2$ is therefore, three [Burges 1998, page 4].

From the inequality (3.6) we can see that for the consistency of the risk minimisation and good generalisation of the pattern recognition system, the confidence term, $\phi(h)$ should be minimised at the same time as the empirical risk. Suppose we only minimised the empirical risk to zero via the principle of ERM, the actual risk will be equal to $\phi(h)$ and a large value of this quantity will result in a system that has poor generalisation. And on the other hand if $\phi(h)$ is kept close to zero, the expected risk will be equal to the empirical risk, which needs to be minimised using the principle of ERM.

Consider again the two-class problem defined above. For any $\alpha \in \Lambda$ and $l > h$ with some value η such that $0 \leq \eta \leq 1$, Vapnik and Chervonenkis [1991] proposed an upper bound on the error and based on this proposal the inequality (3.6) can be redefined as follows and will hold with a probability of at least $1 - \eta$

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(\frac{2l}{h}) + 1) - \log(\frac{\eta}{4})}{l}} \tag{3.7}$$

The second term in equation 3.7 is proportional to the ratio $(\frac{h}{l})$. Therefore to achieve good generalisation the empirical risk and the ratio between the VC-dimension and the number of training samples should be kept to a minimum at the same time. The relation between the two

terms on the right hand side of equation 3.7 is graphically shown in Figure 3.3.

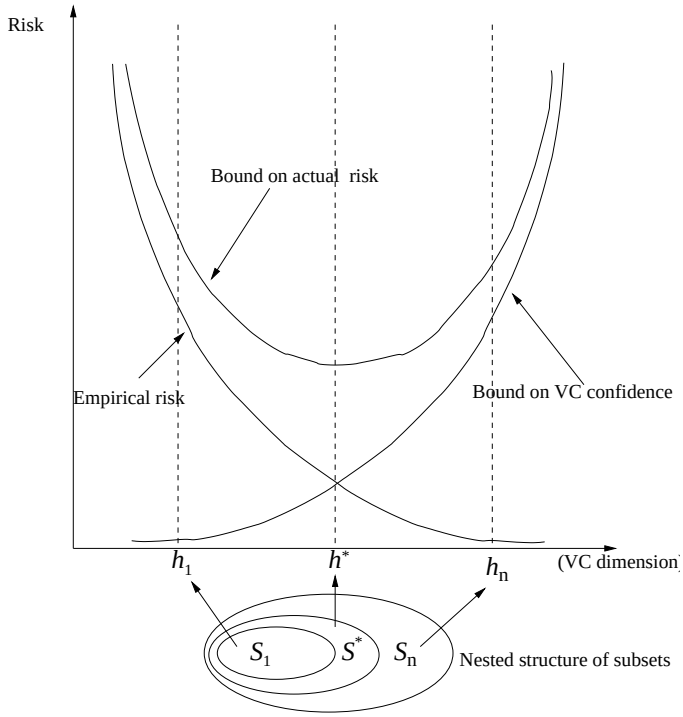

Figure 3.3: The optimal classifier needs to find some appropriate structure that minimises both the empirical risk and the confidence term [Vapnik 1995, page 98]

Burges [1998] pointed out that the empirical risk is a decreasing function of h and to get a good generalisation based on a limited training set one needs to find the optimal VC dimension. To overcome the problem of choosing the appropriate (finite) VC dimension Vapnik [1979] proposed the principle of structural risk minimisation. A detailed discussion on generalisation theory and the derivation of equation 3.7 can be found in Cristianini and Shawe-Taylor [2000].

3.2.2 Structural Risk Minimisation (SRM)

"The principle of structural risk minimisation defines a trade-off between the quality of the approximation of the given data and the complexity of the approximating function." [Vapnik 1995, page 95]. To define the trade-off, both terms in the right hand side of equation (3.7) should be minimised. If we select one hypothesis class $\mathcal{S}$, we are left with the task of minimising the empirical risk since the confidence term will be fixed for the selected hypothesis class. (By defining the hypothesis space, the VC dimension of the set of functions h is known and together with the size on the training patterns l, we can calculate the second term in equation 3.7). Let's define a nested structure of hypothesis space

$$\mathcal{S}_1 \subset \mathcal{S}_2 \subset \ldots \subset \mathcal{S}_n \subset \ldots,$$

with $h_i \leq h_{i+1}$. Note that equation 3.7 also holds for the nested sequence of hypothesis classes. By computing the respective actual risk for each subset, we choose the function f_α^* in the subset

$\mathcal{S}_i^*$ that minimises the upper bound on the risk [Schölkopf 1997; Osuna *et al.* 1997]. This principle is called principle of structural risk minimisation [Vapnik 1995]. Choosing this optimal combination (optimal trade-off between the capacity of the function and the empirical risk) is the task of the learning algorithm. To find the optimal hypothesis space one can use either of the following approaches: fix the confidence term and minimise the empirical risk or for a fixed empirical risk minimise the confidence term. However, the support vector algorithm accomplish this simultaneously [Osuna *et al.* 1997].

3.3 Linear support vector machine

The linear support vector machine also known as maximum margin classifier is the simplest form of support vector machine. In this section linear support vector machines will be presented first and will be used as an introduction to basic principles, notation and approaches that are later extended to more general Support Vector Machine. Furthermore, support vector machines are inherently binary classifiers. Therefore the formulation shown below and in subsequent sections is given for two class pattern recognition problem. In later sections we will see how this will be extended to multi-class pattern recognition problems. To make the presentation in two class classification problem simple and consistent with the conventional mathematical presentation we will use y_i instead of ω_i as class label.

The complete mathematical formulation can be found in Appendix B. Some mathematical steps are removed from the presentation in this section. For the missing mathematical steps refer to the Appendix.

3.3.1 Linear separable case (Maximum margin classifier)

Given set of examples $((x_1, y_1), \ldots, (x_l, y_l)) \in \mathbb{R}^d \times \{\pm 1\}$, where $y_i \in \{\pm 1\}$, and $x_i \in \mathbb{R}^d$, assume there exists a set of hyperplanes which totally discriminate the positive examples from the negative ones. This means we can find a pair (w, b) such that:

$$y_i(x_i.w + b) - 1 \geq 0 \qquad i = 1, 2, \ldots, l \tag{3.8}$$

where w is the direction of the normal or orientation of hyperplane and b is the threshold.

The mapping function which is usually called the hypothesis is then given by:

$$f(w, b) = sign(w.x_i + b) \tag{3.9}$$

Consider the example given in Figure 3.4. These training samples of the two classes can be perfectly separated by a linear hyperplane. Furthermore, one can find an infinite number of hyperplane that can accomplish this task. Some of these hyperplanes are shown in Figure 3.4. As we can seen from the figure, each of these hyperplanes has zero empirical risk, but we wish to find the one that will minimise the right hand side of equation (3.7) as seen in the

previous section. From the figure one can take an educated guess to say the hyperplane that passes through the middle will be more likely to give the minimum risk. Formally defining this hyperplane, the optimal hyperplane is a hyperplane which is likely to minimise the expected risk is the one that maximises the margin, which is defined as the distance between the examples of the two classes that are close to this hyperplane.

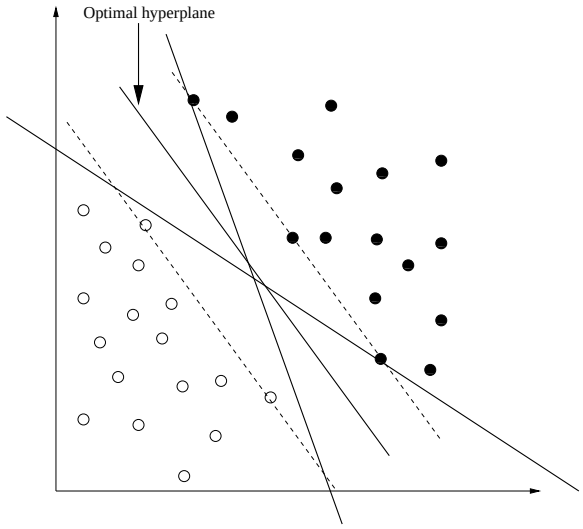

Figure 3.4: Optimal separating hyperplane

Once the optimal hyperplane is found all the training sets will satisfy equations (3.8) and classification will be based on the sign of equation (3.9). The points that lie on the hyperplane separating the data satisfies the equality:

$$w.x_i + b = 0 \tag{3.10}$$

Figure 3.5 gives graphical interpretation of support vector classification. All the points that satisfy the inequality $w.x_i + b \leq \pm 1$ lie on H_1 or to the left of it and those satisfying $w.x_i + b \geq \pm 1$ will lie on H_2 or to the right of it. The margin is therefore defined as the distance between H_1 or H_2 and the optimal hyperplane (see figure 3.5). It is evident that H_1 and H_2 are parallel and for a perfectly separable training set, no point lies between the two hyperplanes. Furthermore, H_1, H_2 and the optimal hyperplane differ only on the threshold b. Formally defining the optimal hyperplane with respect to these two hyperplanes, the separating hyperplane is optimal if the minimum distance between these hyperplane and the optimal hyperplane is maximal.

By computing the margin for a given orientation w, the problem of finding the optimal hyperplane will be defined as:

$$\text{Minimise} \quad \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad y_i(x_i.w + b) - 1 \geq 0 \quad \forall_i$$

Minimising a quadratic function under a linear constraint formulated above is called quadratic programming and can be solved to give the solution to the optimal hyperplane using Quadratic

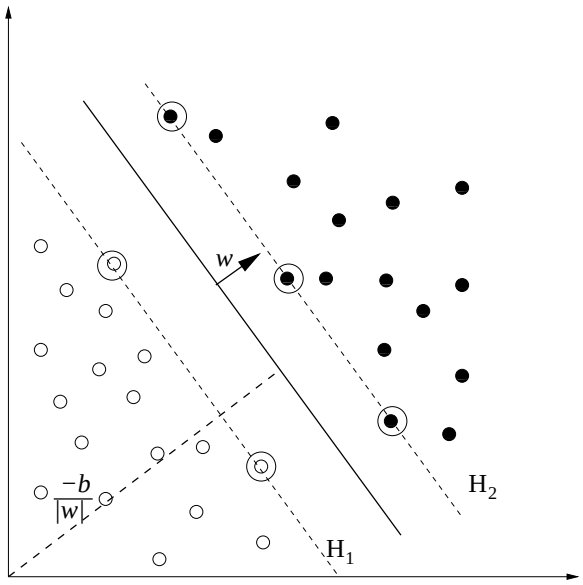

Figure 3.5: Maximum margin hyperplane for linear separable case

Programming (QP) optimisation [Cristianini and Shawe-Taylor 2000]. Solving this problem using the classical Lagrangian multipliers approach has a number of advantages [Cristianini and Shawe-Taylor 2000]. Firstly, this approach gives an alternative formulation of the original problem (dual form) which is easier to solve. Secondly, the dual form is not only easier to solve but also emphasises the importance of some training examples over the other, leading to a minimised but critical sample size and thirdly, the dual form simplifies generalisation beyond linear separable cases.

Introducing a dual vector of non-negative Lagrangian multiplier $\Lambda = (\alpha_1, \alpha_2, \ldots, \alpha_l)$ corresponding to each inequality constraint in (3.8) the constrained minimisation problem given above will be rewritten in the dual form as:

$$\text{Maximise} \qquad L_D(w, b, \Lambda) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i . x_j)$$

$$\text{subject to} \qquad \sum_{i=1}^{l} \alpha_i y_i = 0; \tag{3.11}$$

$$\Lambda \geq 0$$

The Karush-Kuhn-Tucker(KKT) optimisation theory (The KKT theorem is given in Appendix B.2), which guarantees the existence of a solution to the optimisation problem shows that, at the saddle point all points satisfy the constraint (3.8) with strict equality. i.e.

$$\alpha_i (y_i (w.x_i + b) - 1) = 0 \qquad i = 1, \ldots, l \tag{3.12}$$

From equation 3.12, the following two conditions need to be distinguished:

- If $\alpha_i = 0$, then $y_i (w.x_i + b) \geq 1$

- If $\alpha_i > 0$, then $y_i(w.x_i + b) = 1$

Recall that one of the advantages of using the Lagrangian function to solve the optimisation problem is expressing the importance of each pattern in the training set. Consider the value of α_i corresponding to each training pattern. Training patterns with $\alpha_i > 0$ will fall on the hyperplane H_1 or H_2 (see Figure 3.5) and hence are critical in defining the decision boundary. Other training patterns with $\alpha_i = 0$ lies to the left or right of H_1 and H_2 respectively. These training patterns have no effect in determining the decision boundary. Therefore, if those training patterns with $\alpha_i = 0$ value are removed and the training is repeated, the decision boundary will remain the same. Training patterns with nonzero α_i are called *Support Vectors* (the name of this learning technique follows from this).

With the orientation w given by $w = \sum_{i=1}^{l} \alpha_i y_i x_i$, the mapping function (equation 3.9) can be redefined as:

$$f(x, \Lambda, b) = sign(\sum_{i=1}^{l} y_i \alpha_i (x.x_i) + b) \qquad \forall i = 1, \ldots, l \tag{3.13}$$

We have seen that the parameter $\alpha_i = 0$ for all training points except for the support vectors, hence the mapping function will have its final form:

$$f(x, \Lambda, b) = sign(\sum_{i \in \mathcal{SV}} y_i \alpha_i (x.x_i) + b) \qquad \forall i = 1, \ldots, l \tag{3.14}$$

In other words the expression is evaluated in terms of the dot product between the pattern to be classified and the Support Vectors (SV) (x_i), and the sign of the function will be used to classify the pattern to their respective class.

3.3.2 Linearly non-separable case: Soft margin classifier

So far we have seen the case where the training data is perfectly separable using linear hyperplane. However, real-world problems involve non-separable data and the assumption taken in the previous section is too ambitious. To extend the above solution to non-separable data a positive slack variable $\xi_i : i = 1, \ldots, i$ is introduced to associate further cost as a penalty for misclassification whenever necessary (see Figure(3.6)).

Using this relaxed separation constraint equation (3.8) becomes:

$$y_i(x_i.w + b) > 1 - \xi_i \qquad \xi_i \geq 0 \qquad i = 1, \ldots, l \tag{3.15}$$

The problem of finding optimal margin will therefore comprise two parts.

- Maximise the margin (the same as the linear separable case) and
- Minimise the slack variable ξ_i which counts for amount of error

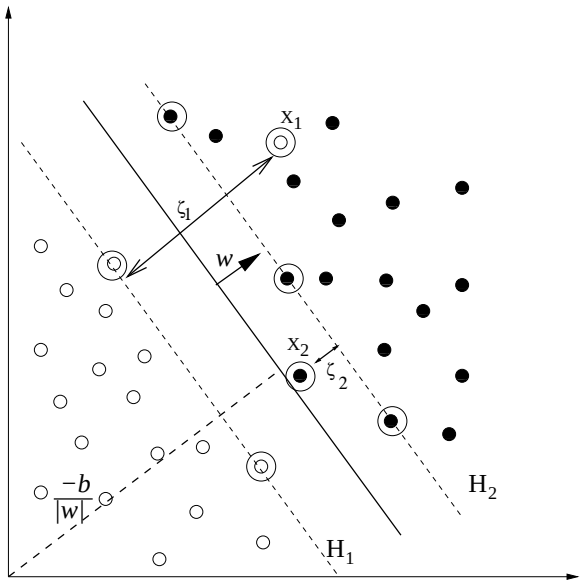

Figure 3.6: Maximum margin hyperplane for linear non-separable case

One way of combining these two conditions into a single function is given below:

$$\Phi(w, \Xi) = \frac{1}{2}\|w\|^2 + C(\sum_{i=1}^{l} \xi_i)^k$$

The constant C is a parameter to be chosen freely by the user to specify the trade-off between the width of the margin and misclassification penalty. Therefore the optimal hyperplane will be the one that minimises the function $\Phi(w, \Xi)$. i.e.

$$\begin{aligned} \text{Minimise} \quad & \Phi(w, \Xi) = \frac{1}{2}\|w\|^2 + C(\sum_{i=1}^{l} \xi_i)^k; \\ & y_i(x_i.w + b) > 1 - \xi_i \quad i = 1, \ldots, l; \\ & \xi_i > 0 \quad i = 1, \ldots, l \end{aligned} \tag{3.16}$$

By choosing $k = 1$, the above optimisation problem can be solved using QP. Introducing a dual vector of non-negative value $\Lambda = (\alpha_1, \alpha_2, \ldots \alpha_l)$ for of each the first constraint and $\Gamma = (\mu_1, \mu_2, \ldots, \mu_l)$ for each of the second constraint the optimisation problem defined above will be rewritten as:

$$\begin{aligned} \text{Maximise} \quad & L_D(w, b, \Lambda) = \sum_{i=1} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i.x_j) \\ \text{subject to} \quad & \sum_{i=1}^{l} \alpha_i y_i = 0; \\ & 0 \leq \Lambda \leq C \end{aligned} \tag{3.17}$$

Applying the KKT condition we have:

$$\alpha_i(y_i(x_i.w + b) - 1 + \xi_i) = 0 \qquad i = 1, 2, \ldots l \tag{3.18}$$

From the above equation three different cases needs to be distinguished:

- If $\alpha_i = 0$, then $\mu_i = C$ $(\xi_i = 0)$ and $y_i(x_i.w + b) = 1$;
- If $0 \leq \alpha_i \leq C$, then $0 \leq \mu_i \leq C$ $(\xi_i = 0)$ and $y_i(x_i.w + b) = 1$;
- If $\alpha_i = C$, then $\mu_i = 0$ $(\xi_i > 0)$ and $y_i(x_i.w + b) = 1 + \xi_i$.

In the first case, the points are on the correct side of the optimal hyperplane and are distant from the hyperplane by more than the margin (i.e. these points lie to the left or to the right of the hyperplane H_1 or H_2 respectively). In the second case, the points lie on the hyperplane H_1 or H_2 and are Support Vectors. In the third case these points are also support vectors, but do not necessarily lie on the hyperplane H_1 of H_2. These points might be on the wrong side of the hyperplane or on the right side but closer than the hyperplanes H_1 or H_2 (For example: X_1 and X_2 in Figure 3.6).

Besides the above additional constraints, the solution for the linear separable case holds with the decision function given by:

$$f(x, \Lambda, b) = sign(\sum_{i \in \mathcal{SV}} y_i \alpha_i (x.x_i) + b) \qquad \forall i = 1, \ldots, l$$

So far we have seen the simplest form of SVM which are designed for the most trivial linear separable case can be further extended to accommodate cases that are linearly non-separable. But most real life classification problems do not have linear dataset but rather non-linear dataset (see Figure 3.7). Hence, we still need to extend the solution to accommodate non-linear datasets. It has been stated at the introduction to this chapter that one of the underlying principle that makes SVMs interesting is the ability to transform non-linear data from input space to higher dimensional feature space where the data can be linearly classified. The following section gives brief introduction to feature space and kernel tricks and show how these concepts are used to generalise linear SVMs to handle non-linear classification problems.

3.4 Non-linear support vector machine: The Kernel trick

In most real life problems, linear combinations of the individual measurements cannot fully describe the properties of the patterns under consideration. Hence a complex representation of these measurements is required resulting in a complex structured dataset in the input space. As is pointed out in the previous chapter changing the representation of the data from the input space into some feature space has a number of advantages. Feature space, as discussed in the previous chapter usually have lesser dimensionality than the input space (as a result of feature

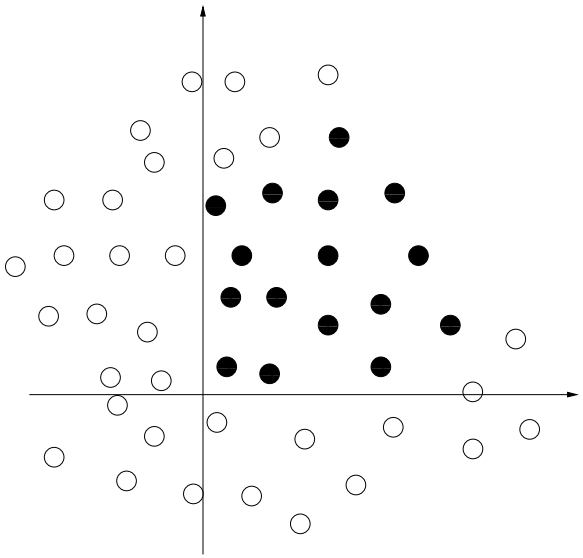

Figure 3.7: Linearly non-separable training sample

selection or extraction). On the contrary, in this section we will see how we can use a feature space with higher dimension than the input space to increase the computational power of the linear SVM described in the previous sections without suffering from curse of dimensionality.

Suppose the training set is $((x_1, y_1), \ldots, (x_l, y_l)) \in \mathbb{R}^d \times \{\pm 1\}$ and there is a function $\phi(x)$ and a mapping given by:

$$X = \{x_1, \ldots, x_d\} \rightarrow \Phi(X) = \{\phi(x_1), \ldots, \phi(x_N)\} \tag{3.19}$$

where $d < N$ and $F = \{\Phi(x) | x \in X\}$ is called the feature space.

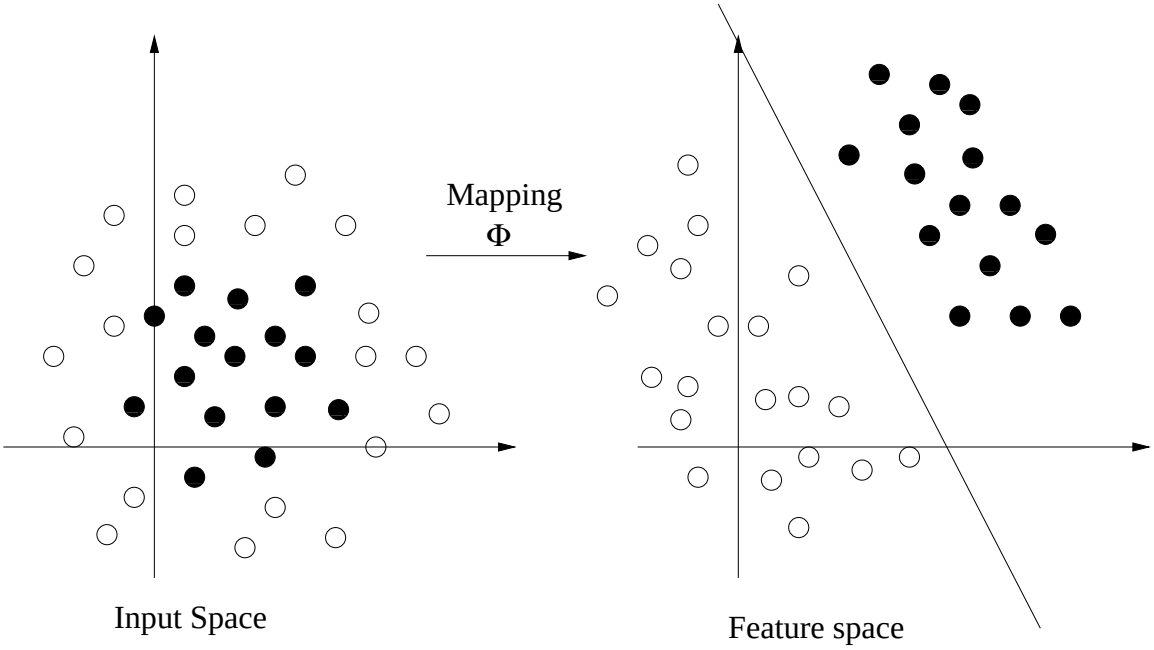

Figure 3.8: Mapping from two dimensional input space into two dimensional feature space where data can be linearly separated

Consider the mapping given graphically in Figure 3.8. It is easy to see that the data cannot be separated using a linear hyperplane in the input space. However, once it is projected to the feature space, separating the data using a linear hyperplane is possible. This simple example demonstrates the power of mapping a non-linear dataset into feature space in simplifying complex classification problems. As it is pointed out in section (3.3.1), one of the advantages of using the classical Lagrangian approach to solve the optimisation problem is simplifying the problem of extending the solution found for linear SVM into the generic non-linear SVM. Also note equation (3.13) that the decision function requires the computation of the dot product between the point to be classified and some of the training examples. Therefore, performing the classification in the feature space will give the hypothesis and the decision function a new form which is given by :

$$f(w, b) = sign(w.\phi(x_i) + b) \tag{3.20}$$

and

$$f(x) = sign(\sum_{i=1}^{l} y_i \alpha_i (\phi(x).\phi(x_i) + b)) \qquad i = 1, \ldots, l \tag{3.21}$$

The dot product $\phi(x).\phi(x_i)$ will be easily defined by introducing the function $K(x, y)$ called Kernel function (or more formally called Mercer Kernels to distinguish them from other kernels used in mathematics). For definition of kernel functions see see Appendix B.2.

The introduction of the kernel function allows us to compute the dot product without explicitly mapping the data into the feature space. To show that the kernel function represents the dot product in the feature space we will use Mercer's Theorem (see Appendix B.2), which guarantees the existence of a mapping ϕ in F for any kernel which is a dot product in some feature space [Ganapathiraju 2002].

To show how kernels are used to transform data implicitly into a higher dimensional space consider the following example:-

First lets how mapping from input to feature space simplifies the classification problem. Consider the target function:

$$f(x, y) = \frac{x^2}{y^2}$$

this target function could not be represeted by a linear machine. However a simple transformation can make it representable by a linear machine:

$$(x, y) \rightarrow (x', y') = (\log x, \log y)$$

gives the representation

$$f'(x', y') = \log f(x, y) = 2 \log x - 2 \log y$$

f' could be learnt by a linear machine.

Now suppose we want the mapping from $\mathbb{R}^2$ to $\mathbb{R}^6$. Let us choose the kernel to be:

$$K(x, y) = (x.y + 1)^2 \tag{3.22}$$

Choosing the kernel function given below:

$$(x.y + 1)^2 = \phi(x).\phi(y) \tag{3.23}$$

Let $x = (x_1, x_2)$ and $y = (y_1, y_2)$. This implies

$$(x.y + 1)^2 = x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 y_1 x_2 y_2 + 1$$

Now if we define

$$\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

it can easily be shown that equation (3.23) holds. Note that the above example can be easily generalised to higher dimensional space.

Having introduced the kernel representation of dot product in feature space, the decision function (3.21) can be rewritten using kernel functions as:

$$f(x) = sign(\sum_{i \in \mathcal{SV}} y_i \alpha_i k(x, x_i) + b)) \qquad i = 1, \ldots, l \tag{3.24}$$

Note that, SVMs can be used to classify non-linear data without the need to transform the input space to a high dimensional feature space explicitly using a kernel function. This strategy also removes the curse of dimensionality, which usually occurs as a result of dimensionality increase for a fixed number of sample size. Also note that only the support vectors are involved in the decision function.

Although kernel functions can be chosen freely, choosing some of the most commonly used kernels SVMs can represent other known classifiers with a better performance [Osuna *et al.* 1997]. Some of these kernels are:

- Polynomial Kernel of degree d

$$K(x, y) = (x.y + 1)^d \tag{3.25}$$

  where d is user-defined

- Gaussian radial basis function (RBF)

$$K(x, y) = exp(-\gamma \|x - y\|^2) \tag{3.26}$$

  where γ is user-defined

- Sigmoid kernel: Two layer Neural network

$$K(x, y) = \tanh(k(x.y) + \Theta) \tag{3.27}$$

where k is gain and Θ is the offset. Both of these parameters are user-defined

Ganapathiraju [2002] reports that though convergence for RBF is the most expensive, this kernel is very powerful and can model a classifier that can be effective when datepoints from one class is totally encapsulated by the other. An example of classification using polynomial and RBF is given in figure 3.9.

Other additional kernels (usually used in function estimation) includes [Vapnik 1995]

- Regularised Fourier (weaker mode regularisation)
  For one dimensional case:

$$K(x, y) = \frac{\pi}{2\gamma} \frac{\cosh \frac{\pi - \|x_i - x_j\|}{\gamma}}{\sinh \frac{\pi}{\gamma}} \tag{3.28}$$

  where $0 \leq \|x_i - x_j\| \leq 2\pi$ and γ is user defined
  For the multidimensional case $K(x, y) = \prod_{k=1}^{d} K_k(x^k, y^k)$

- Regularised Fourier (strong mode regularisation)
  For one dimensional case:

$$K(x, y) = \frac{1 - \gamma^2}{2(1 - 2\gamma \cosh(x_i - x_j) + \gamma^2)} \tag{3.29}$$

  where $0 \leq \|x_i - x_j\| \leq 2\pi$ and γ is user defined
  For the multidimensional case $K(x, y) = \prod_{k=1}^{d} K_k(x^k, y^k)$

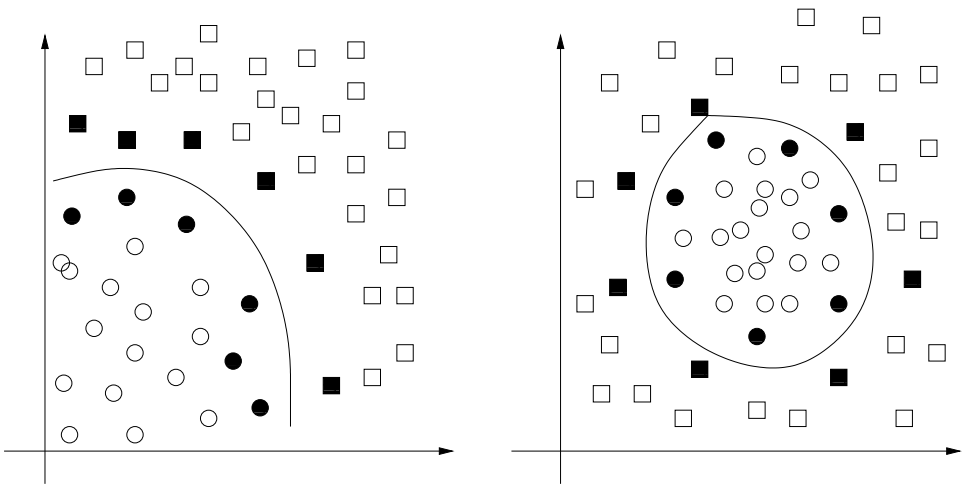

Figure 3.9: Decision surface given by (a) Polynomial kernel, and (b) RBF kernel. Support vectors are indicated by dark filled points [Osuna *et al.* 1997]

3.5 Multi-class classification

SVMs are inherently binary classifiers. The binary SVM discussed in the previous sections use the discriminant function given in equation 3.20. However, most classification problems are not limited to binary classification problems but to multi-class classification problems where the inputs are classified into more than two groups according to their underlying property. To solve this problem, extending the binary SVM to handle multi-class classification is required. The multi-class SVM uses a set of discriminant function $f_\omega : X \rightarrow \Omega$ defined as

$$f_\omega(w, b) = sign(w_\omega.\phi(x_i) + b_\omega)$$

where $X \subset \mathbb{R}^d$ and $\Omega = (\omega_1, \omega_2, \ldots, \omega_c)$ with the same performance evaluation technique. In multi-class SVM classification the decision function will have the form defined below.

$$f(x) = \arg\max_{\omega \in \Omega} f_\omega(x) \tag{3.30}$$

To implement this, a number of approaches have been proposed in the last decade. Some of these approaches are one-against-the-rest presented in Schölkopf [1997], classification by pairwise coupling proposed by Hastie and Tibshirani [1996] based on the idea proposed by Friedman [1996] and the Multi-class SVM proposed by Weston and Watkins [1998]. In this section a brief overview of these three approached will be discussed. For detailed presentation refer to the above cited references.

In the one-against-the-rest approach, a k-class classifier, is modelled by training k different binary SVMs each discriminating members of one class from the rest. This is done by relabelling the training data and assigning say 1 to the class under consideration and -1 to the remaining members of the training. A new data is then classified once it is tested using all the k SVMs and the final class is assigned based on equation 3.30. This technique is easy to implement, most widely used and gives a respectable result. It does however have some limitations [Schölkopf 1997]. Two of the limitations are, variation in the output range of the different SVMs, and mutual exclusion of class member. However, these two limitations have well formulated solutions. More description on the limitation and proposed solution can found in Lee *et al.* [2001] and Mayoraz and Alpaydin [1999].

Classification by pairwise coupling also uses a number of binary classifiers to accomplish the multi-class classification. However, unlike the one-against-the-rest approach, in this approach a number of different binary classifiers are trained to perform pairwise classification. That means to perform k class classification we need $\frac{k(k-1)}{2}$ binary classifiers each performing pairwise classification. Note also that in this approach each binary classifier is trained with a subset of the training data. For example if our training data is equally balanced between classes, we have $\frac{2}{k}$ fraction of the data to train each binary SVM. To classify an unseen pattern, the pattern will be tested against each class and is classified to the one which satisfies the voting criteria. Methods of evaluating the voting criteria are briefly described in Hastie and Tibshirani

[1996].

The third approach is the multi-class SVM proposed by Weston and Watkins [1998]. In this approach the mathematical formulation found for the binary SVM is extended to handle multi-class classification considering all the classes at once.

3.6 SVM implementation

Recall that the KKT condition (see Appendix B.2) guarantees an optimal solution to the constrained optimisation problems formulated in equations 3.11 and 3.17. As it can be seen from these formulations, in order to get an optimal solution computing the dot product $(x_i.x_j)$ between all possible pairs of the training sample is required. For a classification task with a large number of training samples and/or each training sample having a large number of features to represent it, computing the dot product matrix and keeping the result in memory will be both processor and memory intensive task and sometimes a specialised computing resource might be needed. A simple approach to solve the memory problem is computing the dot product on demand basis. However, this approach will make the computation even more CPU intensive.

To make this processor and/or memory intensive computation efficient, a number of approaches has been proposed based on the principle that a global solution can be obtained by solving a smaller sub-problem at a given time. However, they differ in the way they define sub-problem. In the remainder of this section an overview of the different algorithms will be presented.

3.6.1 Chunking

Chunking is the first approach considered in solving SVM learning problem with large training set proposed by Boser *et al.* [1992]. This approach is based on the idea of dividing the optimisation problem into small sub-problems (chunk) that can be solved efficiently. Training with this algorithm is done as follows. Training is started randomly with one sub-problem and then iteratively adding other examples that do not obey the KKT condition. Only support vectors found in the training stage of one sub-problem are to be carried out to the next stage. At the end of the optimisation process, when the KKT conditions are met, only the appropriate support vectors will be assimilated. This approach was proved by Osuna *et al.* [1997] to give the same global optimal solution and takes less resources to converge.

Further extensions of this approach are the decomposition method proposed by Osuna *et al.* [1997] and Sequential Minimal Optimisation (SMO) proposed by Platt [1998]. The decomposition method is based on selecting a working set when there are large number of support vectors that can be handled in memory. This working set is selected in such a way that it is big enough to hold all the support vectors, but small enough to be hand-fed by the computer [Osuna *et al.* 1997]. SMO is a chunking algorithm with the two working sets [Platt 1998].

It is evident that chunking has computational advantage over the naive approach when the

training data is big. And this advantage is further extended by the decomposition method and SMO [Kroon 2003].

3.6.2 Shrinking

The linear memory requirement proportional to the number of training examples and support vectors has many advantages. However, the algorithm may take longer to converge. To overcome this potential disadvantage, Joachims [1999] proposed shrinking. Shrinking is based on predicting data points that will not be support vectors and eliminating these points during optimisation. Eliminating these data points will result in a smaller sub-problem (shrinking of the optimisation problem). Furthermore, this algorithm uses the concept of "Bounded Support Vectors" (BSV), which are support vectors with α_i at upper bound C.

To extract these points the algorithm uses a heuristic approach to study the behaviour of points for a number of iterations. On each iteration only those points which behave consistently are extracted, which consequently shrinks the original problem [Kroon 2003].

3.6.3 Caching

The optimisation process requires evaluating the kernel matrix in each iteration, which is the most expensive process. One heuristic approach to speed up this process is caching kernel evaluation, which trades-off between memory consumption and training time [Joachims 1999]. Caching is a well known technique implemented to speed up memory intensive computations and is widely used in many applications of computer science. There are a number of caching strategies such as First In First Out (FIFO), Least Recently Used (LRU), optimal, etc. The caching strategies used in SVM implementation is LRU, which replaces elements that are not used for a number of iterations, whenever the cache is full.

3.7 Comparison of SVMs to other statistical techniques

In recent years a large variety of pattern recognition techniques have been applied to solve a wide variety of problems. These includes decision trees, neural networks, logistic regression, and SVM. Even though the choice of a particular technique depends on the problem at hand, some techniques are reported to have number of theoretical advantages over the others. One such technique is SVM. Many researchers have pointed the significant advantage of SVM in a wide variety of benchmark application. Furthermore, researchers in the field of computational biology have also consistently shown the outstanding performance of SVMs.

In the last couple of section detailed theoretical background and an overview on the implementation of SVMs was presented. Furthermore, in chapter 2 basic background of decision trees and neural networks was presented. In the remainder of this section a theoretical comparison showing some of the advantages and disadvantages of SVMs compared to decision trees and/or neural networks is presented.

Linear or non-linear model

A true challenge for most pattern recognition techniques is based on how they handle non-linear real life problems. While some techniques can handle non-linear classification directly, other need to approximate or use some techniques to convert the non-linear classification into a linear classification problem.

Decision trees handle non-linear classification problem by approximating the problem with pairwise linear classification [Breiman *et al.* 1984]. Similar to decision trees, SVMs cannot handle non-linear data directly. To address the problem, SVMs map the non-linear data into a high-dimensional feature space where it can be classified linearly (or more specifically SVMs uses kernel function to classify non-linear data without the need for explicit mapping). However, neural networks do not have this problem and address non-linear classification problem directly using multiple hyperplanes by the help of different activation functions such as Sigmoid, Gaussian or Radial Basis functions [Jain *et al.* 2000].

Error minimisation

As pointed out in section 3.2 all pattern recognition techniques are aimed at increasing the performance (minimise error) of the classifier based on the limited training sample available. It was also pointed out in section 3.2 that there are two principles of risk minimisation: namely empirical risk minimisation and structural risk minimisation. Both neural network and decision trees use the principle of empirical risk minimisation and are most likely to suffer from over-fitting [Jain *et al.* 2000]. However, SVMs use the principle of structural risk minimisation.

To address the over-fitting problem both these methods have adopted different techniques. Decision trees use two different techniques. The first technique is stop growing the tree further when the split is not statistically significant [Breiman *et al.* 1984]. The second approach is first constructing the tree and then pruning the fully grown tree upward by considering a subtree with minimum accuracy loss [Breiman *et al.* 1984; Quinlan 1993]. One can address the over-fitting problems in neural networks either by choosing an appropriate number of hidden layers or stopping the training process before the network is fully trained. Even though the suggested techniques for both decision trees and neural networks have helped minimise the possibility of over-fitting, the problem is still there as the result of the basic principle of risk minimisation. Furthermore, there is no robust way of selecting a criterion to stop the training process so that the network will not be over trained, or stop pruning.

As mensioned in section 3.2.2, SVMs use the principle of structural risk minimisation to avoid over-fitting. However, over-fitting is not absolutely avoided. A factor that can cause over-fitting in SVMs is selection of poor kernel function which will result in number of support vectors comparably equal to the size of the training samples. To address this problem new data driven kernel selection techniques have been proposed by Sollich [2000] which gives an intuitive guideline to choose a good kernel function.

Global vs local solution

Depending on the approach taken to solve the optimisation problem, pattern recognition techniques give global or local minimum. Neural networks use gradient descent techniques which gives the local minimum [Vapnik 1995]. On the other hand SVMs use quadratic programming technique which is guaranteed to give the global minimum [Burges 1998] (also see previous number of sections).

Multi-class classification

Most real-world problems are not bound to binary classification problems. Therefore, one needs to consider how multi-class classification is considered when comparing different techniques.

While both decision trees and neural networks can be trained to perform $1 - to - n$ classification, SVMs can only perform $1 - to - 1$ classification. This means for SVMs to perform multi-class classification one needs to train n different classifiers each performing binary classification and hence is computationally demanding.

Performance comparison

Generally both decision trees, neural networks and SVMs have been applied in a many classification problem and the latter two have been the two competitive techniques recently. To show the dominant performance of SVM over the decision trees and neural networks, consider table 3.1 showing selected results from Meyer *et al.* [2002].

Problem	SVMs	Decision trees	Neural Network
BreastCancer	3.14	5.51	24.27
tictactoe	0.14	8.24	33.97
chess	0.49	3.20	39.73
titanic	21.16	21.48	66.90

Table 3.1: Mean error on a test set [Meyer *et al.* 2002]

The result in table 3.1 shows that despite the minor limitations discussed above, SVMs have outperformed the other two classification techniques.

3.8 Previous work

3.8.1 SVMs benchmark applications

So far we have seen the theoretical background and comparison of SVM. In this section, we will see the application of SVM as a tool for pattern recognition.

Text classification

The volume of information found in electronic format has increased incredibly as a result of the fast growing Internet. This fast growing volume of electronic information has urged many to look for a better way to organise these resources so as to find and filter for particular information in an effective and efficient way. Text classification or categorisation can be defined as pattern recognition problem, which classifies natural-language text to one or more predefined classes according to their content [Dumais *et al.* 1998]. So far, much text classification work is done manually, which has created a bottleneck in efficiency and effectiveness in dynamic information.

There are a number of systems already in use to organise electronic documents in such a way that filtering and searching can be done faster and accurately. The speed and accuracy of such systems have a very high impact on their acceptance and usefulness. To increase the speed and accuracy of such systems and development of new systems, many have proposed machine learning techniques. Dumais *et al.* [1998] and Joachims [1998] have shown the performance of machine learning techniques (naive Bayes classifier, k-nearest neighbours, decision tree classifiers and SVM) in text classification in terms of *precision/Recall-Breakeven Point*. In addition, Sun *et al.* [2002] have used SVM to classify the web documents. In the next paragraph, we will see the application of SVM in text classification in more detail giving particular emphasis on the concepts like feature extraction and kernel used and the results obtained.

Joachims [1998] conducted his experiment on 12902 documents (75% training and 25% testing) from Reuters-21578 dataset and 20000 documents (50% training, 50% testing) from Ohsumed corpus. To represent the text document Joachims [1998] used frequency of appearance of the words in the document with the restriction of frequency greater than two to avoid unnecessary large feature vectors. Having represented the document, further features were selected using a method of feature selection called information gain, which ranks all the words according to their information gain and selects those with highest the mutual information. Similarly Dumais *et al.* [1998] used the same number of datasets from Reuters-21578[2] collections and used binary feature value (occurs or does not occur) to reduce the feature space. Both Joachims [1998] and Dumais *et al.* [1998] conducted the experiment on a number of different machine learning techniques: naive Bayes classifier, k-nearest neighbours, decision tree classifiers and SVM. Joachims [1998] used both polynomial kernel of degree $(d = 1, 2, 3, 4, 5)$ and RBF with $\gamma = 0.6, 0.8, 1.0, 1.2$ while Dumais *et al.* [1998] used only linear SVM. The result found in both experiments have confirmed that SVM is superior in performance and training time over the given training set. Furthermore, Sun *et al.* [2002] have extended text classification technique to web classification and have shown that SVM outperforms FOIL-PILFS[3] algorithm.

[2]Due to the different time in the experiment the Reuters collections used in the experiment by Joachims [1998] and Dumais *et al.* [1998] are different

[3]FOIL-PILFS is an algorithm, which is designed to learn rules, which use predicates based on Naive Bayesian models of text instead of keyword tests.

Microarray gene expression analysis and cancer tissue classification

In the previous section, we have seen how SVMs are applied to text recognition and it is emphasised that SVMs perform better than other machine learning techniques when used as a pattern recognition tool. This section shows how SVMs are used to analyse microarray gene expressions [Brown *et al.* 2000] and used to classify cancer tissue samples based on microarray expressions [Furey *et al.* 2000].

As the amount of data from different microarray hybridisation experiments become huge, the need for a means to extract biological significance and classifying genes according to their functional class is increasingly becoming imperative [Brown *et al.* 2000]. There are a number of approaches available to handle such task. Most of these approaches use a clustering algorithm based on the similarities between expression patterns to group genes. However, these approaches have a number of limitations. These limitations are the motivations for the research by Brown *et al.* [2000]. They used SVMs to perform microarray gene expression analysis.

They pointed out that SVMs have the advantage of using a large set of similarity computing functions simultaneously and the ability to use prior knowledge about the true functional classes of the genes. The experiment was based on 2,467 yeast genes, which were selected based on availability of accurate functional annotation. These data sets were converted into 79 element gene expression vectors based on the results from an experiment with $n = 79$ genes on a single chip, which is converted to a vector form by dividing the expression level of the gene in the varying condition of interest by the expression level of the gene in some reference condition[4]. Having represented the data with the appropriate vector form, the set of genes that have common functional class are labelled positive, and negative otherwise.

Based on these examples the SVM is taught to discriminate the positive from the negative examples. To carry out the learning task, polynomial and Gaussian kernels were used. One problem identified was that the set of positive examples was less in number compared to the negative ones and hence were treated as noise rather than examples belonging to a separate class. To handle this problem Brown *et al.* [2000] modified the kernel value during the support vector optimisation by adding the ratio of the positive examples multiplied by a scaling factor to the diagonal elements of the matrix defined by the kernel function. The result showed that higher degree polynomials and Gaussian kernels give a superior result as compared to previous analysis techniques.

As a continuation of the research by Brown *et al.* [2000], Furey *et al.* [2000] applied SVMs in classifying cancer tissues based on the microarray expression data. The features extraction criteria used are different from that of Brown *et al.* [2000]. Furey *et al.* [2000] used the ratio of the difference between the mean of the positive examples (cancer tissue) and the negative examples (normal tissue) and the sum of their standard deviation[5], which helped discriminate

[4] $X_i = \frac{log(\frac{E_i}{R_i})}{\sqrt{\sum_{j=1}^{79} log^2(\frac{E_j}{R_j})}}$

[5] $F(X_j) = \frac{\mu_j^+ - \mu_j^-}{\sigma_j^+ + \sigma_j^+}$

between the two classes. Genes with the highest score of the ratio were extracted.

Unlike Brown *et al.* [2000], Furey *et al.* [2000] used only simple dot-product kernel and were not only able to classify tissues to the appropriate class but were also able to detect outliers in the example set. The result was confirmed using biological experiments.

3.8.2 Other machine learning techniques in predicting drug resistance

In the previous section, we have seen the power of SVM as a tool for pattern recognition and have showed some application of SVM in bioinformatics and real-world pattern recognition problems. In this section, we will see how other machine learning techniques are used to predict drug resistance of HIV mutants. Among many researches carried out in this particular topic I will try to briefly present the methods used, features targeted, results achieved and the limitations of the research by Lathrop *et al.* [1999] and Draghici and Potter [2003]. Draghici and Potter [2003] have taken two different methods: Structure-based data mining and Sequence-based data mining. In this literature, we are interested in the Sequence-based method, which is based on the amino-acid sequence rather than the structural features.

Lathrop *et al.* [1999] used rule-based expert systems based on the set of 55 rules extracted from different literatures on mutation and drug resistance and used both RT and PRO portions of the POL gene to predict drug resistant and nearby drug resistant mutants over all FDA[6] approved HIV drugs (11 at the time of the research) to suggest optimal combination of drugs for a patient under therapy. On the other hand Draghici and Potter [2003] used Neural Networks to predict resistance to *Saquinavir* using HIV Protease amino-acid sequences so that the HIV mutant is classified as low, medium or high resistant.

The first step taken by both experiments was extracting features that can represent the input sequence, where both used the codons (433 codons in RT and 99 in PRO). Lathrop *et al.* [1999] extracted the codons that are targeted by the approved drugs, which reduced the input to 31 different codon positions (20 in RT and 11 in PRO). Draghici and Potter [2003] extracted all the codons in PRO and assigned a value between 0 and 1 based on their difference from the wild type [7] (HXB2). Moreover, two codons, which do not show any variation from the wild type were eliminated resulting in 97 codon positions in the input vector. Each input vector was then converted into a normalised numeric pattern to make it suitable for Neural Network processing. Having extracted the relevant features Lathrop *et al.* [1999] applied the 55 set of rules in the form of *IF* $\langle$ *antecedent* $\rangle$ *THEN* $\langle$ *consequence* $\rangle$ *WITH* $\langle$ *weight* $\rangle$ with a high level of confidence, where the weight varies from 0.1 (low resistant) to 1.0 (high resistant). Based on the rule weight the current resistance of the mutant is calculated. The nearby resistant mutants were predicted by applying the rule on newly generated sequences from those sequences that originally do not trigger the rules. The neural network approach taken by Draghici and Potter [2003] was different from the rule-based expert system. In their research, Draghici and Potter

[6] Food and Drug Association

[7] "value is assigned between 0 and 1 in n equal increment where n is the number of different mutations from the wild type"[Draghici and Potter 2003, page 103]

[2003] used thirty-six network architectures with different output matrix (12×12 to 3×3), different learning rates (0.9 to 0.5) and different initial neighbourhoods. Each network is trained 32 times with 75 % of the data with 10 iterations (except one).

The result found by both research approaches confirms that drug resistance can be predicted using the protein sequence of the HIV mutant. Results found by Lathrop *et al.* [1999] showed up to $25\times$ viral load reduction in patients that completed one year of therapy of optimal drug combination recommended by the rule-based expert system. Likewise, the results from Draghici and Potter [2003] showed 69% coverage and 68% accuracy on a single network (8×8 output matrix, 0.6 learning rate and 8 neighbourhoods) and as high as 85% average coverage and 78% accuracy over multiple networks.

Although both research methods showed positive results there are some limitations. The rule-based expert system needs continuous maintenance with addition of new rules and discovery of new mutants and the neural network approach is theoretically hard to verify.

3.9 Summary

In this chapter a detailed background and mathematical formulation of SVM was presented. The two principles of error minimisation was covered in depth in the first section of the chapter. The simplest form of SVMs, the maximum margin classifiers was described in more detail and was also used to introduce a number of new terms and techniques that was latter used to address the more complex model if SVMs, the non-linear SVMs. This chapter has also covered how the inherent binary classifier is extended to multi-class classification and an overview on the implementation details of SVMs. Finally, comparison of SVMs with decision trees and neural network was given followed by some relevant previous works done on SVMs as a pattern recognition tool and HIV drug resistance prediction.

In addition to the SVMs background and benchmark applications given so far, the next chapter will motivate why SVMs are particularly an ideal choice for the pattern recognition this research is solving.

Chapter 4

The Experiment

4.1 Introduction

In the previous chapters, detailed background on the HIV biology and a brief discussion on how this problem is best solved using pattern recognition techniques were presented. Different approaches in pattern recognition with more emphasis on statistical pattern recognition and SVMs as a classification algorithm were discussed. The different pattern recognition techniques discussed in Chapter 2 and Chapter 3 have been applied in a number of bioinformatics application and all of them have registered good results. However, depending on the type of problem at hand, the number of training patterns, the dimensionality of the problem, one technique outperform the other with no one technique showing absolute superiority. A brief theoretical comparisons of some of these techniques were also presented. The results extracted from previous research (Table 3.1) and the theoretical comparisons presented reasons out why SVMs are likely to perform better than the traditional pattern recognition techniques. This research compared different pattern recognition techniques on the HIV drug resistance problem but it is not the intention of this research to rank the different techniques in this problem. The main objective of this work is to assess the performance of SVMs as a tool for HIV drug resistance prediction.

In this Chapter a recap on the different techniques of HIV drug resistance testing and their limitations, why the problem is well suited to be solved as a pattern recognition problem and some of the benefits of using SVMs as a classification technique are presented before formulation the research question. Performance evaluation of classification algorithms can be based on a number of different metrics. This chapter discussed the approach taken to measure the performance of the different algorithms and motivates why the approach was chosen. One of the major issues in pattern recognition problem is the input encoding technique. Input encoding have a major effect on the performance of the classifier. Input data and encoding is also discussed in this Chapter. Moreover, this Chapter will give a detailed background on different model selection, error estimation and generalisation methods.

In the next section the research question will be motivated and the corresponding research hypothesis will be formulated. Section 4.3 gives description of the implementation of the dif-

ferent classification algorithms used for this comparative study. Section 4.4 gives a detailed background on different approaches on model selection, error estimation and generalisation. Section 4.5 gives the evaluation criteria for this research and motivates why a particular evaluation criterion is adopted. Section 4.6 will describe the data to be used and motivate why this data is used and the selected input encoding scheme. In section 4.7 an overview of the approach to be taken will be given. Finally a Section 4.8 concludes the chapter.

4.2 Research question

The AIDS epidemic has already caused a major economic, political and social problems all over the world specially in developing countries. Although there are a number of reasons for the failure in AIDS treatment, the high rate of mutation in the genetic code of the HIV virus is considered the major cause and is the main focus on combating the epidemic. Mutations are frequently exhibited on the genetic codes encoding the viral reverse transcriptase (RT) or protease (PRO) proteins and some of these mutations are known to confer drug resistance. Since not all mutations cause drug resistance the identification of those particular mutation points known to confer drug resistance will have advantage both during drug designing and clinical therapy of an HIV patient. The two ways of accomplishing this are phenotypic and genotypic testing. In genotypic testing the viral reverse transcriptase and protease are sequenced and checked for existence of mutations known to confer drug resistance and hence indirectly predict the drug resistance behaviour of the HIV strain upon therapy.

The two approaches for drug resistance testing mentioned above have their advantages and disadvantages compared to each other (for more detail see section 2.3.4). One of the advantages of genotypic approach is that it is well suited for a computerised approach. This advantage is the reason for the involvement of non-biological disciplines in addressing the drug resistance problem. Some of the techniques adopted from other disciplines are expert systems and a number of pattern recognition techniques. These proposed and already implemented systems for predicting the drug resistance behaviour based on sequenced reverse transcriptase or protease of an HIV strain extracted from a patient have already improved pharmaceutical therapy of the patients. However, these systems are still in a-work-in-progress stage and have a number of both biological and computational limitations. Firstly, most of the existing application to predicting HIV drug resistance are based on rule-based algorithms [Ravela *et al.* 2003]. These applications are designed to predict the drug resistance of an HIV strain based on the sequenced reverse transcriptase or protease of the virus, highly rely on the biological knowledge available about the drugs and sets of mutation points known to confer resistance to these drugs. Therefore, the steady growth of mutation points causing drug resistance, due to failed retroviral therapy is negatively affecting the performance of these systems. To address the effect of the newly discovered mutations and incorporate the discovered biological information (about the drugs and new mutation points known to confer drug resistance), redesigning the systems might be required. Furthermore, as presented by [Ravela *et al.* 2003] this approach has resulted in differ-

ent interpretations of the biological knowledge resulting in discordance between the different interpretation algorithms. Secondly, the growing number of mutation points, the dimensionality of the sequenced reverse transcriptase and protease genetic codes, the commonality of noise in most biological data, together with the cost and politics of data generation, leave these techniques to make the maximum out of a limited (usually small) training samples. As mentioned before many of the already existing techniques (based on neural networks and decision tree) rely on probabilistic density estimation as a statistical tool for prediction. This approach leaves these techniques to suffer from the "curse of dimensionality due to the big ratio difference between the dimensionality of the data and the number of available training samples (high dimensional and limited samples). Furthermore, most of these techniques use the principle of empirical risk minimisation (presented in section 3.7), which makes them highly vulnerable to over-fitting. Thirdly, discovery of additional drugs will also impose additional restrictions on these systems.

To overcome the above specified computational limitations there are a number of well studied computational approaches that could be considered during model designing. Some of these approaches are reducing the dimensionality of the patterns, introducing steps to avoid over-fitting during the training process and incorporating advanced techniques to enable these techniques to accommodate newly discovered mutation points. One can also explore different pattern recognition techniques theoretically known to address the computational limitations specified above and have shown good performance in other (related) pattern recognition tasks. This research considers SVMs, a pattern recognition technique which has been outperforming other pattern recognition techniques in many benchmark applications and computational biology problems. As specified in the previous chapter, the principle of structural risk minimisation, the fact that, only support vectors determine the classification model and the kernel trick have enabled SVMs outperform other pattern recognition techniques in a number of classification problem where the number of available samples is limited, high dimensional and noisy. Moreover, SVMs are pattern recognition technique with a strong mathematical foundation to achieve good generalisation while maintaining a high classification accuracy [Ganapathiraju 2002]. For the above mentioned reasons, SVMs have been recently applied as a classification tool in a number of pattern recognition problems. In section 3.8.1 a couple of applications of SVMs on problems in bioinformatics and computational molecular biology was presented. Additional applications of SVMs on this domain can be found at `http://www.support-vector.net/bioinformatics.html`.

In section 1.4 we have seen how HIV drug resistance prediction is mapped to a general pattern recognition problem. Therefore, this research explores the performance of SVMs in predicting the drug resistance behaviour of an HIV strain extracted from a patient to the different reverse transcriptase and protease inhibitors based on the genetic codes of the viral reverse transcriptase and protease respectively. Furthermore, we want to explore the possibility of designing a model without incorporating the already existing biological knowledge to have a model capable of accommodating not yet discovered mutations but contributing to the drug resistance behaviour of the strain to these drugs. Following the above argument the research

question for this study can be posed as follows:

> Given a set of drug resistance data pairs $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\}$, where x is a valid genetic sequence of HIV reverse transcriptase or protease for an HIV strain and $y = \{\text{Susceptible, Resistant}\}$ is the drug resistance behaviour of the corresponding strain for the drug under investigation: can SVMs learn from these examples and predict the drug resistance of unseen HIV strains?

To answer this question, the performance of SVMs was evaluated based on accuracy of classification and the trade-off ability between the sensitivity and specificity (the performance evaluation criteria is presented in section 4.5). Evaluating the performance of SVMs alone will fall short of answering the research question. Therefore, in addition to SVMs, a number of different pattern recognition techniques; namely: decision trees, neural networks and logistic regression methods was investigated. Previous research by Beerenwinkel *et al.* [2002] showed the capacity of decision trees for predicting HIV drug resistance based on the genetic sequence of the viral protease or reverse transcriptase. Similarly, Draghici and Potter [2003] and [Wang and Larder 2003] showed the capacity of neural networks for HIV drug resistance prediction using the genetic sequence of the viral protease and reverse transcriptase and the presence of mutation on specific positions on the viral protease known to confer drug resistance respectively. For these reasons, the answer to the research question is affirmative or othewise based on the performance of SVMs incomparison to decision trees, neural network and/or logistic regression. If SVMs perform equally or better than these techniques, the research question will be answered affermative.

The performance of the different classification algorithms was evaluated using the same performance evaluation technique. The best configuration for each of these algorithms was tuned. And finally comparison of performance was done. The search for the best configuration for the different algorithms have resulted in different data pre-processing steps for the different classification algorithms, however it did not compromise the performance comparison since the focus of the research is accessing the performance SVMs in comparison to the other popular pattern recognition techniques but not ranking the algorithms according to their performance.

4.3 Classification algorithms description

As presented in Chapter 2 and Chapter 3 pattern recognition/classification techniques can be categorised differently based on the underlying background, the approaches taken, the assumptions made, etc. Furthermore, a single classification technique in turn can be implemented in a number of different ways each making a distinct design decision and assumption based on the underlying principles of the technique. Some implementations are designed to be more efficient on large data sets while others are implemented to have fast training time. The underlying background that lead to the different techniques is presented in the previous chapters. In this

section, a brief description and a high level comparison of the different classification algorithm used in this research will be presented.

The following is the list of algorithms used in this research:

1. Support Vector Machines (SVMs)
    - SVM^{light} [Joachims 1999]
2. Neural Networks
    - Feedforward Multilayered perceptron [Hagan and Menhaj 1994; Riedmiller and Braun 1993]
3. Decision Trees
    - C4.5 [Quinlan 1993]
4. Regression methods
    - Penalised logistic regression [Zhu and Hastie 2004]

Support vector algorithms: SVM^{light}

As presented in Chapter 3 finding the optimal solution that satisfies the constrained optimisation problem is computationally expensive. To address this problem, a number of different implementation of SVMs that differ not only in the approach taken to divide and conquer the problem but also implemented for different platforms and problem size are proposed. A list of different SVMs implementations and utilities can be found from `http://www.kernel-machines.org/software.html`. The different approaches taken to make this CPU and memory intensive computation more efficient were discussed in Chapter 3. In this section a brief description of SVM^{light}, an SVMs implementation used in this research will be presented.

SVM^{light} written by Joachims [1999] is a C implementation of support vector machine [Vapnik 1995]. SVM^{light} is reported to be the most popular implementation of SVM classification technique for a number of reasons. Firstly it is capable of learning of ranking functions, in addition to classification and regression. Secondly, it has scalable memory requirements. SVM^{light} used least-recently-used caching strategy to trade-off between memory consumption and training time. This property enables the algorithm effectively handle large range problems with many thousands of support vectors. The core optimisation method used in this algorithm is based on 'LOQO' algorithm, which is a software package implementing infeasible-primal-dual path following method to solve nonlinear optimisation problems [Vanderbei 1999, as cited in Joachims [1999]]. Thirdly, it also provides methods for assessing the generalisation performance using leave-one-out cross-validation technique. It also gives both error rate and precision/recall on the training and test set

SVM^{light} has a method to classify unseen problems based on the constructed model. Unlike many classification algorithms which return ± 1, SVM^{light} returns continuous-valued output $(-\infty, +\infty)$, which makes it more suitable than other SVM implementation for computing the area under the ROC curve.

For non-commercial use SVM^{light} is freely available at `http://svmlight.joachims.org`.

Neural Networks: Resilient and Levenberg-Marquardt methods

The most popular neural networks architecture over a wide range of real-world applications including computational biology problems is the feedforward multilayered perceptorns with backpropagation algorithm [Baldi and Brunak 2001]. As mentioned in Section 2.6 the standard backpropagation algorithm is too slow for real-world problems. The two alternative ways of modifying this algorithm are heuristic approach or numerical optimisation. The Resilient backpropagation implements the first approach while Levenberg-Marquardt backpropagation algorithm that implement the later. Details of these two algorithms were presented in section 2.6.

The Matlab neural netwoks toolbox is used for this experiment.

Decision Trees: C4.5

As presented in Section 2.7 there are several heuristics methods in constructing decision tree each differing in the approaches taken in the construction step. The decision trees algorithm used in this research is C4.5 written by Quinlan [1993] as replacement for his original implementation ID3 (iterative dichotomizer 3^{rd}). C4.5 is a top-down induction of decision tree algorithm where for a given set of labelled examples the decision tree is constructed in a top-down fashion. Unlike its predecessor this algorithm can handle both nominal and continuous attributes.

The splitting criteria used by C4.5 is information gain ratio. C4.5 uses this important metric to avoid the problem that arises from using information gain as a split criteria. Recall from Section 2.7 that information gain favours attributes with distinct values. A fully grown decision tree is the pruned to reduce the size of the tree and avoid possible over fitting. C4.5 prunes the fully grown tree with out the need for a separate validation set. The C4.5 algorithm also has the ability to handle missing attributes.

Release 8 of the algorithm can be found in `http://www.rulequest.com/Personal/`.

Regression Model: Logistic regression

In this research we used penalised logistic regression methods. The version of penalised logistic regression used in this research is proposed by Zhu and Hastie [2004] as an alternative for SVMs for microarray cancer diagnosis problem. This algorithm uses Sequential Minimal Optimisation

(SOM) proposed by Platt [1998] to make it computational feasible for problem with larger attribute set.

The Matlab implementation of the algorithm is available at `http://www.tsi.enst.fr/~gfort/GLM/Programs.html`

4.4 Model selection, generalisation and error estimation

Model selection is one of the primary steps in pattern recognition application. This step includes selection of a particular pattern recognition approach (for example: statistical or structural pattern recognition), a particular pattern recognition technique within the selected approach (for example: neural networks or SVMs). Once the appropriate classification algorithm for the problem is identified, parameter tuning is carried out to determine the best configuration that gives the best performance. The best configuration parameters which is likely give better performance not only on the training sets but also unseen problems can be selected based on the theoretical bound on generalisation or error estimation on the training set.

Generalisation and error estimation are important because they increase the confidence in the model selected. There are a number of ways of error estimation and generalisation. Estimating the bound on the generalisation requires detailed theoretical information on the problem to be solved such as the underlying distribution probability which is generally difficult to compute from limited training samples. Therefore, estimating the bound is not an easy task for most pattern recognition tasks. However, error estimation is easy to conduct compared to evaluating the bound on generalisation. The simplest method of error estimation is cross-validation. This method is used in this experiment and is described in detail. There are more sophisticated methods in deriving the bound on the generalisation error and tuning the classifier parameters. Such methods of error estimation and generalisation methods include PAC bound, complexity penalisation and Bayesian model selection. However, these methods require a detailed knowledge about the behaviour of the data and the underlying distribution. The underlying distribution of the data is usually unknown and hence error estimation using cross validation is used in many practises.

4.4.1 Hold-out and cross-validation method

Cross validation is a model evaluation method that does not apply the entire data set to training the selected model. In this method some of the available samples are removed from the training set. Once the model under consideration is trained, those samples that were removed will be used to test the performance of the trained model. This is the basic idea behind the model evaluation methods called cross validation. However depending on the nature of the division of the training sample between the training and testing sample, we have three different cross-validation techniques. These techniques will be described next. The material in this section is based on Goutte [1997], Plutowski *et al.* [1994], Shao [1993].

Hold-out method

The hold-out method is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. The pattern recognition system will be trained using the training set only. Then the system will be tested to classify those samples in the testing set. The performance of the system will be evaluated by the mean absolute test set error. One disadvantage of this method is the consistency of the result. The evaluation may depend heavily on which points are used as a training set and which points are used as test set. This means evaluation may be significantly different depending on how the available sample is divided between the training set and the testing set.

k-fold cross validation

One way to improve the limitations of the hold-out method is the k-fold cross validation method. In this approach the data set is divided into k disjoint subsets. Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are put together to form a training set. That means the hold-out method is repeated k times. The average error over all k trials will be used to evaluate the model under investigation. The advantage of this method is that, unlike the hold-out method, how the data gets divided is less significant. Every data point gets to be in a test set only once, and gets to be in a training set $k-1$ times. Hence the consistency of the resulting estimate is increased as k is increased. One disadvantage of this method is that the training algorithm has to be run for k times for each training-testing set combination and takes k times as much computation to make an evaluation.

A variation of this method is to randomly divide the data into a test and training set k different times. This approach have the advantage that you can independently choose how large each test set is and how many trials you average over.

Leave-one-out cross validation

Leave-one-out cross validation is k-fold cross validation taken to its logical extreme, with k set to be equal to n, the number of data points in the set. This means the pattern recognition system is trained on all the data except for one point n separate times and a classification is made for that point. As before, the average error is computed and used to evaluate the model. The evaluation given by leave-one-out cross validation error is good, but more expensive to compute than the other two methods.

So far we have seen the different techniques of cross-validation. Although leave-one-out technique is the most powerful, it is also the most expensive. Therefore in this research, the second variation of k-fold cross-validation where 75% of the data will be used for training and the remaining 25% will be used for testing. The performance will be the average of 10 trials.

4.5 Performance evaluation criteria

The main objective of this research is to investigate the performance of SVMs relative to some of the popular machine learning/classification techniques. Like most research on empirical comparison of classification techniques, the major task of this research is investigating the best classification technique/algorithm along with its optimal parameter sets that not only give the best performance on the given data sets but also is likely to have better generalisation. In the last couple of decades, a number of articles comparing the performance of different classification techniques/algorithms on a wide variety of classification problem have been published. Although the authors of these articles were able to claim that one classifier is better than the other for the given task, not all of them agree on the dominant performance of one classifier/algorithm over the rest. This is evidence showing that different classifiers are more appropriate for different tasks and one can not absolutely claim that one classifier is globally dominant. The performance of a classifier depends on a number of factors such as the assumptions governing the problem, the design of the classifier, and the dimensionality of the problem. Hence it is hard to claim that one classification technique is generally better than others.

Traditionally performance of a classifier is measured in terms of the accuracy of prediction defined as the percentage classified correctly out of the tested samples. Practically available samples are divided into training and testing set. The performance of the classifier which is trained using the training set is then measured based on the mis-classification rate on the test set. However, this approach has a number of limitations due to the assumptions made about the test data set [Provost *et al.* 1998; Fawcett 2003]. Accuracy or error rate as a measure of performance provides an insight when the available data sets (training and testing sets) are sufficiently large, the training and test sets are independent and the distribution of the patterns across the different pattern classes is well balanced [Jain *et al.* 2000; Ferri *et al.* 2003]. However, these assumptions are very hard to satisfy in most real-world problems where the number of counter-examples are very few relative to the large set of patterns available. This problem even becomes more emphasised in classification problems in the domain of computational biology where data generation is not only costly but also the available data is likely to be highly skewed (see Saitta and Neri [1998] for example). When the number of patterns belonging to one class is proportionally very big relative to the other (say 99:1), a complete mis-classification of patterns belonging to the class with fewer test pattern will not have a significant effect despite a likely overall poor generalisation ability of the classifier on unseen problem. In addition to this, using accuracy as classification performance measure assumes equal misclassification cost or penalty. This assumption limits the insight we have about the classifier, such as the type of error made (sensitivity of the classifier) [Provost *et al.* 1998].

There are a number of alternative approaches to address this bias when evaluating the performance of a classifier and making empirical comparison on classifier performance. One approach is to evaluate the performance of the classifier on each pattern class separately. An alternative approach is a technique originated in signal detection theory which is a graphical approach to

visualise, organise and select classifiers based on their performance called Receiver Operation Characteristics (ROC) [Provost *et al.* 1998; Fawcett 2003]. Spackman [1989] is one of the early adaptors of ROC in machine learning for comparing and evaluating different classification algorithms [Fawcett 2003]. Following Spackman [1989] and other early adopters of this technique, increasing number of researchers are applying ROC to evaluate and compare the performance of machine learning algorithms on classification problems that are hard to satisfy assumptions that makes the use of the traditional approach less appropriate [Fawcett 2003]. In this research area under the ROC curve (AUC) will be used in conjunction with the accuracy to measure and compare the performance of the different classification algorithms used.

Application of the trained classifier will give us information showing the difference between the true and the predicted class for the set of labelled patterns in the test set. For a binary (two class) classification problem the information can be summarised in the four metrics listed below:

True Positive (TP): Number of correctly classified positive patterns

True Negative (TN): Number of correctly classified negative patterns

False Positive (FP): Number of misclassified positive patterns

False Negative (FN): Number of misclassified negative patterns

These four metrics can be effectively represented using contingency table or confusion matrix as shown in Table 4.1. This matrix is a basis for many performance measures that combine these metrics to ease comparison of classifiers.

	Predicted Positive	Predicted Negative	
Positive Examples	TP	FN	Pos (TP + FN)
Negative Examples	FP	TN	Neg (FP + TN)
	PPos (TP + FP)	PNeg (FN + TN)	N

Table 4.1: A contingency table or confusion Matrix

From Table 4.1, the row total “Pos” and “Neg” are actually positive and negative examples, the column total “PPos” and “PNeg” the number of predicted negative and positive patterns respectively. N is the total number of test patterns (N = Pos + Neg = PPos + PNeg). The numbers along the major diagonals of represents correctly classified patterns while the off-diagonals are the confusion between the two classes. Common performance metrics which gives meaningful

measurements can be calculated from the confusion matrix:

$$\text{error rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

$$\text{accuracy } (1 - \text{Error rate}) = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{sensitivity (TP rate)} = \frac{TP}{TP + FN}$$

$$\text{FP rate} = \frac{FP}{FP + TN}$$

$$\text{Specificity } (1 - \text{FP rate}) = \frac{TP}{TP + FP}$$

Some of the above metrics allows us to measure the performance of the classifiers with respect to the individual classes. The two terms which are usually associated with ROC are sensitivity and specificity, also known as recall and precision respectively. For classifiers with continuous-valued output $(-\infty, +\infty)$ (for example SVM^{light}, neural networks with $tanh$ activation function), these terms are subjected to different values based on the threshold value (cut-off value to label a pattern as positive or negative, it its value greater or less than the threshold value respectively) chosen for classifier. In such cases the major diagonals and the off diagonals of the confusion matrix will have a different value based on the chosen threshold value. An ROC curve, a two dimensional graph where FP rate is plotted on the x-axis and TP rate is plotted in the y-axis represents all possible combination of TP rate and FP rates [Fawcett 2003]. An example of ROC curve for three classifiers A, B and C is given in Figure 4.1.

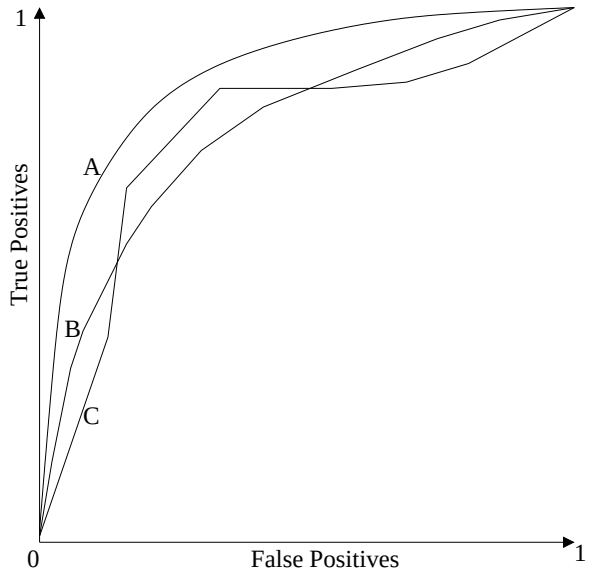

Figure 4.1: An ROC curve for three classifiers A, B and C

The ROC curve in figure 4.1 shows the performance of the three classifiers in terms of the trade off between sensitivity and specificity. An ROC curve which is plotted on an FP rate - TP rate space always passes through the points (0,0) and (1,1). At (0,0) the classifier has classified all the test sets as negative. On the contrary, at (1,1) the classifier has classified all the test sets

as positive. For a random classifier, the ROC curve will be a straight line connecting these two points. For a perfect classifier the ROC curve is a curve connecting the three points ((0,0), (0,1) and (1,1)). From the ROC curves given in Figure 4.1 it can be said that classifier A performs better than a random classifier and both classifiers B and C. However, for any two classifiers where the ROC curve intersects (Classifiers B and C from Figure 4.1) it is not always easy to compare the performance. It can be seen that both classifiers have a better performance than a random classifier but it is hard to say B is better than C and vice versa. Depending on a threshold value chosen (particular sensitivity) one is better than the other.

In such instances where the ROC curve crosses using the area under the ROC curve (AUC) as a single number performance evaluator is an appropriate tool [Bradley 1997]. For a random classifier the AUC equals 0.5 and for a perfect classifier it will be 1.0. Although it is possible for a classifier with higher accuracy to have a lower AUC than one with lesser accuracy, AUC has been voted as a better way of evaluating classifiers performance than accuracy [Ferri *et al.* 2003]. The algorithm used in this research to calculate the points on the ROC curve and the area under the curve (AUC) is adopted from Ferri *et al.* [2003].

4.6 Data and input encoding

4.6.1 Data

Ideally, a large training and test data set for which the phenotypic drug resistance status of the different samples was known would be used. Unfortunately, these are difficult to obtain in sufficient quantity. Instead, a data set that has previously been classified by other researchers is used. While not ideal, we emphasis that our objective is to evaluate machine learning algorithms for future mutations and drugs rather than to discover new biological knowledge for existing drugs and mutations. The data set used has a high-degree of biological fidelity but more important, it contains typical patterns and mutations. Thus, if the model can learn the mutations that others have predicted or spotted, it can conclude that the machine learning technique can be used to learn new patterns for new drugs and mutations.

This research compared the performance of SVMs, neural networks, decision trees and logistic regression method on the HIV-1 sub-type B, common in Europe and Americas. There is no reason to believe that performance of other sub-types would be different. The recognition performance of these algorithms is tested by its ability to classify a given nucleotides sequence as drug resistant or susceptible and their trade-off between sensitivity and specificity, which is measured by the metric area under the ROC curve.

Because of the drugs used in the sample data, the genetic sequences of the viral protease or/and reverse transcriptase were used. The genetic sequences for the protease and reverse transcriptase of the HIV-1 virus used in this experiment are the same set of sequences used by Ravela *et al.* [2003] in their investigation of mutation points responsible for the discordances between different genotypic drug resistance interpretation algorithms. This data set comprises

sequences of isolates from 2045 individuals each constituting position 1–99 of protease and 1–240+ of reverse transcriptase along with the corresponding drug resistance status labelled as **S** for susceptible, **I** for intermediate and **R** for resistant for 15 different retroviral drugs according to the different algorithms used in Ravela *et al.* [2003] (some of the sequences contain positions 1 – 250 for RT). These algorithms had in some cases classified some sequences differently. These discordances are reported to result from several frequently occurring simple mutation patterns and small number of drug resistance mutations in the cases of nucleoside reverse transcriptase inhibitors (NRTIs) and non-nucleoside reverse transcriptase inhibitors (NNRTIs) respectively. However, discordances in protease inhibitors (PIs) are results of a large number of complicated mutation patterns [Ravela *et al.* 2003]. Therefore, to avoid these discordances, this research considered drug resistance status assigned by HIV RT and Protease sequence Database (HIVDB) (`http://hivdb.stanford.edu`). A preliminary experiment has shown no performance difference as a result of drug resistance algorithm selection.

As few of the test data sequences were labelled as intermediate, for this research, intermediate sequences are considered as resistant sequences. Table 4.2 shows the final count of resistant and susceptible sequences for the five selected drugs.

Drug category	Drug	Resistant (R)	Susceptible (S)
PI	IDV	884	1161
PI	NFV	1037	1008
NRTI	AZT	1090	955
NRTI	D4T	1058	987
NNRTI	NVP	1377	668

Table 4.2: Sequence distribution between resistance (R) and susceptible(S)

4.6.2 Input encoding

The first step towards applying the different classification techniques used is pre-processing and filtering of the selected data sets to meet the specific data representation requirement of these algorithms. The sequences from Ravela *et al.* [2003] are composed of the codons positioned 1 – 99 of the protease and 1 – 240+ of reverse transcriptase. Each sequence has a reading frame stating from the first base in the sequence and hence the first ($99 \times 3 = 297$) bases corresponds to protease while the remaining ($240+ \times 3 = 720+$) bases correspond to reverse transcriptase. Therefore the first step was extracting part of each sequence corresponding to the particular drug category under investigation.

Mutations on different parts of the viral genome are responsible for drug resistance behaviour of the virus. Resistance to PIs are caused by exhibited mutations in the viral protease. However, resistance to both NRTIs and NNRTIs are caused due to some mutations in the viral reverse transcriptase. The first 297 nucleotides of the sequences are used for the drugs in the protease inhibitors category. For the other two drug categories, namely NRTI and NNRTI, two

alternative approaches were taken. The first approach taken was considering the entire sequence which is the combination of protease and reverse transcriptase and leaving the task of discriminating those features that are not responsible for the resistance behaviour of the strain for the particular drug for the classification algorithm. In the second approach, only part of the whole sequence which belongs reverse transcriptase was considered. After removing the first 297 nucleotides from each sequence to work with the protease inhibitors, the remaining sequence is part of the original sequence that comprised position 1 – 240+ of the reverse transcriptase. Since most of the mutation points responsible for drug-resistance for drugs in these two categories is the first 240 codon positions of the reverse transcriptase, the remaining sequence after elimination the protease part had all the necessary information needed for this experiment. There is one problem that could possibly arise when removing the first part of the sequence is the question of open reading frames of the reverse transcriptase sequence (the remaining part of the sequence) for those sequences where the protease might be shorter or longer that 297 nucleotides. To address the possibility of such problem a preliminary sequence alignment was carried out and no such problem was identified. Having done this, the second approach has resulted in a sequence of 720 nucleotides long (240 codons). Thus, there is one set of sequence with the corresponding drug resistance label for the protease inhibitors and another two sets for the nucleoside and non-nucleoside reverse transcriptase inhibitors.

Designing a complete classifier includes identification of a proper pre-processing and post-processing techniques beside selecting appropriate classification algorithm and tuning the corresponding parameters. Like most computational biology problems, the first pre-processing step is encoding the nucleotides (character) sequence into an appropriate vector (numeric) representation that can efficiently work across the different classification algorithms used in this research. The choice of the encoding technique affects the quality of information retrieved from the raw data (the nucleotide sequences), and consequently the performance of the classifiers. Therefore, when selecting an encoding technique a number of factors should be taken into consideration. As presented in Wu [1997], when encoding sequences one needs to ask whether to consider fixed or variable length of sequences, if the local or global information more important, if the information has any positional dependency, if the intention is searching for signal or content, etc. Recall that drug resistance is caused by exhibited mutations in a specific position in the viral genome. Furthermore, not all mutations have the same contribution to wards the drug resistance behaviour of the strain. There are major and minor mutation points. To address these underlying facts, data representation which can highlight the global position of the mutation points and their positional dependency with other mutations is of interest is considered. For this purpose the input encoding in this research is as follows.

The sequences in each data set were converted into vectors as follows. Before converting each sequence into an equivalent numeric representation, each sequence was first transformed into a consistent nucleotide representation by substituting those nucleotide codes that represent any two or more nucleotide bases. The conversion of these nucleotide codes into the bases was done according to the IUPAC codes (see Table C.2 in Appendix C). Each set of 3 nucleotides

makes up a *codon*, which encodes for part of the corresponding enzyme. We give each possible codon a numerical value using a standard numbering scheme. The conversion starts by assigning an integer value for each of the nucleotide bases: A, C, U and G were assigned 0, 1, 2 and 3 respectively. With a numeric value assigned to each nucleotide base, the numeric value of the codon is the sum of the integer representation of the each nucleotide base multiplied by 4^i, where i is the position of the nucleotide in the codon. For example, the codon AUG will be $0 \times 4^2 + 2 \times 4^1 + 3 \times 4^0$ = 11.

Using this scheme, each sequence was converted into a vector. The protease sequences are 99-dimensional vectors ($99 \times 3 = 297$) and the reverse transcriptase sequences are 240-dimensional vectors ($240 \times 3 = 720$). The vector representation of each sequence paired with the corresponding drug resistance label makes up the data sets used in this research for training and testing. This representation scheme allows us to detect mutations at which positions caused drug resistance. Note that this particular input encoding in HIV drug resistance prediction task has the advantage that it models both the global and relative position of mutations exhibited.

Given the vector representation of the above selected data sets, data sets were divided into a training and test set for cross-validation. As described in Section 4.4, we randomly selected 75% of the data sets for training and the remaining 25% for testing. With this approach the training set compromises 1500 sequences with the corresponding drug resistance profile for each drug. The testing set contains 545 sequences. The random selection gives equal probability for each sequence and usually guarantees evenly distribution between the Resistant and Susceptible classes relative to the overall distribution over the sample. 10 training and testing sets were created for each drug in table 4.2.

4.6.3 Alternative input encoding

A lot of questions can be asked about the input encoding technique and we will not claim that this is the best input encoding that can be used for this problem. However, we believe that the input encoding used has done the job to answer the research question and had an advantage compared to the other two approaches were tested as a pre-experiment.

The first alternative approach tested was a sight variation of the above mentioned scheme. In this approach, the effect of giving different order of significance to the nucleotides in a codon was investigated. In the above mentioned scheme, the first nucleotides the codon was the most significant bit. In this alternative approach, the last nucleotide (ie. the third nucleotide) was given the highest significance. With this scheme, the codon AUG which was equivalent to 11 in the above mentioned scheme will be equivalent to $0 \times 4^0 + 2 \times 4^1 + 3 \times 4^2$ = 56.

The performance of the different classifiers on the data set with this input encoding scheme was slightly lower than the performance with the above scheme. Furthermore, this input encoding scheme did not produce a different ordering of the classifiers in terms performance. Besides the slight performance advantage of the above scheme, there is a biological explanation that makes the above representation more favourable. As it can be seen from Table C.1 in Appendix C, the most significant nucleotide is the first nucleotide in the codon encoding an amino-acid

and the third nucleotide is the least significant (for example, both GAU and GAC encode the amino-acid Aspartic acid (Asp).

The second alternative approach tested was direct sequence encoding presented in Wu [1997]. In this approach a vector of four units with three zeros and a single one was used for a nucleotide. The four nucleotides was represented as 1000 (A), 0100 (U), 0010 (G) and 0001 (C). With this scheme, the dimensionality of the data is 12 times more than the two schemes presented above. For example, the protease sequence which is 297 nucleotides long will have a new dimensionality of 1188 vector components compared to the 99 dimensional of the above schemes.

The average performace of the different classifiers with this input encoding scheme was slightly better than the scheme selected for this research and presented in the previous section. However, the models were less stable and the standard deviation was bigger. Furthermore, similar to the above scheme, this input encoding did not result in a different ordering of the classifiers in terms of performance. Besides the higher dimensionality and the unstable behaviour of some of the models, interpretation of the models in terms of mutation points was more complicated. For example, with the selected input encoding scheme comparing the split criteria with the mutations points known to confirm drug resistance is easier compared to the bit encoding. Therefore, the above mentioned reasons, the input encoding scheme presented in the previous section was selected.

4.7 Detailed methodology

The experiment reported in this dissertation compares the performance of four classification techniques: SVMs, neural networks, decision trees and logistic regression. The performance (as described in Section 4.5) of the each classification algorithm described in Section 4.3 was tested in terms of the ability to classify the sequences for the different HIV strain described in Section 4.6.1 as susceptible or resistant. As described in the precious section, the input data was pre-processed to make it suitable for the different classification algorithms. These standard input representation was used across the different classification algorithms. However, further pre-processing such as normalisation of the vectors, dimensionality reduction was required for some of the algorithms. These further pre-processing steps are described in this section together with the experimental setup for the different classification algorithms.

For SVMs and neural networks significant testing was done to find the best parameters for this problem. For this phase of the research much focus have been given to neural networks and decision trees because previous work has been done on drug resistance using neural networks and decision trees. A thorough study is not conducted in the case of logistic regression.

4.7.1 Support vector machines

The first set of experiments was done using SVMs. There is a general approach to use SVMs as a classification tool, which can be effectively summarised in the following sequence of steps. Having defined the problem set as a classification problem and prepared the training test where our learning system is going to acquire its knowledge the next step will be kernel selection. Kernel selection is based on the available prior knowledge or previous similar experiments or in the absence of both empirical test on the different kernels from simple to more complicated ones. For the selected kernels, the kernel parameters are then tuned based on the performance of the trained learning system on the kept aside test data. Unseen objects are classified based on the sign of the decision function. In this section we will describe the experimental setup starting from the kernel selection step. As described in the research question and motivation for the research, existing biological knowledge about the problem will not incorporated and hence kernel selection is not based on prior knowledge about the problem. Furthermore, to our knowledge there is no other previous research on drug resistance prediction using SVM as a classification tool. Hence, empirical testing is used to select the kernel and tune the respective parameters.

In this experiment we used a base SVM kernel to normalise the encoded input before starting training and testing with the other kernels. This means the data is transformed using normalisation kernel $k(x.y)$ such that the data is contained in a sphere of unit radius. $k(x, y)$ is defined as:

$$k(x, y) = \frac{x.y}{\sqrt{(x.x)(y.y)}} \tag{4.1}$$

This transformation makes the data suitable for the domain-restricted kernels and simplifies the generalisation and error estimation during optimisation. The data was further transformed using the polynomial or radial basis kernel during the training process.

There are two parameters of interest for each kernel selected, the kernel parameter and the regularisation constant. For the polynomial kernel (see equation 3.25) the parameters of interest is the degree of the polynomial d. In this experiment polynomials of first to third degree ($d = 1, 2$ and 3) are investigated. The parameters of interest for the RBF kernel (see equation 3.26) is the width of the kernel (γ). The values of γ investigated in this research are ($\gamma = 0.1, 0.5, 1, 2, 5$ and 10). Different values for the regularisation constant (C), which indicates the trade-off between the training error and the separating margin, are investigated for each selected kernel parameter. The values of the regularisation constants are ($C = 0, 1, 10, 100$ and the default value). The default value for the regularisation is computed from the training data during the learning process. The selection of the kernel parameters is done starting from a simple dot product kernel to a higher degree polynomials and different RBF kernels. The a couple of values for the degree of the polynomial kernel, the width of the RBF kernel and the regularisation constant are selected with a view that, these values are enough to answer the research question.

4.7.2 Neural networks

The second set of experiments was done using neural networks. The network architecture used in this research is the popular feedforward multilayered perceptrons with fully-connected neurons and backpropagation learning. Neural networks with backpropagation learning has a number of parameters affecting its performance such as, the learning rate, the momentum term and the architecture of the network. The network architecture is characterised by the number of hidden layers, the number of neurons in each layer, their activation function and so on. There are different modifications to the standard backpropagation algorithms each with their advantage in terms of memory usage, convergence speed and training set size. Two of these algorithms are the Levenberg-Marquardt algorithm proposed by Hagan and Menhaj [1994] and Resilient propagation algorithm proposed by Riedmiller and Braun [1993]. The modification to the standard backpropagation algorithm adopted by these algorithm is theoretically best suited to certain problems than other. However, the practical effect is not always in line with the theory. Hence one needs to test these algorithms and determine the best for the problem at hand.

One of the parameters that define the architecture of the network is the activation function. The activation function is responsible for the reaction for each neuron it is attached to as a result of the input. Different activation functions have different input and output range and hence might have different training time. Choosing the appropriate activation function(s) for our problem at hand is another important step of this experiment. A number of preliminary experiments will be carried out to choose the appropriate learning algorithm and activation function before further network parameters are tuned.

The number of neurons in the input layer equals the dimensionality of the data and there is only one output. However, the number of hidden layers and the number of neurons in each layer should march the complexity of the classification problem. As described in Riedmiller and Braun [1993] and numerous previous research, a network with two hidden layers can be tuned to address any complex problem by choosing appropriate number of neurons in each layer. There is a theoretical way of choosing the number of neurons. The number of neurons in the hidden layers should be large enough to address the complexity of the problem but not too large for the network to estimate the corresponding weight from the available training data. In this experiment we will start with small number of neurons and increase the number of neurons to obtain the best network configuration. Furthermore, depending on the learning algorithm, the respective parameters such as the number of epochs, learning rate, momentum term, etc. will be tuned to achieve better performance.

Like most pattern recognition algorithms, the performance of the neural network is affected by the large dimensionality and the representation of the data. As presented in section 4.6.2, the order of magnitude of each attribute ranges from 0 to 63. This might be a source of problem for some activation functions which are defined in a standard universe. The former problem will be addressed by preprocessing the data using dimensionality reduction technique. The dimensionality reduction technique used in this research is principal component analysis (PCA). The dimensionality reduction is carried according to the recommendation by Jolliffe [1986].

The latter is addressed by scaling (normalising) the data. The Neural Networks toolbox and MatLab implementation of PCA is used in this research.

4.7.3 Decision trees

The third set of experiment was evaluating the performance of decision trees. Similar to the previous two experiments, we started with converting the available training and test data into a format suitable for the selected decision tree algorithm, C4.5. This algorithm has a module to build a decision tree based on a given training set and evaluate the resulting tree on a given test set. This module has a number of parameters, some general and others affecting the performance. In this experiment the default value for all the parameters except one is used. The parameter confidence factor (CF) affects decision tree pruning. A small value means heavy pruning. During our experiment we will start with the default value (25%) and decrease the value if the actual error of the pruned tree on the test is higher than the estimated error.

4.7.4 Logistic regression

The final set of experiment is evaluating the performance of logistic regression. In this experiment we used Matlab implementation penalised Logistic regression with Ridge estimator proposed by Zhu and Hastie [2004]. The recommended valued were used for all the parameters and the range of the possible values for the regularisation parameter was set between 0.001 and 1000 in step of 0.1 following recommendations from previous research.

4.8 Summary

This work answered the question “can SVMs predict HIV drug resistance based on the genetic sequence of the viral protease or reverse transcriptase”. To answer this question we will be using a comparative approach and compare the performance of SVM to other popular classification algorithms. The performance evaluation criteria used in this work is the accuracy of classification and the area under the ROC curve. Previous researchers have argued that area under the ROC curve is a better single value performance indicator hence more emphasis was given for this value.

The performance of the different classification algorithms will be evaluated using cross-validation testing where randomly 75% of the data was used as training and the remaining 25% for testing. This procedure was repeated 10 times on different training-testing data sets and the average performance is considered.

The results of these experiments is be presented in the next chapter

Chapter 5

Results and Discussion

5.1 Introduction

Model selection can be carried out by analysing the theoretical bound on generalisation or by performing empirical experiments and analysing the performance of the classifier based on a given unbiased test sets. In this work the model selection is carried out using cross-validation technique. As discussed in section 4.5, the performance of the different classification algorithms was evaluated in terms of the accuracy of classification and the area under the ROC curve (AUC) based on cross-validation results as outlined in the previous chapter. In this procedure 75% of the data randomly selected were used for the training. The accuracy and the AUC for the different classification algorithms was then tested on remaining 25% of the data. The results presented in this chapter are the mean accuracy and AUC over 10 trials. Each of the 10 trials used different pairs of training and testing sets.

As described in the previous chapter, the main objective of this research is evaluating the performance of SVMs in comparison to some of the other popular pattern recognition techniques. Hence the conducted research was aimed at achieving this by designing prediction models for neural networks, decision trees and logistic regression and comparing them with the designed SVMs model.

The remainder of this chapter is organised as follows. In the next section, the performance of SVMs as a function of kernel parameters and the regularisation constant will be discussed. This section also gives general remark on SVMs and tries to relate some of our results with the theory. In section 5.3 the performance of neural networks models is presented. Different architectures and learning algorithms were investigated and this section presents the results for the best configuration found. Moreover, the effect of dimensionality reduction is investigated. Section 5.4 presented the performance of decision trees models followed by section 5.5 which presents the performance of logistic regression models. The performance comparison of the different classification algorithms and the finding of this experiment is discusses in section 5.6. Section 5.7 gives some remark on statistical significance test. Section 5.8 discusses the limitations of the research and hints ways of addressing these limitations. Finally, the chapter is summarised in section 5.9.

The results for the different set of experiments are presented in following tables.

Description	Table No	Page No
SVM Polynomial kernel	Table 5.2	Page 95
SVM RBF Kernel	Table 5.5	Page 102
	Table 5.3	Page 97
	Table 5.4	Page 97
Neural networks	Table 5.6	Page 104
Decision trees	Table 5.7	Page 105
Logistic regression	Table 5.8	Page 108
Selected Results	Table 5.9	Page 108

Table 5.1: List to result tables

5.2 Performance of SVMs

The success around SVMs is highly attributed to the kernel trick which simplifies highly complex real life problems by projecting the patterns into a higher dimensional feature space where it can be solved with ease. Hence kernel selection is the critical part of SVM approach. As presented in section 3.4 the three commonly used kernels are the polynomial, RBF and sigmoid kernels. Osuna *et al.* [1997] has pointed out that, with these kernels SVMs can emulate different previously well studied classifiers. The implementation of SVMs used in this research, SVM^{light}, does not incorporate automatic selection criteria for the kernels or their corresponding parameters. Hence, the task of selecting kernel(s) and tuning their parameters are left for the expert designing the classifier.

Kernel selection is motivated by the information at your disposal. When enough prior information is available about the problem, one can choose the best kernel that well suits the problem or modify existing kernels by incorporating these information or design a specific kernel for the problem. However in the absence of such information we are forced to make our kernel choice based on previous research on similar domain or empirically test the available kernels. In this research empirical testing using cross-validation is used.

The polynomial kernel is one of the most commonly used kernels. Depending on the degree of the polynomial, the shape of the resulting boundary becomes more complicated. Hence by tuning the degree of the polynomial either based on the estimated VC-dimension or by assessing the generalisation error (for example cross-validation), the polynomial kernel can be tuned to adopt the required decision boundaries. The RBF kernel is the most popular kernel because of its capacity to generate a decision boundaries that can accommodate a wide range of classification problems. The RBF kernel is best suited for a classification problem where one class is enclosed by the other class. It can also classify simple linearly separable case, which according to Keerthi and Lin [2003] can be considered a special case of RBF. For an SVM classifier with RBF kernel the support vectors are the centres of the RBF and γ, which is the width of the RBF, determines the area which is influenced by these support vectors [Vapnik

1995]. The Sigmoid kernel which emulates the multilayered perceptron is a tricky one to use since it satisfies the Mercer condition only for some values of the gain and the threshold [Vapnik 1995]. For this reason and its comparative poor performance compared to the other two kernels, the sigmoid kernel is less recommended by many researchers [Lin and Lin 2005]. However, it is used for historical reasons. The Sigmoid kernel is not used for this research.

The model selection was done by tuning the kernel parameters for the polynomial and the RBF kernels. With the polynomial kernel we have tested different values for the degree of the polynomial. Similarly, for the RBF kernel model, various values for the width of the kernel were tested. For each of the polynomial or RBF kernel, we further tuned the models by selecting a different value for the regularisation constant.

The tuning of these parameters was carried out using grid-search using cross-validation techniques over the available samples. As pointed out in the previous chapter we used 75% of the data for training and the remaining 25% for test. The patterns for the training and testing sets were selected randomly. These steps were repeated 10 times and the results presented are the mean value over the 10 experiments.

5.2.1 Effect of polynomial kernel parameter on classification

The polynomial kernel (equation 3.25) has two free parameters that need to be tuned according to the complexity of the problem: the degree of the polynomial d and the regularisation constant C. While the degree of the polynomial controls the complexity of the classifier or shape of decision boundaries, the regularisation constant controls the intensity or sharpness of these boundaries. In other words the regularisation constant counts for the trade-off between the width of margin of the classifier and the misclassification penalty. Therefore, model selection for this kernel is determining a good value for both d and C. To do this we have used grid-search using cross validation technique for d = 1,2 or 3 and C in {default value, 1, 10, 100}. The default value of the regularisation constant is calculated from the training set during the training steps and equals

$$\frac{N}{\sum_{i=1}^{N} K(x_i.x_i)} \tag{5.1}$$

where $K(x, y)$ is the chosen kernel (in this case the polynomial kernel)and N is the size of the training set. The results for the performance of the SVMs classifier with polynomial kernel are presented in Table 5.2 on page 95.

As it can be seen from the table the performance of classifier has increased as we increase the degree of the polynomial except for the case that the default value is used for the regularisation constant. However, the performance gain was not continuous for all of the patterns and there are instances where the performance has deteriorated. For NVP(PR) and NVP(PR+RT) when $C = 1$ and for NFV when $C = 100$ there was no improvement gained as we increased the degree of the polynomial from two to three (see Figure 5.1 on page 94). This observation is in line with the theory, that for a given classification problem, if effectively classified by

a polynomial of lower degree, polynomials of higher degree are likely to generate identical decision boundary. For example, if a classification problem is linearly separable, one can expect the same classification boundary from a linear and quadratic and higher degree polynomial kernel SVMs.

SVM^{light} is designed to optimise the generalisation in terms of accuracy of classification. However, as mentioned above in this research we are giving more value to the AUC as a performance measure. As you can see from the table there are instances where the accuracy of classification has increased, however the AUC remained constant or decreases (for example NVP(RT) and NVP(PR + RT) for $C = 100$). Hence when talking about performance gain as a result of parameter tuning, both AUC and accuracy were considered.

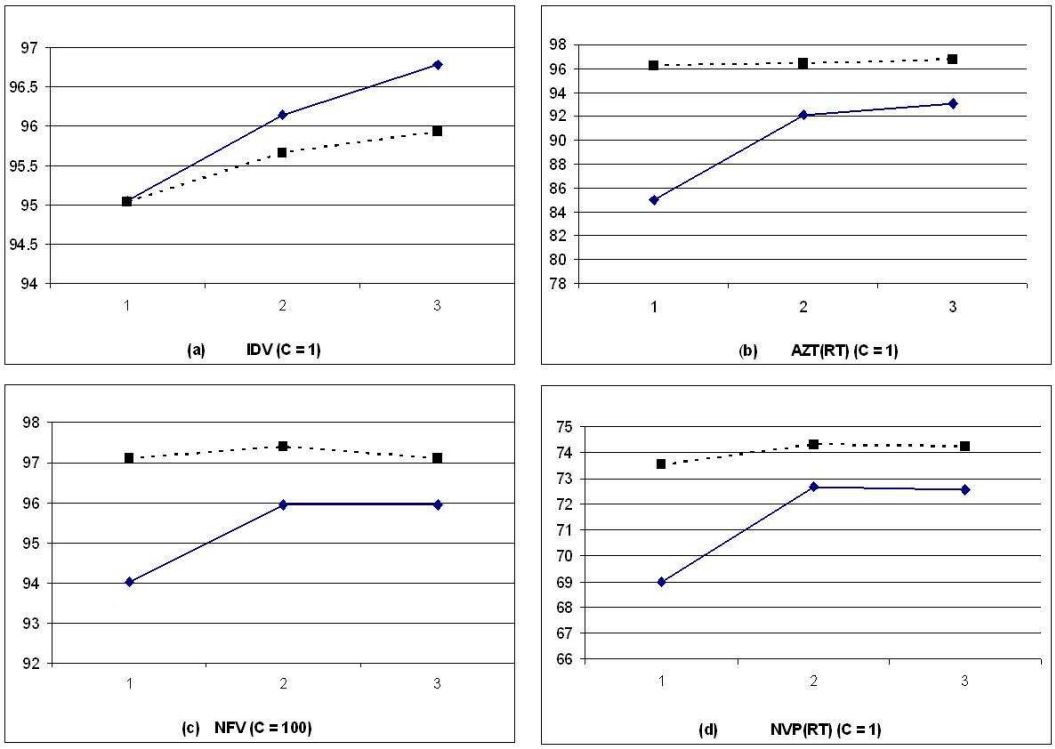

Figure 5.1: Selected graphs (degree (d) vs Accuracy/AUC) showing the behaviour of SVMs as a function of the polynomial kernel degree d. The solid lines in the graph show the accuracy of the model, the broken lines show the AUC. These graphs are intended to show how the accuracy of classification and the AUC are related as a function of the degree of the polynomial.

Similar to the degree of the polynomial, the SVM classifier has shown performance gain as we increase the value of the regularisation constant keeping the degree of the polynomial fixed. As can be seen from the table (for instance NFV) the performance increased from 92.66% accuracy and 96.96% AUC to 94.31% accuracy and 97.10% AUC as we increase the regularisation constant from the default value to 10. However, there was no more performance gain as we further increased the value of regularisation constant for the given polynomial degree. Although the results presented in the table do not exhibit this behaviour consistently, it is expected to happen as we further increase the regularisation constant for all patterns for a selected degree. This is due to the fact that, once we reach a certain value for the regularisation constant that best

	degree (d) = 1			degree (d)= 2			degree (d)= 3		
Drug	Accu.	AUC	No. SV	Accu.	AUC	No. SV	Accu.	AUC	No SV
Regularisation constant = default									
IDV	95.05	94.96	906	91.19	94.43	1335	95.05	94.97	905
NFV	92.66	96.96	956	79.08	95.67	1376	92.66	96.97	955
AZT (RT)	84.59	96.12	950	64.77	93.86	1486	84.59	96.12	949
AZT (PR + RT)	81.83	96.05	1032	63.12	93.44	1496	81.83	96.04	1030
D4T (RT)	85.14	88.41	971	69.36	86.69	1486	85.14	88.41	969
D4T (PR + RT)	84.22	96.70	1038	68.62	94.85	1496	84.22	96.70	1038
NVP (RT)	68.99	73.53	1020	68.99	73.55	1026	68.99	73.54	1022
NVP (PR + RT)	68.99	77.02	1021	68.99	76.95	1027	68.99	76.98	1019
Regularisation constant = 1									
IDV	95.05	95.03	879	96.15	95.66	526	96.78	95.93	880
NFV	92.66	97.02	931	93.39	97.10	523	93.66	97.02	928
AZT (RT)	84.95	96.23	907	92.11	96.43	555	93.01	96.72	907
AZT (PR + RT)	82.94	96.16	987	92.29	96.37	636	92.29	96.16	986
D4T (RT)	85.14	88.45	929	92.29	89.38	573	92.29	88.45	929
D4T (PR + RT)	84.22	96.86	989	92.66	96.83	650	92.73	96.86	989
NVP (RT)	68.99	73.51	1020	72.66	74.31	1015	72.56	74.21	1022
NVP (PR + RT)	68.99	77.01	1022	72.11	77.07	1008	72.11	77.02	1020
Regularisation constant = 10									
IDV	94.27	95.96	388	95.23	95.42	289	95.96	94.26	385
NFV	94.31	97.10	355	94.68	97.06	239	94.92	96.85	353
AZT (RT)	92.29	96.80	373	92.48	96.73	241	92.87	96.80	371
AZT (PR + RT)	92.29	96.50	440	92.48	96.67	276	93.01	96.50	440
D4T (RT)	93.02	96.69	489	94.13	96.61	333	94.13	96.68	491
D4T (PR + RT)	92.00	89.46	426	93.94	89.03	283	94.13	89.46	422
NVP (RT)	74.86	76.78	979	79.45	78.93	912	80.25	79.54	979
NVP (PR + RT)	74.68	77.74	970	82.02	79.01	887	82.23	79.55	970
Regularisation constant = 100									
IDV	95.73	95.78	258	95.78	95.73	246	95.78	95.73	258
NFV	94.02	97.10	203	95.96	97.40	168	95.96	97.11	205
AZT (RT)	92.84	97.06	208	93.76	96.84	174	94.23	97.06	209
AZT (PR + RT)	92.48	97.13	233	93.76	96.85	208	94.17	97.44	233
D4T (RT)	94.13	88.79	240	94.86	89.33	220	95.05	88.79	240
D4T (PR + RT)	93.97	97.32	278	94.13	97.11	255	94.31	96.94	278
NVP (RT)	82.57	80.12	844	83.49	91.36	816	83.76	80.11	845
NVP (PR + RT)	83.49	80.16	814	83.41	81.25	766	83.54	80.17	816

Table 5.2: SVMs classifier performance with polynomial kernel

address the trade-off, further increase will either have no or negative effect on the performance of the classifier.

Another result that needs to be discussed is the number of support vectors. As you can see from the table, the number of support vectors in most of the cases less than 50% of the total number of training patterns. As we have mentioned in section 2.5.2 one of the major problem that compromising the performance of the traditional classification techniques is the curse of dimensionality. For a polynomial kernel of degree d, the dimension of the feature space will be at least $\binom{n+d-1}{n}$ [Cristianini and Shawe-Taylor 2000]. However, the ability of SVMs to implicitly perform this higher dimensional classification task in the input space allows the classifier to construct hyperplanes in this high-dimensional space without suffering the curse of dimensionality and/or over-fitting (see Section 3.4). As you can also see from the table the increase in the number of support vectors compare to the increase in the dimension of the feature space (or increase the degree of the polynomial for a fixed value of C), is very low and sometimes it even decreased.

Finally, one interesting aspect of support vectors that need to be discussed is the overlap between the support vectors that shape up different hyperplanes. Based on the reported findings by both Vapnik [1995] and Schölkopf [1997] on the commonality of support vectors among different classification boundaries on the same problem, we tried to see this behaviour in this research. Although numerical results are not presented, commonality on a number of the support vectors for different classification hyperplanes for a particular drug with different polynomial degree and regularisation constant was also witnessed in this reseach.

5.2.2 Effect of RBF Kernel Parameter on classification

The Radial Basis kernel (equation 3.26) is the most popular kernel for practical applications since it is complex enough to address complex non-linear classification problems by transforming the data into a higher dimensional space and simple enough to solve linear classification problems, which is considered as a special case [Keerthi and Lin 2003]. The classical approach of estimating RBF classifiers involves finding the RBF centre using k-means clustering mechanisms and then estimating the corresponding weight for each cluster using error backpropagation. However, the SVM approach has a more elegant way of computing the centre, weight and the threshold that results in best generalisation automatically [Schölkopf *et al.* 1996].

There are two parameters of interest for the RBF kernel: the width of the kernel γ and the regularisation constant C. Like the polynomial kernel our goal is to find a good value for both γ and C. These pair of values are also determined using grid-search using cross validation technique. The complete result from these experiments is presented in Table 5.5 on page 102.

This kernel is most favourable when one class is totally encircled by the other as shown in the Figure 3.9(b). However, as presented in Keerthi and Lin [2003] by varying the value of either of these parameters, fixing the other one, we can achieve almost any kind of decision boundaries including linear hyperplanes. The value of γ is related to the diameter of the enclosing boundaries, the smaller γ gets, the tighter the closed boundaries (circles). In other words,

the smaller the value of γ gets the smoother and more regular the decision boundaries. And for a regularisation constant C which is a function of γ, and by keeping γ close to zero, we can have a linear hyperplane.

To aid our discussion selected results from Table 5.5 on page 102 for three drugs: one from each drug category are presented in Table 5.3 on page 97 and Table 5.4 on page 97. Similar to the polynomial kernel, there was performance gain as a function of γ and C. As it can be seen from the Table 5.3, (for example NVP(RT)) there was a performance gain as we gradually increase γ from 0.1 to 1. However, as we further increase γ to 10 the performance deteriorates (see Figure 5.2). It can also be seen from this table and Table 5.5 on page 102 this is not an isolated incidence.

γ	**IDV** $C = 1$			**D4T(RT)** $C = 100$			**NVP(RT)** $C = 100$		
	Accu.	AUC	No. SV	Accu.	AUC	No. SV	Accu.	AUC	No SV
0.1	89.72	94.04	1418	93.94	96.22	328	93.94	78.34	328
0.5	95.05	95.03	892	93.94	96.02	254	93.93	80.02	256
1	95.78	95.37	694	94.50	96.96	239	**94.50**	**81.26**	239
2	96.15	94.92	542	**94.86**	**97.17**	259	94.86	81.17	259
5	96.33	95.25	426	93.76	97.24	288	93.76	78.23	288
10	**96.15**	**95.63**	407	93.03	96.94	325	93.03	76.52	325

Table 5.3: Selected results for SVM with RBF kernel showing the effect of γ on the classifier performance

C	**IDV** $\gamma = 10$			**D4T(RT)** $\gamma = 2$			**NVP(RT)** $\gamma = 1$		
	Accu.	AUC	No. SV	Accu.	AUC	No. SV	Accu.	AUC	No SV
default	96.33	95.52	473	92.66	96.61	725	91.01	77.63	816
1	**96.15**	**95.63**	407	92.29	96.55	593	92.29	77.52	752
10	95.05	96.97	335	93.94	96.35	309	93.94	78.55	343
100	93.76	96.76	333	**94.86**	**97.17**	259	**94.50**	**81.26**	239

Table 5.4: Selected results for SVM with RBF kernel showing the effect of C on the classifier performance

Similar to the width of the kernel, performance gain was also recorded as a function of the regularisation constant. It can be seen from Table 5.4 for the drug IDV, as we increased the value of C from the default value (usually in the range $(0, 1]$) to 1, the performance increased from 94.33% accuracy and 95.52% AUC to 96.15% accuracy and 95.63% AUC. However, there was also a saturation point where the performance started to decline (see Figure 5.3 on page 98). As described in the previous section, this is also an indication that, some value of the regularisation constant around $C = 1$ has already accounted for the penalty for misclassification. The same result can also be seen from the Table 5.5. The result from Table 5.4 for the drugs D4T and NVP shows that the performance increased gradually as we increased the value of the regularisation

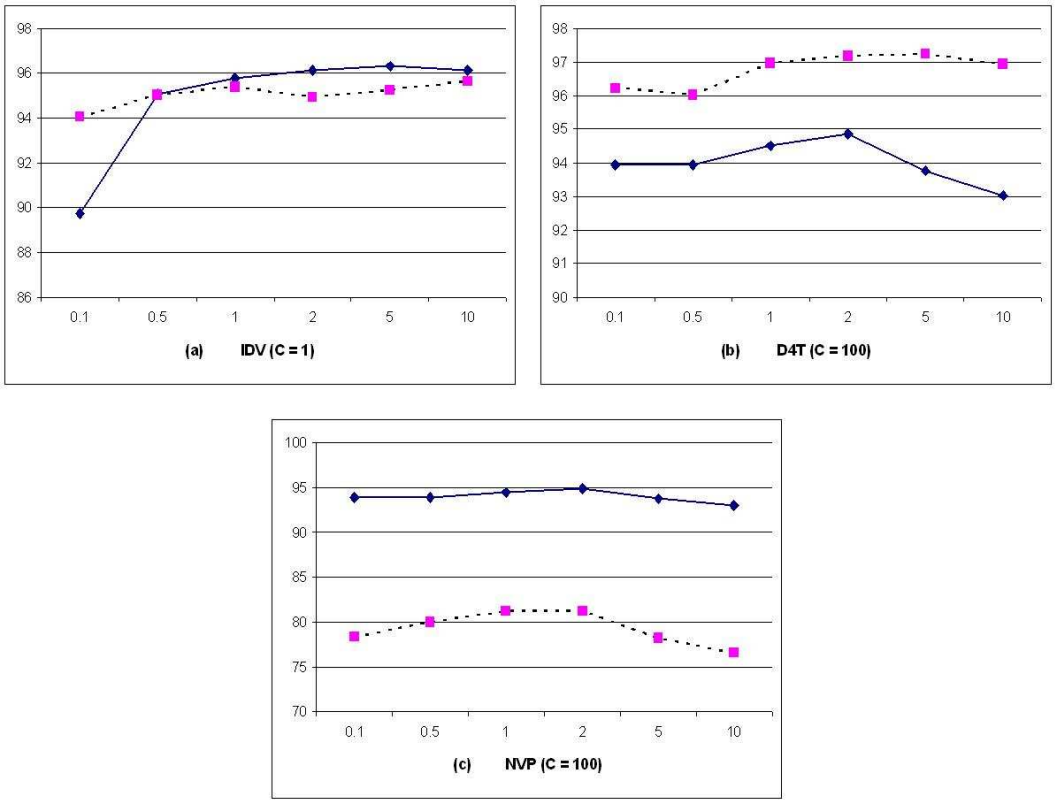

Figure 5.2: γ vs Accuracy/AUC graph for SVM with RBF kernel. The solid lines in the graph show the accuracy of the model, the broken lines show the AUC.

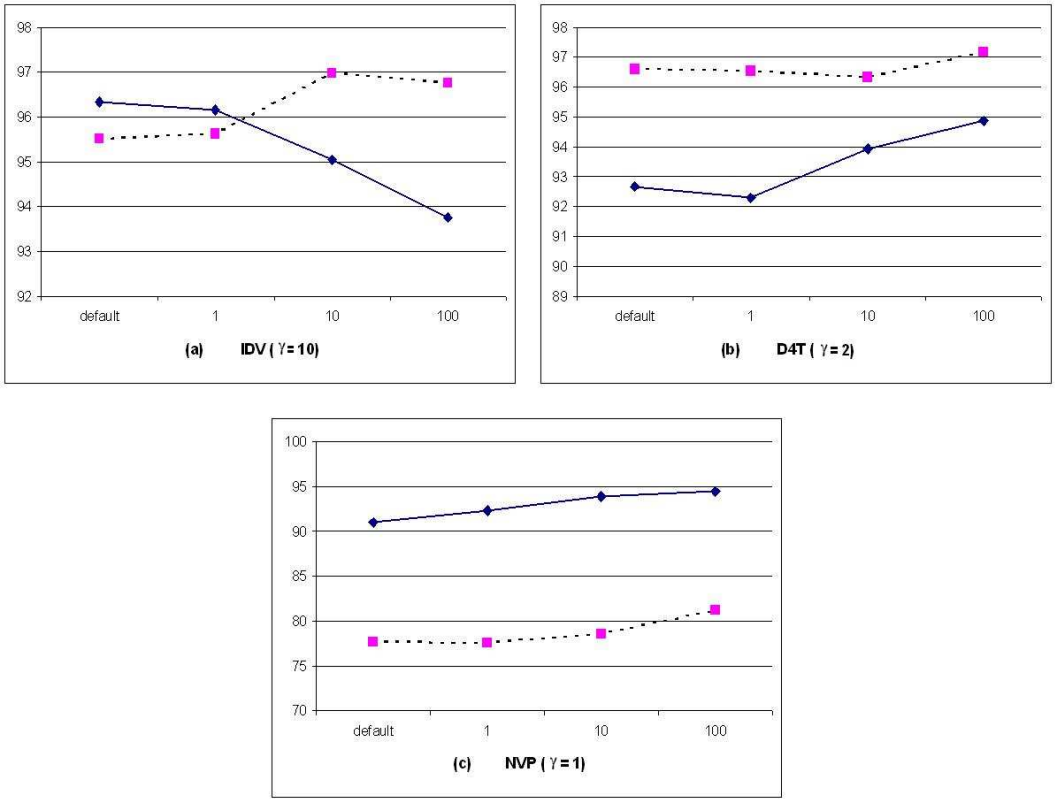

Figure 5.3: C vs Accuracy/AUC graph for SVM with RBF kernel.The solid lines in the graph show the accuracy of the model, the broken lines show the AUC.

constant. However, as we further increase the value of C we are expecting the performance to deteriorate or show no gain after some value.

Finally, similar to the result for the polynomial kernel, the number of support vectors is not as high as the dimensionality of the classification problem. The dimensionality of the feature space for a Gaussian kernel is ∞ [Cristianini and Shawe-Taylor 2000]. As we can see from the tables (Table 5.3 and Table 5.4) the number of support vectors are once again less than 50% of the sample size. Commonality of support vectors among the different models was also exhibited.

5.2.3 Remark on SVMs performance

In this research the performance of the SVMs classifier is assessed using the polynomial and the RBF kernel. As we have discussed in the previous two section, the performance of the classifier range from 68.99% Accuracy and 73.51% AUC for NVP to 96.78% accuracy and 95.93% AUC for IDV for the polynomial kernel. The performance also ranges from 68.99% accuracy and 73.55% AUC for NVP to 96.15% accuracy and 95.63% AUC for IDV for the RBF kernel. These performances were recorded using grid-search on range of values of the degree of the polynomial and/or the width of the kernel and regularisation constant. The performance for these models might be increased by doing extensive search around the point where the best reported performance was recorded. Furthermore, the recorded performances are achieved without incorporating any of the prior biological information about the mutation points and the resistance they are known to confer to the input encoding or kernel designing process. By incorporating these information, further performance gain might be recorded.

As mentioned previously, SVM^{light} is designed to optimise the accuracy of the classifier but as you can see from the graphs in the previous section (Figure 5.1, Figure 5.2 and Figure 5.3) optimising the accuracy usually leads to optimised AUC. This observation was also published previously by Rakotomamonjy [2004]. Consider the highlighted results from the table 5.4 for D4T and NVP have the accuracy 94.86% and 94.50% respectively. The AUC has a bigger variance and equals 97.17% and 81.26% for these drugs respectively. These shows that the AUC variance is higher than the variance in accuracy when the data is skewed. This is in agreement with the fact that AUC is more sensitive towards skewed datasets. The same result was also reported by Rakotomamonjy [2004].

Dimensionality of the patterns under investigation is one of the major issues that compromising the performance of traditional pattern recognition techniques. As you can see from the table we have a pair of data sets for each of the drugs in the nucleoside and non-nucleoside reverse transcriptase inhibitors category. The difference in the dimension between each pair of data is 99 attributes (the 99 codons belonging to the protease part of the sequence). This dimensionality difference together with the fact that SVMs constructs classification hyperplanes in a high dimensional feature space, one can expect a huge performance difference of a model on these two datasets. As described in section 2.5.2 to avoid curse of dimensionality and maintain the performance, the number of training sample was supposed to be increased to match the di-

mensionality difference. The reported result in the previous sections has also showed no much performance loss as a result of dimensionality. These shows the capacity of SVMs to handle high dimensional classification tasks without the need for comparably large data sets.

Drug	c =default			c = 1			c = 10			c = 100		
	Accuracy	AUC	No. SV	Accuracy	AUC	No. SV	Accuracy	AUC	No. SV	Accuracy	AUC	No. SV
γ **= 0.1**												
IDV	95.05	95.06	870	89.72	94.04	1418	95.78	95.39	683	95.60	94.61	331
NFV	92.66	97.45	912	74.50	95.12	1446	92.84	97.09	698	94.50	96.81	282
AZT (RT)	85.14	96.06	898	66.24	94.12	1452	92.11	96.46	732	92.11	96.50	284
AZT(PR + RT)	83.67	96.03	978	64.04	93.37	1494	88.99	96.24	802	92.11	96.55	342
D4T (RT)	85.14	96.25	914	70.83	94.69	1450	92.29	96.34	738	93.94	96.22	328
D4T(PR + RT)	84.95	96.68	980	68.62	94.80	1490	87.89	96.82	805	93.94	96.94	396
NVP (RT)	85.14	77.03	914	70.83	77.23	1450	92.29	77.06	738	93.94	78.34	328
NVP(PR + RT)	68.99	73.55	1023	68.99	73.41	1024	70.09	73.56	1024			
γ **= 0.5**												
IDV	95.23	95.3	806	95.05	95.03	892	95.96	94.35	398	95.60	95.74	266
NFV	92.84	97.63	843	92.66	97.03	933	94.31	96.99	366	95.96	97.37	212
AZT (RT)	87.34	96.31	863	85.14	96.15	914	92.11	96.65	371	92.66	96.93	207
AZT(PR + RT)	85.69	96.15	926	83.12	96.10	911	92.11	96.41	455	92.66	97.02	243
D4T (RT)	85.5	96.45	867	85.14	96.42	933	93.94	96.46	426	93.94	96.02	254
D4T(PR + RT)	86.24	97.23	937	84.59	97.27	1000	93.21	96.55	497	93.94	97.34	286
NVP (RT)	85.5	77.25	867	85.14	77.37	933	93.94	77.76	426	93.94	80.02	254
NVP(PR + RT)	69.36	73.59	1023	68.99	73.52	1019	74.86	76.8	979	82.02	79.97	876
γ **= 1**												
IDV	95.23	95.46	759	95.78	95.37	694	95.78	94.85	340	95.23	95.90	260
NFV	92.84	97.54	769	93.03	97.52	701	94.50	97.14	290	96.15	97.49	203
AZT (RT)	91.93	96.44	809	92.11	96.56	745	92.11	96.58	289	93.39	97.21	193
AZT(PR + RT)	86.79	96.32	880	88.62	96.35	812	92.11	96.58	353	92.66	97.10	226
D4T (RT)	91.01	96.48	816	92.29	96.54	752	93.94	96.31	343	94.50	96.96	239
D4T(PR + RT)	86.97	97.39	881	88.26	97.37	819	93.94	96.62	401	94.50	97.18	288
NVP (RT)	91.01	77.63	816	92.29	77.52	752	93.94	78.55	343	94.50	81.26	239
NVP(PR + RT)	70.09	73.56	1016	70.09	73.65	1017	76.88	78.38	958	82.20	80.16	869

Drug	c =default			c = 1			c = 10			c = 100		
	Accuracy	AUC	No. SV	Accuracy	AUC	No. SV	Accuracy	AUC	No. SV	Accuracy	AUC	No. SV
γ **= 2**												
IDV	95.78	95.17	664	96.15	94.92	542	95.78	95.4	310	94.13	96.65	268
NFV	93.03	97.07	668	93.39	97.12	531	94.86	97.43	258	95.96	97.75	200
AZT (RT)	92.29	96.55	715	92.11	96.56	575	92.29	96.71	263	93.76	97.46	192
AZT(PR + RT)	89.36	96.36	791	91.93	96.43	658	92.29	96.72	309	93.21	96.71	236
D4T (RT)	92.66	96.61	725	92.29	96.55	593	93.94	96.35	309	94.86	97.17	259
D4T(PR + RT)	89.17	97.34	791	92.48	96.86	672	93.94	96.61	372	93.58	97.11	303
NVP (RT)	92.66	78.05	725	92.29	77.99	593	93.94	80.02	309	94.86	81.17	259
NVP(PR + RT)	70.09	73.74	1020	72.66	74.15	1021	79.27	79.09	935	81.83	79.41	886
γ **= 5**												
IDV	96.15	95.28	252	96.33	95.25	426	95.05	96.43	312	94.13	96.84	291
NFV	93.39	97.11	508	94.13	97.54	397	95.96	97.93	251	95.41	98.14	227
AZT (RT)	92.29	96.53	543	92.11	96.55	408	92.66	97.07	246	94.13	96.64	230
AZT(PR + RT)	92.11	96.48	652	92.11	96.51	513	92.66	97	303	93.21	97.16	297
D4T (RT)	93.03	96.64	565	93.94	96.50	460	94.13	96.85	291	93.76	97.24	288
D4T(PR + RT)	92.66	97.40	658	93.21	96.76	538	94.31	97.47	351	92.11	96.81	353
NVP (RT)	93.03	78.11	565	93.94	78.80	460	94.13	80.84	291	93.76	78.23	288
NVP(PR + RT)	73.39	74.72	1032	73.94	76.53	1016	81.10	78.83	937	77.43	74.96	928
γ **= 10**												
IDV	96.33	95.52	473	96.15	95.63	407	95.05	96.97	335	93.76	96.76	333
NFV	94.5	97.12	434	94.86	97.79	369	95.78	98.2	274	95.78	98.24	249
AZT (RT)	92.29	96.54	444	92.11	96.56	356	93.21	97.38	259	94.13	96.18	274
AZT(PR + RT)	92.29	96.49	565	92.11	96.45	469	92.84	96.52	344	93.39	97.2	347
D4T (RT)	93.76	96.60	485	93.94	96.54	417	94.68	97.16	324	93.03	96.94	325
D4T(PR + RT)	93.39	97.39	590	94.13	97.18	504	94.31	97.19	386	92.29	96.68	405
NVP (RT)	93.76	78.92	485	93.94	79.73	417	94.68	80.45	324	93.03	76.52	325
NVP(PR + RT)	73.94	75.76	1034	76.15	77.25	1012	79.63	77.37	989	74.13	73.15	971

Table 5.5: SVMs classifier performance with Radial Basis Function (RBF) kernel

5.3 Performance of neural networks

The performance of a neural network is highly affected by its architecture. Some of the factors defining the architecture and hence affecting the performance of a neural network model are: the connection type, the learning or training algorithms, the number of hidden layers and the number of neurons in them, the activation functions, learning rate, initial values of the weight and the training stopping criteria. Although we did not perform exhaustive search for all of these factors affecting the performance, in this experiment we have tried different network architectures with different numbers of neurons in the hidden layer, activation function and learning algorithm. The architecture were chosen using empirical testing as there is no automatic way of selecting a network architecture for a given problem.

In search of a simple network architecture which is complex enough networks with one hidden (input layer, one hidden layer and output layer) and two hidden layer architectures were tested. There were no performance difference between these two architectures and hence an architecture with one hidden layer network was used as it convergence faster. The number of neurons in the input layers equals the dimensionality of the data and the network has one outout (the output layer has one neuron). Different number of neurons in the hidden layer were tested starting with a simple network with 5 neurons. The number of neurons were increased by 5 until no performance gain was recorded and computational time increased highly (sometimes longer than an hour). For most of the drugs, architecture with 10 – 15 neurons were complex enough.

For each architecture, different learning algorithms were also tested. The three learning algorithms tested are standard backpropagation, Resilient backpropagation and the Levenberg-Marquardt algorithm. Each of these learning algorithms have their theoretical pros and cons. However, empirically testing is required to see their perform on the problem at hand. For this experiment, the Levenberg-Marquardt algorithm was selected. This algorithm was the best in terms of performance, convergence and memory usage. Similarly different activation functions were tested and log-Sigmoid transfer function was selected. Finally, feedforward neural networks with fully connected neurons and log-sigmoid activation function were used.

The performance of these architectures were evaluated on the same data sets with the identical input encoding technique with the SVMs model. However, the data was further preprocessed to make it more suitable for neural network model. The data was normalised in a such away that it will have zero mean and unit standard deviation. The data was then divided into three sets, where 50% is used for training, 25% for validation and the remaining 25% for testing. Each experiment was repeated 10 times and the mean accuracy and AUC is presented in Table 5.6 on page 104.

As it can be seen from Table 5.6 the performance of the network has reduced as the dimensionality of the data increased. The mean accuracy of the network on the protease inhibitors is between 89% and 91%. However, the performance is reduced to the range 82% to 84% for the AZT and D4T as a result of increase in the dimensionality of the patterns. The performance was

	Original dimensionality		Reduced Dimensionality	
Drug	Accuracy	AUC	Accuracy	AUC
IDV	89.75	91.76	89.91	86.76
NFV	91.74	95.31	78.90	87.51
AZT (RT)	82.57	87.79	82.20	88.57
AZT(PR + RT)	82.02	87.70	59.63	60.94
D4T (RT)	84.34	91.85	84.22	92.25
D4T(PR + RT)	82.33	87.09	60.37	54.80
NVP (RT)	69.72	68.93	70.49	69.50
NVP(PR + RT)	64.77	71.10	68.99	47.06

Table 5.6: Performance of neural networks model

further reduced to the range 64% to 69% for NVP due to the higher dimensionality of the data and its relatively skewness. Similar to SVMs algorithm, the Levenberg-Marquardt algorithm is accuracy optimised, however it also optimises AUC in most of the cases.

To address the performance loss as a result of dimensionality increase, we performed one more pre-processing technique to reduce the dimensionality of the data. For these task we used principal component analysis. As recommended by Jolliffe [1986] we reduced the dimensionality of the data up to 90%. The data was then reduced to 2 or 3 dimensional patterns. As we have done in the experiment with the original dimensionality, the data was divided into training, validation and test sets and the mean accuracy and AUC over 10 repeats of the each experiment is given in Table 5.6.

As it can be seen from the table there is no performance gain as a result of dimensionality reduction except for NVP(RT). The reason for these reduced performance could be the information loss due to dimensionality reduction. Remember that the data have discrete representation with each numeric value representing one of the 20 amino acids. Hence, dimensionality reduction we might have caused some valuable information to be lost. From the performance on AZT(PR+RT), D4T(PR + RT) and NVP(PR + RT) it can be seen that the information loss was even higher for these data sets.

5.4 Performance of decision trees

In this part of the experiment we generated decision tree models that describes the drug resistance profile of an HIV strain in terms of the numeric composition of the vector, which in turn represents the different amino acids composition of the enzyme targeted by the drug. To estimate the prediction power of the decision tree model we performed 10 independent tests, each time we used 75% of the data for training and the remaining 25% for testing. The result for the performance of this model is given in Table 5.7 on page 105. The result gives a mean number of interior nodes of the tree, number of leaves of the tree, accuracy of prediction and AUC over the 10 experiments.

Each training session resulted in one decision tree for each drug. Most of the resulting

Drug	No. Interior vertex	No. Leaves	Accuracy	AUC
IDV	16	17	97.06	92.70
NFV	6	7	97.06	94.43
AZT (RT)	14	15	96.33	93.41
AZT (PR + RT)	18	19	94.86	90.96
D4T (RT)	29	30	93.58	88.29
D4T (PR + RT)	29	30	91.93	85.39
NVP (RT)	25	26	94.31	85.52
NVP (PR + RT)	25	26	94.68	85.99

Table 5.7: Performance of pruned C4.5 classification algorithm

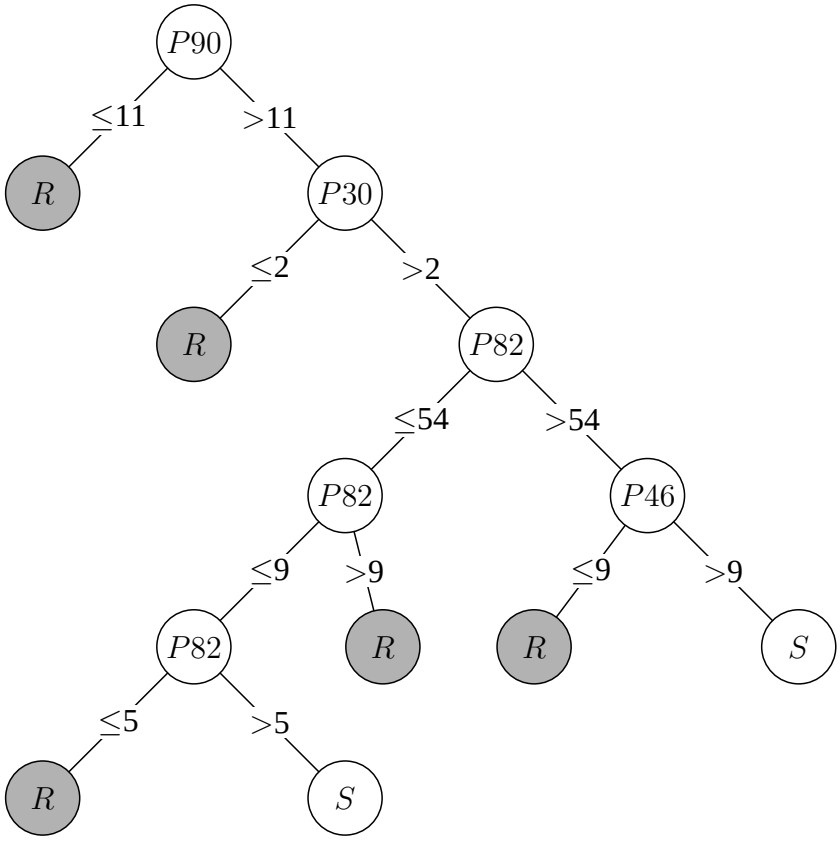

Figure 5.4: Example Decision tree for NFV. The label of the interior nodes refers to the position of the codon in the sequence (for example P30 means amino acid at position 30) and the label of the leaves indicated the drug resistance property of the strain

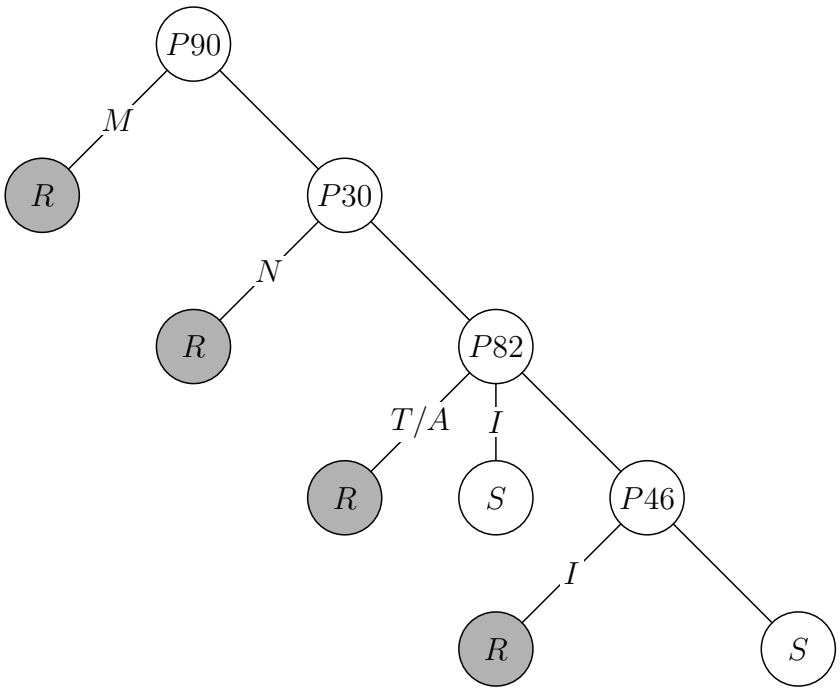

Figure 5.5: Biologically interpreted decision tree for NFV given in Figure 5.4. The label for the edges represents the amino acid (codon) at the position give in the node above it. M = *Methionine* (AUG), T = *Threonine* (ACC), N = *Asparagine* (AAU), I = *Isoleucine* (AUC) and A = *Alanine* (GCU) (see Table C.1 in Appendix C)

decision trees were simple with the most complicated one having an average of 29 interior nodes (see Table 5.7). An example of a decision tree for the drug NFV is given in Figure 5.4. This tree is one of the 10 trees generated for this drug and is only selected as an example because it has the average tree size. The biological interpretation the decision trees is not in the scope of these research and requires deep biological knowledge. However, there are some biological aspects of these trees we want to discuss. For each of the interior nodes of the resulting trees for IDV, NFV, AZT(RT), D4T(RT) and NVP(RT) we tried to compare if these mutation points are previously reported. We have found (result not shown for all) that most of the interior nodes are reported as significant mutation points. For example, for the drug NFV (decision tree given in Figure 5.5) all four of the interior nodes P30, P46, P82 and P90 are identified as positions associated with drug resistance[1] (see Beerenwinkel *et al.* [2002] and Drug resistance summaries on `http://hivdb.stanford.edu/index.html`). However, not all of the position associated with drug resistance are interior nodes on our tree. For example position 71 of the protease sequence is indicated as a position associated with drug resistance to NFV by Beerenwinkel *et al.* [2002] does not appear as a split criteria in our tree. The absence of these and other positions as an interior node from our tree might be due to a number of reasons. Firstly, the number of samples in our data set associating drug resistance with these positions might be very few. Secondly, most of the mutation patterns are correlated and hence might be ignored as a split criteria. On the contrary, we have also seen positions which are not previously associated with mutation to the particular drugs appear as an interior node. For example, some

[1]The mutation related to these positions are 30N, 46I, 82A, 82T, 82I, 90M

positions inside the protease part of the sequence have appeared as an interior node for the decision tree for D4T(PR + RT). This phenomenon is very hard to reason out without further clinical study. However, it has been argued by a number of researchers that one of the limitations of current clinical research is that only an isolated gene is studied during drug design. And there might be a possibility that these positions have an effect in the drug resistance behaviour of the drug.

The result shown in Table 5.7 gives the average performance of the pruned tree. During the experiment we first constructed the unpruned tree and then each of these trees were pruned using information gain ratio criterion. We tested different values for pruning confidence factor in order to prevent both over and under fitting. As it can be seen from table we have found accuracy in the range 91.93% to 97.06% and area under the ROC curve in the range 85.29% to 94.43%. The variation of performance between the decision tree models for D4T(RT)and D4T(PR + RT), and AZT(RT) and AZT(PR + RT) might be due to the codons from protease which are used as a split criteria for decision trees for D4T(PR + RT) and AZT(PR + RT). However, the performance variation between NVP(RT) and NVP(PR + RT) is very small compared to the previous two. This variation in the performance between the two models of decision tree for NVP might be due to the reason than there is only one position belonging to protease which has occurred as a split criteria deep inside the tree (decision tree not shown).

The variation in the performance is not in contradiction with the argument made in the above paragraph about the possible effect of protease mutation in drug resistance behaviour of reverse transcriptase inhibitors. The main reason behind this variation might be that the data used in this research is already classified with drug resistance prediction algorithm that considered protease or reverse transcriptase genes alone for the respective inhibitor drug resistance prediction.

Finally, the high performance rate of the decision tree model together with the confirmation of most interior nodes as positions associated with drug resistance can be considered as a confirmation for input encoding technique used.

5.5 Performance of logistic regression

The final part of the experiment was performance evaluation of logistic regression techniques. As mentioned in Section 2.8 the choice of an optimal value for the ridge parameter is crucial. It was also mentioned that cross-validation is the most successfull way of estimating it. Hence, like the rest of the experiment, we also used 75% of the data randomly selected for training and the remaining 25% for testing. The result given in Table 5.8 is the mean value for the accuracy and the AUC over 10 experiments.

In this part of experiment we used logistic regression with ridge estimate. Different values for ridge parameter were tested. The results presented in the table are the best performance recorded. The accuracy for logistic regression model ranges 75.78% to 93.39% and AUC ranges from 50.96% to 85.06%.

Drug	Accuracy	AUC
IDV	92.66	85.06
NFV	93.39	83.58
AZT (RT)	81.10	78.24
AZT (PR + RT)	84.40	78.01
D4T (RT)	80.55	65.37
D4T (PR + RT)	80.38	63.31
NVP (RT)	75.78	46.26
NVP (PR + RT)	77.98	50.96

Table 5.8: Performance of logistic regression

5.6 Discussion

Evaluation of different algorithms usually leads to a biased ranking of the algorithms. As specified in section 4.4, theoretical estimation of the generalisation ability of different algorithm requires deep knowledge about the underlying distribution of data. These information is usually not available hence researchers are forced to find alternative approach that can give unbiased estimator of the true error rate of a classifier. When the number of training and testing patterns are limited, a single train-and-test experiment will result in a misleading information. To avoid this problem and get unbiased estimate on the performance of the different algorithms, this research used cross-validation. The performance for each of the different classification algorithms is given in Table 5.9. The results show that SVMs produce the best results in all cases. Overall, decision trees were the second best (except for NVP). The performance of the different models on NVP is relatively poor compared to the other drugs. This might be due to the relative skewness of the data sets for this drug. Neural networks performed well compared to logistic regression but was inferior to the other two classifiers.

	SVM		NN		DT		Logistic Reg	
Drug	Accu.	AUC	Accu.	AUC	Accu.	AUC	Accu.	AUC
IDV	96.33	95.52	89.75	91.76	97.06	92.70	92.66	85.06
NFV	96.15	97.49	91.74	95.31	97.06	94.43	93.39	83.58
AZT (RT)	94.13	96.64	82.20	88.57	94.86	90.96	81.10	78.24
AZT (PR + RT)	93.21	96.64	82.02	87.70	96.33	93.41	84.40	78.01
D4T (RT)	94.86	97.17	84.34	91.85	91.93	85.39	80.55	65.37
D4T (PR + RT)	93.58	97.11	76.33	84.09	93.58	88.29	80.38	63.31
NVP (RT)	94.50	81.26	70.49	69.50	94.31	85.52	75.78	46.26
NVP (PR + RT)	82.20	80.16	64.77	71.10	94.68	85.99	77.98	50.96

Table 5.9: Performance of the different classification algorithms

Decision trees had a slightly better accuracy than SVMs on almost all data sets however, SVMs have overall better area under the curve which means the SVM models have an overall better trade-off between sensitivity and specificity compared to decision trees (see the graphs in Figure 5.6 and 5.7). There can be a number of reasons for the impressive performance of

decision trees. Firstly, as we have mentioned in section 5.4 the training and testing set was previously classified using a knowledge based approach, hence best suits decision tree model. Secondly, the discrete data encoding can be argued to suit the decision tree model than SVMs.

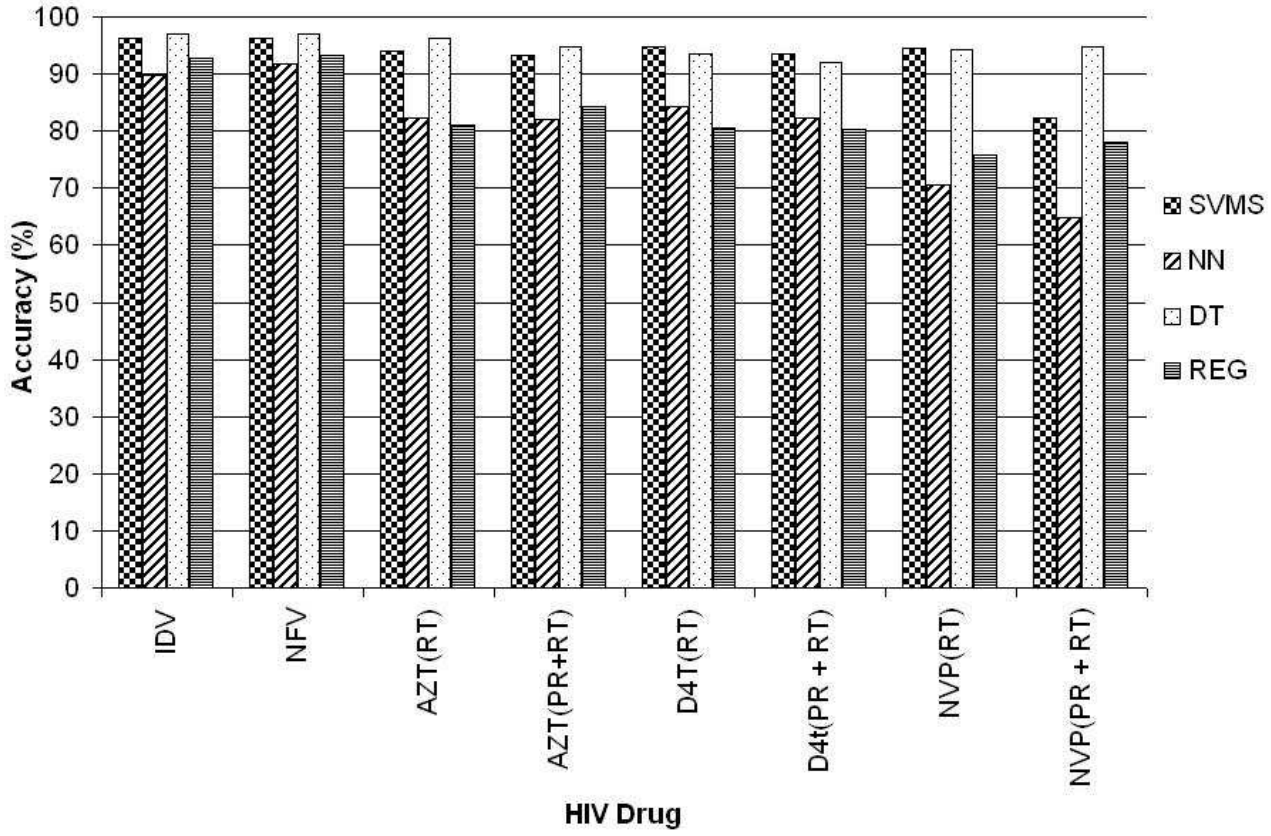

Figure 5.6: Accuracy of the different classifiers

Comparing the performance of SVMs and neural network, it can be seen from the table that neural networks were outperformed by SVMs on all datasets (see the graphs in Figure 5.6 and 5.7). This might be attributed to the dimensionality of the data compared to the number of available training samples and the loss of information during dimensionality reduction. As we have mentioned in section 5.4 and Chapter 2, mutation points complement each other to cause drug resistance. While some mutations are major and cause resistance alone, others are secondary and needs the existence of other mutations to cause drug resistance. That means these attributes are highly correlated and might be given lower rank during dimensionality reduction using principal component analysis. Besides its performance, SVMs converged rapidly compared to neural network even on the reduced dimensionality. The results also shows that the SVM models also outperformed the logistic regression models.

Although there are no previous research known to us which used SVMs to directly predict HIV drug resistance, to check the performance of our model we made comparison to the SVM regression model by Beerenwinkel *et al.* [2003a], the decision tree models by Beerenwinkel *et al.* [2002] and the neural networks models by Draghici and Potter [2003] and Wang and Larder

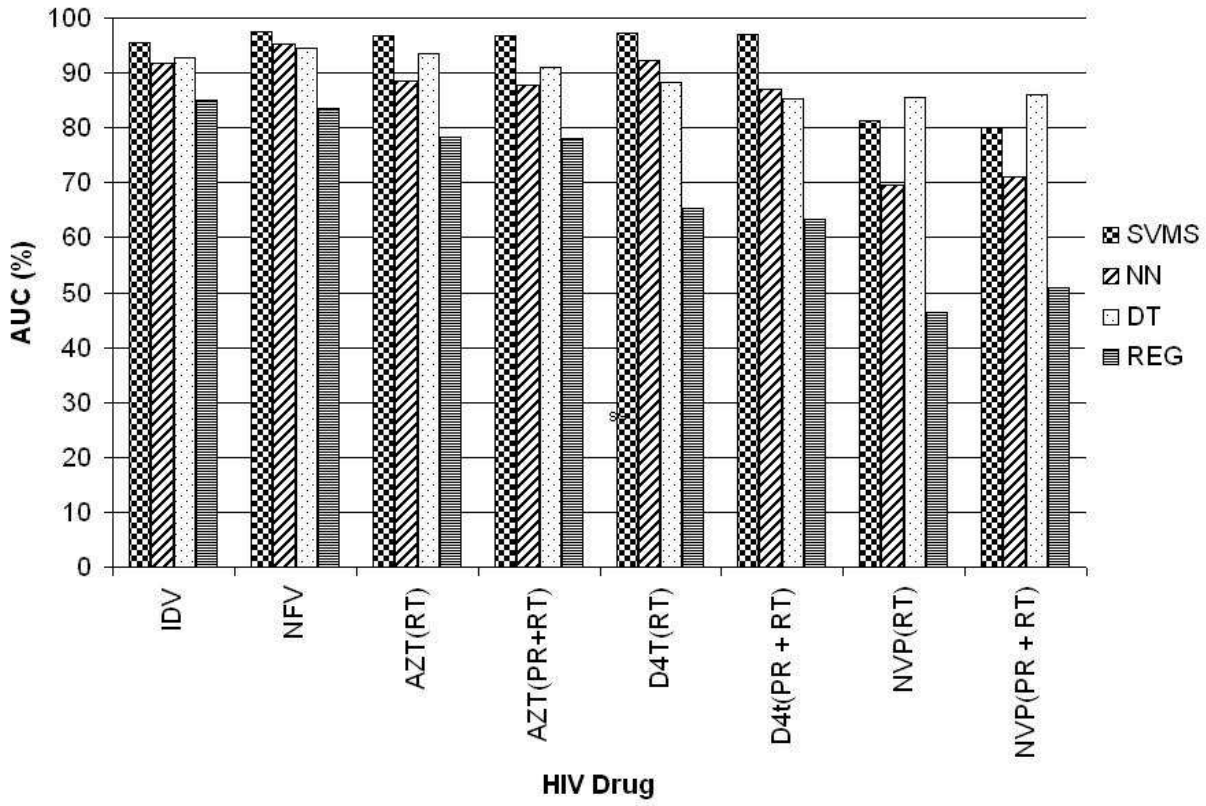

Figure 5.7: Area under the ROC curve of the different classifiers

[2003].

The difference in the size of the data sets used, the error estimation techniques and the test method are different and makes direct comparison of our models with models in previous work difficult. The SVMs models for this research can be compared positively to SVMs model by Beerenwinkel *et al.* [2003a] where SVM regression is used to predict the genotypic resistance from phenotypic resistance. Similarly the decision tree models for this research produced a better performance than the decision tree models presented in Beerenwinkel *et al.* [2002] in all of the five drugs. Comparing our neural networks models with the neural networks models in Draghici and Potter [2003] our model performed better than the single network model of Draghici and Potter [2003]. However, the performance of the neural network model by Wang and Larder [2003] on one drug in the protease inhibitor category was slightly higher than what we have achieved which might be attributed to the incorporated previous knowledge about the drugs.

A key result is minimal biological knowledge or expert knowledge was used (we only used the fact of what the drug was targeted at to select the general genomic region to study). We do not wish to downplay the contribution that expert knowledge can make to understanding the mutation process (after all, we are only using the one-dimensional knowledge given by the genomic sequence, when the virus and its RNA/DNA are both complex three dimensional objects). However, these results show that machine learning can significantly complement and

assist the work of human experts working on the virus.

5.7 Statistical significance

It has become a tradition to report statistical significance if a two-tailed, paired t-test produces a p-value less than some significance level (usually 0.05) [Salzberg 1999]. According to Salzberg [1999], these statistical tests should be used very carefully when used for classifiers comparison as they are not designed for computation experiments. Furthermore, Salzberg [1999] has highlighted following problems:

- The t-test assumes that the test sets are independent for each algorithm. In this research cross-validation with random partitioning of the data set between training and testing was used. And the different algorithms are trained and tested on the same data set. This means that the test sets share some of the patterns and hence are not independent. Dietterich [1996, as cited in Salzberg [1999]] has pointed out that this has a "high probability of Type I error ... and should never be used".

- The use of wrong p-value finds statistical significance where there is none. This is usually addressed by making adjustment to the significance level called *Bonferroni* adjustment provided that the experiments are independent of one another.

Salzberg [1997] highlighted k-fold cross validation as a recommended approach, but has also emphasised that it will not produce valid statistics because the test data are interdependent. Recommended alternative approaches can be found in Salzberg [1997], Salzberg [1999] or [Gascuel and Caraux 1992]. For this research, we have chosen to present the performance of the different classifier graphically as given in Figure 5.6 and Figure 5.7.

5.8 Limitations of the research

The results achieved in this research are not without their limitations. Some of the limitations can be easily addressed while the others are more complicated and costly. One of the limitation is the availability of the data. As stated in the introduction section, our wish was designing a model without incorporating the existing biological knowledge. To do this one needs a pure phenotypic HIV drug resistance data which is very expensive to generate, has a very low quality as a computational data and might present a different level of difficulty.

Another limitation is number of training samples. Again due to the cost of data generation and the politics behind patient-doctor privilege it is always hard to collect a large number of data. As an example due to the small number sequences labelled intermediate (I) we were forced to perform two-class classification rather than multi-class classification. Another limitation that needs to be addressed is the input encoding technique. In this research, an input encoding technique used converts the patterns in to a discrete vector. This encoding might be well suited

for one algorithm more than the other and might have affected the ranking of the algorithms according to performance.

This research can further be extended to address the above limitations. Further improvement to this research can be obtained by collecting high quality and large number of data from different gene databases. The same target can also achieved by working collaborations with HIV laboratories. This research can also be extended to make it more user friendly, web based so that it can be accessible a wide range of experts working in the field. Further more this research can be extended to by incorporating different feature extraction and reduction techniques.

5.9 Summary

Performing comparative research is always a source of controversy among pattern recognition societies. The reason for conducting comparative analysis of performance of SVMs against neural network, decision trees and/or logistic regression was not to verify or falsify the superiority of one classification technique over the other. As a relatively new addition to the pattern recognition techniques, SVMs have been compared to numerous pattern recognition techniques in a number of benchmark application. The result from these research claimed SVMs has a better performance on high dimensional data when faced with limited training samples. Dimensionality of the patterns and shortage of training sample is one of the common characteristics of many computation biology problems. Following this, this research investigates if SVMs can be used as a drug resistance tool based on an HIV strain. The results found might not be used directly to answer the question however, it has promised positive result.

The research have a number of limitations, however its significance in the bioinformatics and computational biology research can not be ignored.

Chapter 6

Conclusion and Future work

6.1 Conclusion

The Human Immuno-deficiency Virus (HIV) has infected millions of people and is spreading with a very high rate. One of the reasons for the failure in combating the epidemic is the drug resistant behaviour of the virus, which results from the rapid mutation rate. HIV drug resistance testing that helps optimise drug administration has shown a positive result in prolonging viral suppression and helps to reconstruct patient immunity. Phenotypic testing and genotypic testing are the two approaches for doing HIV drug resistance tests. The genotypic approach has gained a lot of interest because it is cost effective, easy to conduct and the interpretation of the result is independent of the process. The latter advantage has also made genotypic testing an ideal application for computerized expert systems and/or pattern recognition techniques.

Pattern recognition techniques have been used to solve problems that cannot be solved using the traditional algorithmic approach. These techniques are particularly useful when the relation between the input and the output is not defined properly, the data to be processed is enormous in size and very hard to analyse manually. This research used SVMs as a pattern recognition technique.

Although SVMs are relatively new, have shown outstanding performance as a tool for pattern recognition. The principle of structural risk minimisation and the ability of SVMs to transform the input data to higher dimensional feature space (kernel trick) are the secret behind the success achieved by SVMs. Other traditional pattern recognition techniques use the principle of empirical risk minimisation, which minimises the training error and usually suffers from over-fitting. On the other hand, the principle structural risk minimisation, which minimises the upper bound on the test error, gives SVMs the ability to generalise better. SVMs can be simply described as a combination of linear network, regularisation and kernel trick. It discriminate the positive examples from the negative ones using a hyperplane while maintaining a maximum margin between them. Many real-world problems are not linearly separable (separable by a linear hyperplane). To classify non-linear data sets, SVMs use a kernel function that implicitly works in a higher dimensional feature space, where the data can be discriminated using a linear hyperplane.

There are a number of different implementations of SVMs. This research used SVM^{light}, a C implementation of Vapnik's support vector machine. SMV^{light} is selected because it the most popular implementation, it is optimal and can handle large problems. It also has the ability to incorporate user defined kernel function and a cost model besides the three popular kernel functions.

The input data used to answer the research question is the genetic sequence of the viral protease and reverse transcriptase from 2045 individuals. These sequences were previously classified using some of the popular drug resistance interpretation algorithms. On this data set the performance of SVMs, neural networks, decision trees and logistic regression were tested. The best performance for these algorithms was configured using grid-search over a range of model parameters.

The results showed that SVMs outperformed the neural networks and logistic regression. The performance of the decision tree models and the SVMs models was almost similar. The results in this research confirms the findings of many researchers outlining the power of SVMs as a classification tool.

6.2 Future work

Although this research showed the ability of SVMs as a tool for predicting the HIV drug resistance based on the genetic sequence of the virus, it has a number of limitations. Following the achievements of this work and to some of address the limitations, the following future directions are proposed:

1. The two major limitations identified arise from the number and quality of data available. To address this limitation further research should be conducted by gathering large size of quality data and performing not only binary but multi-class prediction of HIV drug resistance.

2. As pointed out in the previous chapter, the input encoding technique might have favoured some classification algorithms than the others. Section 4.6 also highlighted the effect of different input encoding scheme however, a detailed study was not done. To investigate the effect of input encoding schemes in the performance of the different classification algorithm further research should be conducted.

3. The data used in this research was previously classified using rule-based algorithms. Although this does not compromise what is achieved in this research, conducting the research using pure phenotypic data will give more insight and might present a different level of challenge.

4. In this research we only concentrated on the computation property and computational achievement of the different classification tools. However, further work should be done

in close collaboration with experts in the field to investigate the biological implications of the results.

5. The results for the decision tree model showed that interior nodes were related to existing mutation points, even though we did not use any of the prior biological information. Further research can be conducted to investigate the application of SVMs to identify mutation points using SVMs for feature selection. This can be carried out by performing sensitivity analysis on SVMs and investigate if this technique can be used to identify mutation points. This work should also be done in collaboration with biological experts.

6. Similar to the above future work, Boz [2002] investigated ways of converting a trained neural network into a decision tree in order to make the complex rules discovered by the neural network model more human understandable. In our particular problem this can also be used identify mutation points which are known to confer drug resistance. This mutation points are the split criteria. Hence, future research can be carried out to investigate ways of converting a trained SVMs model into a decision tree.

Bibliography

[Baldi and Brunak 2001] Pierre Baldi and Soren Brunak. *Bioinformatics: the machine learning approach*. MIT Press, Cambridge, MA, USA, 2001.

[Bean 2000] Pamela Bean. An introduction to HIV drug resistance testing. *Clinical Note*, pages 10–11, May 2000.

[Beerenwinkel *et al.* 2002] Niko Beerenwinkel, Barbara Schmiot, Hauke Walter, Rolf Kaiser, Thomas Lengauer, Daniel Hoffmann, Klaus Korn, and Joachim Selbing. Diversity and complexity of HIV-1 drug resistance: A bioinformatics approach to predicting phenotype from genotype. *Proceeding of the National Academy of Sciences of the United States of America*, 99(12):8271 – 8276, June 2002.

[Beerenwinkel *et al.* 2003a] Niko Beerenwinkel, Martin Daumer, Mark Oette, Klaus Korn, Daniel Hoffmann, Rolf Kaiser, Thomas Lengauer, Joachim Selbig, and Hauke Walter. Geno2pheno: estimation phenotypic drug resistance from HIV-1 genotypes. *Nucleic Acids Research*, 31(13):3850 – 3855, 2003.

[Beerenwinkel *et al.* 2003b] Niko Beerenwinkel, Thomas Lengauer, Martin Dumer, Rolf Kaiser, Hauke Walter, Klaus Korn, Daniel Hoffmann, and Joachim Selbig. Methods for optimizing antiviral combination therapies. *Bioinformatics*, 19:i16–i25, 2003.

[Boser *et al.* 1992] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 27-29 1992. ACM Press.

[Boz 2002] Olcay Boz. Extracting decision trees from trained neural networks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 456–461, Edmonton, Alberta, Canada, July 23-26 2002.

[Bradley 1997] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[Breiman *et al.* 1984] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. The Wadsworth Statistics/Probability Series. Wadsworth International Group, Belmont, California, USA, 1984.

[Brown *et al.* 2000] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, Jr. M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences of the United States of America*, 97(1):262–267, January 2000.

[Burges 1998] Christopher J. C. Burges. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Cessie and van Houwelingen 1992] S. Le Cessie and J. C. van Houwelingen. Classification of gene microarrays by penalized logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[Chan 1996] Lia-Wan Chan. Levenberg-Marquardt learning and regularization. In Shun-Ichi Amari, Lei Xu, Lai-Wan Chan, Irwin King, and Kwong-Sak Leung, editors, *Progress in Neural Information Processing: Iconip'96 : Proceedings of the International Conference on Neural Information Processing*, pages 139 – 144, Hong Kong, 24-27 September 1996. Springer-Verlag Singapore.

[Coffin 1999] J. M. Coffin. Molecular Biology of HIV. In Keith A. Crandall, editor, *Evolution of HIV*, pages 3–41. John Hopkins Univeristy Press, 1999.

[Cortes and Vapnik 1995] C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20:273 – 297, 1995.

[Cristianini and Shawe-Taylor 2000] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[Delen and Kadam 2005] G. Walker Delen, D. and A. Kadam. Predicting breast cancer survivability: a comparison of three data mining methods. *Journal of Artificial Intelligence in Medicine*, 34(2):113 – 127, June 2005.

[Demuth and Beale 2004] Howard Demuth and Mark Beale. *Neural Network Toolbox: User guide*, June 2004.

[Diem 1962] Konrad Diem, editor. *DOCUMENTA GEIGY : SCIENTIFIC TABLES*. Geigy Pharmaceuticals, Basle, Switzerland, 1962.

[Dietterich 1996] T. Dietterich. *Statistical tests for comparing supervised learining algorithm*. Technical report, Oregon State University, Corvallis, OR, 1996.

[Draghici and Potter 2003] S. Draghici and R. B. Potter. Predicting HIV drug resistance with neural network. *Bioinformatics*, 19(1):98 – 107, January 2003.

[Draghici *et al.* 2000] S. Draghici, L. Cumberland, and L. C. Kovari. Correlation of HIV protease structure with Indinavir resistance: a data mining and neural network approach.

In Belur V. Dasarathy, editor, *Proceedings, the International Society for Optical Engineering*, volume 4057 of *Data Mining and Knowledge Discovery: Theory, Tools, and Technology II*, pages 319 – 329, Orlando, Florida, April 2000.

[Duda *et al.* 2000] Richard O. Duda, Peter E. Hart, and David G. Stock. *Pattern Recognition and Scene Analysis*. John Wiley & Sons, New York, 2000.

[Dumais *et al.* 1998] Susan T. Dumais, John Platt, David Hecherman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. 7th International Conference on Information and Knowledge Management*, pages 148–155, Bethesda, Maryland, United States, November 3-7 1998. ACM Press.

[Eilers *et al.* 2001] P. H. Eilers, J. M. Boer, G.-J. van Ommen, and H. C. van Houwelingen. Classification of microarray data with penalized logistic regression. In *Proc. SPIE Vol. 4266, p. 187-198, Microarrays: Optical Technologies and Informatics, Michael L. Bittner; Yidong Chen; Andreas N. Dorsel; Edward R. Dougherty; Eds.*, pages 187–198, June 2001.

[Fawcett 2003] Tom Fawcett. *ROC Graphs: Notes and Practical consideration for Researchers*. Technical report, HP Laboratory, 2003.

[Ferri *et al.* 2003] César Ferri, José Hernández-Orallo, and Miguel A. Salido. Volume under the ROC surface for multi-class problems. In *ECML*, pages 108–120, Cavtat-Dubrovnik, Croatia, November 2003. LNAS Springer Verlag.

[Friedman 1996] Jerome H. Friedman. *Another Approach to Polychromous Classification*. Technical report, Stanford University, 1996. `http://www-stat.stanford.edu/~jhf/ftp/poly.ps.Z`.

[Fu 1974] K. S. Fu. *Syntactic Methods in Pattern Recognition*. Academic Press, New York, 1974.

[Furey *et al.* 2000] Terrence S. Furey, Nigel Duffy, Nello Cristianini, David Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[Ganapathiraju 2002] Aravind Ganapathiraju. *Support Vector Machines for speech recognition*. PhD thesis, Mississippi State University, May 2002. `http://www.ece.msstate.edu/academics/gradprog/dissertations/ganapathiraju_dissertation.pdf`,.

[Gascuel and Caraux 1992] Olivier Gascuel and Gilles Caraux. Statistical significance in inductive learining. In *Proceedings of the European Conference on Artificial Intelegence (ECAI)*, pages 435 – 439, New York, USA, 1992. Wiley.

[Goutte 1997] Cyril Goutte. Note on free lunches and cross-validation. *Neural Computation*, 9(6):1245–1249, 1997.

[Greene and Peterlin 2003] W. C. Greene and B. M. Peterlin. Molecular Insights into HIV biology. *Nature Science of HIV*, February 2003. HIV InSite Knowledge Based Chapter at `http://hivinsite.ucsf.edu/`.

[Gutierrez-Osuna] Richardo Gutierrez-Osuna. Pattern recognition lecture notes. `http://research.cs.tamu.edu/prism/lectures.htm`.

[Hagan and Menhaj 1994] Martin T. Hagan and Mohammad B. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE Transaction on Neural Networks*, 5(6):989 – 993, November 1994.

[Hastie and Tibshirani 1996] T. Hastie and R. Tibshirani. *Classification by pairwise coupling*. Technical report, University of Toronto, 1996. `http://www-stat.stanford.edu/~hastie/Papers/nips97.ps`.

[Health Canada 2001] Health Canada. *HIV-1 Strain and Primary Drug Resistance in Canada. surveillance report to June 30 2002*. Division of HIV/AIDS Epidemiology and Surveillance, Centre for Infectious Disease Prevention and Control, Health Canada, 2001. `http://www.hc-sc.gc.ca/pphb-dgspsp/publicat/hiv1-vih1/`.

[Hearst *et al.* 1998] M.A. Hearst, B. Schölkopf, S. Dumais, E. Osuna, and J. Platt. Trends and Controversies - Support Vector Machines. *IEEE Intelligent Systems*, 13(4):18 – 28, July 1998.

[Hoffmann and Kamps 2003] Christian Hoffmann and Bernd Sebastian Kamps. *HIV Medicine 2003*, January 2003. `www.hivMedicine.com`.

[Hunter 1993] Lawrence Hunter. *Artificial Intelligence and Molecular Biology*, chapter Molecular biology for computer scientists. American Association for Artificial Intelligence, Menlo Park, CA, USA, March 1993.

[Jain and Chandrasekaran 1982] Anil K. Jain and B. Chandrasekaran. Dimensionality and sample size consideration in pattern recognition practice. In P.R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics*, volume 2, pages 835 – 855. North-Holland Publishing Company, Amsterdam: North Holland, 1982.

[Jain *et al.* 2000] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, January 2000.

[Joachims 1998] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, April 1998. Springer Verlag, Heidelberg, DE.

[Joachims 1999] Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Chris Burges, and Alex Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–185. MIT Press, 1999.

[Jolliffe 1986] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[Kanal 2000] Laveen Kanal. Pattern in pattern recognition: 1968 - 1974. *IEEE Transactions on Information Theory*, IT-20(6):697 – 722, November 2000.

[Kass 1980] G.V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–172, 1980.

[Keerthi and Lin 2003] S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation.*, 15(7):1667–1689, 2003.

[Klatt 2003] Edward C. Klatt. *Pathology of AIDS*. Florida State University, College of Medicine, Florida, USA, May 2003. `http://medlib.med.utah.edu/WebPath/AIDS2004.pdf`.

[Kohavi and Quinlan 2002] Ronny Kohavi and J. Ross Quinlan. *Handbook of data mining and knowledge discovery*, chapter Data mining tasks and methods: Classification: decision-tree discovery, pages 267–276. Oxford University Press, Inc., New York, NY, USA, 2002.

[Kroon 2003] Rodeney Stephen Kroon. *Support Vector Machines, Generalization Bounds and Transduction*. Master's thesis, University of Stellenbosch, December 2003.

[Lathrop and Pazzani 1999] R. H. Lathrop and M. J. Pazzani. Combinatorial Optimization in Rapidly Mutating Drug-Resistant Viruses. *Journal of Combinatorial Optimization*, 3(2):301 – 320, July 1999.

[Lathrop *et al.* 1999] R. H. Lathrop, N. R. Steffen, M. P. Raphael, S.Deeds-Rubin, M. J. Pazzani, P. J. Cimoch, D. M. See, and J. G. Tilles. Knowledge-based Avoidance of Drug-Resistant HIV Mutant. *AI magazine*, 20(1):13 – 25, spring 1999.

[Lee *et al.* 2001] Yoonkyung Lee, Yi Lin, and Grace Wahba. *Multicategory Support Vector Machines*. Technical Report TR-1043, Department of Statistics, University of Wisconsin-Madison, Madison, USA, 2001. `http://www.galaxy.gmu.edu/interface/I01/I2001Proceedings/YLee/YLee.pdf`.

[Lin and Lin 2005] Hsuan-Tien Lin and Chih-Jen Lin. *A study of Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SOM-type Methods*. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, March 2005. `http://www.csie.ntu.edu.tw/~cjlin/papers/tanh.pdf`.

[Loh and Shih 1997] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statisica Sinica*, 7:815 – 840, 1997.

[Marquardt 1963] Donald Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431 – 441, 1963.

[Mayoraz and Alpaydin 1999] Eddy Mayoraz and Ethem Alpaydin. Support Vector Machines for multi-class classification. In *International Work-conference on Artificial and Natural Neural Networks (2)*, pages 833–842, Alicante, Spain, June 2-4 1999.

[Meyer *et al.* 2002] David Meyer, Friedrich Leisch, and Kurt Hornik. *Benchmarking Support Vector Machines*. Report Series Report No 78, Vienna University of Economics and Business Administration, 2002.

[NIAID 2004] NIAID. *National Inistitute of Aliergy and Infectious Diseases: HIV Vaccine Glossary*, May 2004. `www.niaid.nih.gov/factsheets/GLOSSARY.htm`.

[Norris 2002] Amy Norris. *Multivariate Analysis and Reverse Engineering of Signal Transduction Pathways*. Master's thesis, Institute of Applied Mathematics, Uniersity of British Colombia, Canada, April 2002. `http://www.iam.ubc.ca/theses/Norris/ANorris_MSc_thesis.pdf`.

[Olson 2000] Clark F. Olson. Maximum-Likelihood Template Matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 52–57, Hilton Head, South California, June 2000.

[Olszewski 2001] Robert T. Olszewski. *Generalized Feature Extraction for Structural Pattern Recognition in TimeSeries Data*. PhD thesis, School of Computer Science; Carnegie Mellon University, February 2001. `http://reports-archive.adm.cs.cmu.edu/anon/2001/abstracts/01-108.html`.

[Osuna *et al.* 1997] E. Osuna, R. Freund, and F. Girosi. *Support Vector Machines: Training and Application*. Technical Report AIM-1602, MTI A.I. Lab, 1997. `http://hdl.handle.net/1721.1/7290`.

[Pavlidis 1977] Theodosios Pavlidis. *Structural Pattern Recognition*. Springer, Berlin, Germany, 1977.

[Perlich *et al.* 2004] Claudia Perlich, Foster Provost, and Jeffrey S. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research*, 4(2):211 – 255, February 2004.

[Platt 1998] J. Platt. *Sequential minimal optimization: A fast algorithm for training support vector machines*. Technical Report MSR-TR-98-14, Microsoft Research, April 1998. `http://research.microsoft.com/users/jplatt/smoTR.pdf`.

[Plutowski *et al.* 1994] Mark Plutowski, Shinichi Sakata, and Halbert White. Cross-validation estimates imse. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *In Advances in Neural Information Processing Systems*, volume 6, pages 391 – 398. Morgan Kaufman, San Mateo, CA, 1994.

[Provost *et al.* 1998] Foster J. Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453, Madison, Wisconsin USA, July 24-27 1998. Morgan Kaufmann Publishers Inc.

[Quinlan 1987] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

[Quinlan 1993] J. Ross Quinlan. *C4.5: Programmes for Machine learning*. Morgan Kaufmann, San Meteo, California, 1993.

[Quinlan 1996] J. Ross Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[Rakotomamonjy 2004] Alain Rakotomamonjy. *SVMs and Area under ROC curves*. Technical report, Département Architecture des systmes d'Informations de l'INSA de Rouen, 2004. `http://asi.insa-rouen.fr/~arakotom/publi/svmandauc.pdf`.

[Raudys and Jain 1991] Sarunas J. Raudys and Anil K. Jain. Small sample size effects in statistical pattern recognition: Recommendation for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252 – 264, March 1991.

[Ravela *et al.* 2003] Jaideep Ravela, Bradley J. Betts, Francoise Brun-Vzinet, Anne-Mieke Vandamme, Diane Descamps, Kristel Van Laethem, Kate Smith, Jonathan M. Schapiro, Dean L. Winslow, Caroline Reid, and Robert W. Shafer. HIV-1 protease and reverse transcriptase mutation patterns responsible for discordances between genotypic drug resistance interpretation algorithms. *Journal of Acquired Immune Deficiency Syndromes, JAIDS*, 33:8 – 14, 2003.

[Riedmiller and Braun 1993] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learining: The RPROP algorithm. In *Proceedings of the*

IEEE International Conference on Neural Networks, pages 586–591, San Francisco, CA, March 28 1993. IEEE Computer Society.

[Riedmiller 1994] Martin Riedmiller. Advanced supervised learning in multilayer perceptions - from backpropagation to adaptive learning techniques. *International Journal of Computer Standards and Interfaces. Special Issue on Neural Networks*, 16:265–278., 1994.

[Rosenblatt 1962] Frank Rosenblatt. *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*. Spartan Books, Washington D.C., 1962.

[Roweis 1998] Sam Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[Rumelhart *et al.* 1994] David E. Rumelhart, Bernard Widrow, and Michael A. Lehr. The basic ideas in neural networks. *Communication of the ACM*, 37(3):87–92, 1994.

[Russell 1991] Ingrid F. Russell. Neural networks in the undergraduate curriculum. *Collegiate Microcomputer*, 9(1):1–6, February 1991.

[Saeys 2004] Yvan Saeys. *Feature Selection for classification of Nucleic Acid Sequences*. PhD thesis, Faculty of Science, Universiteit Gent, Gent, Belgium, September 2004. `http://www.psb.ugent.be/~yvsae/phd/phdyvsae.pdf`.

[Saitta and Neri 1998] Lorenza Saitta and Filippo Neri. Learning in the "Real World". *Machine Learning.*, 30(2-3):133–163, 1998.

[Salzberg 1997] Steven L. Salzberg. On comparing classifiers: Pitfalls to avoid and recommended approach. *Data Mining and Knowledge Discovery*, 1:317–328, 1997.

[Salzberg 1999] Steven L. Salzberg. On comparing classifiers: A critique of current research and methods. *Data Mining and Knowledge Discovery*, 1:1–12, 1999.

[Schalkoff 1991] Robert J. Schalkoff. *Pattern recognition: statistical, structural and neural approaches*. John Wiley & Sons, Inc., New York, USA, 1991.

[Schimek 2003] Michael G. Schimek. Penalized logistic regression in gene expression analysis. In *Proceedings of 2003 semiparametic conference*, Berlin, Germany, 2003.

[Schölkopf and Smola 2001] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, USA, December 2001.

[Schölkopf *et al.* 1996] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. *Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers*. Technical Report AIM-1599, Massachusetts Institute

of Technology, Cambridge, MA, USA, 1996. http://cbcl.mit.edu/cbcl/publications/ai-publications/1500-1999/AIM-159%9.ps.

[Schölkopf 1997] Bernhard Schölkopf. *Support Vector Learning*. PhD thesis, Technischen Universität Berlin, 1997. Published by: R. Oldenbourg Verlag, Munich http://www.kernel-machines.org/papers/book_ref.ps.gz.

[Shafer 2002a] R. W. Shafer. Genotypic Testing for Human Immuniodeficency Virus Type 1 Drug Resistance. *Clinical Microbiology Reviews*, 15(2):247 – 277, April 2002.

[Shafer 2002b] Robert W. Shafer. Assays for antiretroviral resistance. *HIV Insight*, June 2002.

[Shannon 2001] Claude. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[Shao 1993] Jun Shao. Linear model selection by cross validation. *Journal of the American Statistical Association*, pages 486 – 494, 1993.

[So 1995] Ying So. *A Tutorial on Logistic Regression*. Technical Report 450, SAS Institute Inc., Cary, NC, 1995. http://www.sas.com/service/techsup/tnote/tnote_index4.html.

[Sollich 2000] Peter Sollich. Probabilistic methods for support vector machines. In S.A. Solla, T.K. Leen, and K.-R. Mller, editors, *Advances in Neural Information Processing*, volume 12, pages 349 – 355. MIT Press, Cambrige, MA, USA, 2000.

[Spackman 1989] Kent A. Spackman. Signal detection theory: valuable tools for evaluating inductive learning. In *Proceedings of the sixth international workshop on Machine learning*, pages 160–163, Ithaca, New York, USA, June 29 - July 1 1989.

[Sun *et al.* 2002] Aixin Sun, Ee-Peng Lim, and Wee-Keong Ng. Web classification using Support Vector Machine. In *Proceedings of the Fourth International Workshop on Web Information and Data Management*, pages 96–99, McLean, Virginia, USA, November 8 2002. ACM Press.

[Theodoridis and Koutroumbas 1999] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, London, UK, 1999.

[Tou and Gonazalez 1974] Julius T. Tou and Rafael C. Gonazalez. *Pattern Recognition Principles*. Addison-Wesley Publishing Company, Massachusetts, USA, 1974.

[Trunk 1979] G. V. Trunk. A problem of dimensionality: A sample example. *IEEE Transactions on Pattern analysis and machine intelligence*, PAMI-1(3):306 – 307, July 1979.

[UNAIDS 2004] UNAIDS. *AIDS epidemics update: 2003*, 2004. http://www.who.int/hiv/pub/epidemiology/epi2003/en/.

[Vanderbei 1999] Robert. J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 11:451–484, 1999.

[Vapnik and Chervonenkis 1991] V. N. Vapnik and A. Ya. Chervonenkis. The necessary and sufficient condition for consistency in the empirical risk minimization methods. *Pattern Recognition and Image Analysis*, 1(3):283 – 305, 1991.

[Vapnik 1979] V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

[Vapnik 1995] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[Wang and Larder 2003] Dechao Wang and Berndan Larder. Enhanced prediction of lopinavir resistance from genotype by use of artificial neural network. *The Journal of Infectious Diseases*, 188:653 – 660, September 2003.

[Watson *et al.* 1997] James Watson, Michael Gilman, Jan Witkowski, and Mark Zoller. *Recombinant DNA*. Scientific American Books, New York, USA, 2nd edition, 1997.

[Webb 1999] Andrew Webb. *Statistical Pattern Recognition*. Arnold Publishers, London, UK, 1999.

[Weston and Watkins 1998] J. Weston and C. Watkins. *Multi-class Support Vector Machine*. Technical Report CSD-TR-98-04, Royal Holloway College, University of London, Egham, England, 1998. `http://www.cs.rhbnc.ac.uk/research/compint/areas/comp_learn/sv/pub/repo%rt98-04.ps.gz`.

[Whitney *et al.* 2002] James B. Whitney, Maureen Oliveira, Mervi Detorio, Yongjun Guan, and Mark A. Wainberg. The M184V mutation in reverse transcriptase can delay reversion of attenuated variants of simian immunodeficiency virus. *Journal of Virology*, 76(17), september 2002.

[Wikipedia 2004] Wikipedia. *Wikipedia: The free Encyclopedia*. `http://en.wikipedia.org/wiki/HIV`, 2004.

[Wu 1997] Cathy H. Wu. Artificial neural networks for molecular sequence analysis. *Computers & Chemistry*, 21(4):237 – 256, 1997.

[Zhu and Hastie 2004] Ji Zhu and Trevor Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5:427–443, 2004.

Appendix A

Statistical and Parametric Classifiers

A.1 Bayesian decision theory

Bayesian decision theory is a tool broadly used to solve pattern recognition problems, provided the problem is defined in terms of probability densities and all the probability values are defined. Having defined all the relevant probability valued, Bayesian decision theory is based on finding optimal trade offs between the various classification decisions and the accompanying cost (misclassification penality) [Duda *et al.* 2000]. The material on this section is extracted form Duda *et al.* [2000] unless otherwise specified.

Given a set of patterns represented by a *d*-dimensional feature vector $\mathbb{X} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ called the feature space, let ω be any of the *c* finite states of nature $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ and $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_a\}$ be the finite set of possible actions. The decision problem is then defined as choosing an appropriate action among the finite set of possible actions for an event in the world (an event belonging to one of the finite states of nature based on the measurement given by the feature vector x). The performance of the decision system is then measured by the loss function $\lambda : \mathcal{A} \times \Omega \rightarrow \mathbb{R}$, which establishes the cost $\lambda(\alpha_i|\omega_j)$ that describes the loss incurred by choosing action α_i when the state of nature is ω_j. However, we cannot observe the world and hence we need to consider the measurement $x \in \mathbb{R}^d$ and the mapping function $\alpha(x) : \mathbb{R}^d \rightarrow \mathcal{A}$ called the decision function, which is the function of measurement, that tells us which action to take for the given measurements.

The measurement x and the corresponding state of nature ω can be viewed as a single observation and might be considered in terms of probability. In this probabilistic framework, decision will be based on the posterior probability $P(\omega_j|x)$ for $j = 1, \ldots, c$, which is the probability that the measurement x belongs to the state of nature ω_j. An action α_j will be taken based on the largest probability once the posterior is computed for each state of nature. But how do we know the posterior probability?

Let $x \in \mathbb{R}^d$ be a random variable and assume the prior distribution $P(\omega_j)$, which tells how likely the nature of state is ω_j and let $p(x|\omega_j)$ be the state-conditional probability, which describes the relationship among the state of nature ω_j and measurement x are known. With the prior distribution and state-conditional density, the joint distribution, $p(\omega_j, x)$, of finding a

measurement in the state of nature ω_j and having the vector value x will be given as:

$$p(\omega_j, x) = P(\omega_j|x)p(x) = p(x|\omega_j)P(\omega_j) \tag{A.1}$$

rearranging the above equation we get the Bayes rule:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)} \tag{A.2}$$

where $p(x) = \sum_{\omega_j \in \Omega} p(x|\omega_j)P(\omega_j)$ is the probability density function for x.

$$\text{Therefore:} \qquad P(\omega_j|x) = \frac{p(x|\omega_j)P(w_j)}{\sum_{\omega_j \in \Omega} p(x|\omega_j)P(w_j)} \tag{A.3}$$

The Bayes rule tells us how the probability of the state of nature is updated by inverting the relationship among world state and measurement. The term $P(\omega_j|x)$, the probability that the true state of nature is ω_j, accounts for the fact that once the measurement is observed the probability of having a certain state changes. Suppose for the measurement x the action α_i is taken while the true state of nature is ω_j. According to the definition the penalty of taking action α_i while the true state of nature is ω_j is given by $\lambda(\alpha_i|\omega_j)$. And the average loss also known as the *conditional risk* of choosing this action is given by:

$$R(\alpha_i|x) = \sum_{\omega_j \in \Omega} \lambda(\alpha_i|\omega_j)P(\omega_j|x)$$

If we have a decision rule α which tells us which action to take for every possible observation we can substitute α_i by $\alpha(x)$ and obtain the average loss $R(\alpha(x)|x)$ given the measurement x. The overall average loss (*expected risk*) therefore will be obtained by averaging over all possible measurements and is given as:

$$R = \int_{\mathbb{R}^d} R(\alpha(x)|x)P(x)dx \tag{A.4}$$

The expected risk is a single scalar that measures the overall performance. Then the best decision rule is found by computing the expected risk for all possible actions and selecting the one that minimises the risk, i.e.

$$\alpha = \arg\min_{\alpha} R_\alpha = \arg\min_{\alpha} \int_{\mathbb{R}^d} R(\alpha(x)|x)P(x)dx, \qquad \forall x \in \mathbb{R}^d \tag{A.5}$$

By minimising point-wise the function under the sign of integral, it is easy to conclude that the optimal function is

$$\alpha(x) = \arg\min_{\alpha_i \in A} R(\alpha_i|x), \qquad \forall x \in \mathbb{R}^d \tag{A.6}$$

The resulting overall minimum risk is called the *Bayes risk*.

Bayesian pattern classification is based on Bayes decision theory to define the decision boundaries to perform pattern recognition tasks. Pattern recognition is a particular decision problem where the world is seen as a pattern source, the state of nature is seen as the pattern classes with the measurement being a pattern. Each action α_i is simply identifying the pattern to be in one of the pattern class. The decision rule $\alpha(x)$ is a discriminant function which maps the pattern x to one of the pattern classes. To evaluate the performance lets assume misclassification is equally bad and consider the 0/1 loss function, which is defined to be

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j, \\ 1 & i \neq j \end{cases} \qquad i, j = 1, \ldots, c \tag{A.7}$$

The optimal decision function is then

$$\begin{aligned} \alpha(x) = \arg\min_{\alpha_i} R(\alpha_i|x) &= \arg\min_{\alpha_i} \sum_{\omega_i} \lambda(\alpha_i|\omega_j) P(\omega_j|x) \\ &= \arg\min_{\omega_i} \sum_{\omega_j \in \Omega \wedge i \neq j} P(\omega_j|x) \\ &= \arg\min_{\omega_i} 1 - P(\omega_i|x) \\ &= \arg\max_{\omega_i} P(\omega_i|x) \end{aligned}$$

Therefore the Bayes decision rule can be restated as follows

$$x \rightarrow \omega \qquad \text{if} \qquad P(\omega|x) = \max_{\omega_j} P(\omega_j|x) \tag{A.8}$$

Recall equation (2.2) and note that the denominator ($p(x)$) is independent of ω_i therefore the best decision rule is given by

$$\alpha(x) = \arg\max_{\omega_i} p(\omega_i|x) P(\omega), \qquad \forall x \in R^d \tag{A.9}$$

In the formulation of the Bayes decision rule it is assumed that all the class-conditional densities are defined. However, this assumption does not hold in practise and hence one needs to learn these parameters from the available training samples. Sometimes one can assume something about the form of the class-conditional density. Depending on the assumption taken we have parametric and nonparametric approaches for density estimation [Jain *et al.* 2000]. If we can assume that the class-conditional density have specific form but have unknown parameters, which needs to be estimated based on the sample, equation (2.2) will be used to calculate the posterior probability. In this case we have a parametric classification problem. Otherwise the posterior probability should be directly estimated on the feature space (training data) or alternatively construct the decision boundary on the feature space and thus resulting in a non-parametric

classifier.

Appendix B

SVM: Mathematical formulation

B.1 Linear support vector machines

The linear support vector machine also known as maximum margin classifier is the simplest form of Support Vector Machine. Hence this section will be used as an introduction to basic principles, notation and approaches that are later extended to a more general support vector machine. Furthermore, Support Vector Machines are inherently binary classifier. Therefore the formulation shown below and in subsequent sections is given for two class pattern recognition problem. In later sections we will see how this will be extended to multi-class pattern recognition problems. To make the presentation in two class classification problem simple and consistent with the conventional mathematical presentation we will use y_i instead of ω_i as class label. The material presented in this section is based on Vapnik [1995]; Burges [1998]; Osuna *et al.* [1997]; Cristianini and Shawe-Taylor [2000]; Schölkopf [1997]. If any other reference is used it will be cited accordingly.

B.1.1 Linear separable case - Maximum margin classifier

Given set of examples $((x_1, y_1), \ldots, (x_l, y_l)) \in \mathbb{R}^d \times \{\pm 1\}$, where $y_i \in \{\pm 1\}, x_i \in \mathbb{R}^d$ assume there exists a set of hyperplanes which totally discriminate the positive examples from the negative ones. This means we can find a pair (w, b) such that:

$$y_i(x_i.w + b) - 1 \geq 0 \qquad i = 1, 2, \ldots, l \tag{B.1}$$

where w is the direction of the normal or orientation of hyperplane and b is the threshold.

The mapping function which is usually called the hypothesis is then given by:

$$f(w, b) = sign(w.x_i + b) \tag{B.2}$$

Consider the example given in Figure 3.4. These training samples of the two classes (circle and square) can be perfectly separated by a linear hyperplane. Furthermore, one can find infinite number of hyperplane that can accomplish this task. Some of these hyperplane are shown in

Figure 3.4. As we can seen from the figure, each of these hyperplane have zero empirical risk, but we should find the one that will minimise the right hand side of equation (3.7) as seen in Section 3.2. From the figure one can take an educated guess to say the hyperplane that passes through the middle will be more likely to give the best minimum risk. Formally defining this hyperplane, the optimal hyperplane that is likely to minimise the expected risk is the one that maximises the margin, which is defined as the distance between the examples from the the opposite class that are close to this hyperplane.

Once the optimal hyperplane is found all the training sets will satisfy equations (B.1) and classification will be based on the sign of equation (B.2). The points that lie on the hyperplane separating the data satisfies:

$$w.x_i + b = 0 \tag{B.3}$$

Figure 3.5 gives graphical interpretation of Support Vector Classification. All the points that satisfy the inequality $w.x_i + b \geq \pm 1$ lie on H_1 or to the left of it and those satisfying $w.x_i + b \geq \pm 1$ will lie on H_2 or to the right of it. The margin is therefore defined as the distance between H_1 or H_2 and the optimal hyperplane (see figure 3.5). It is evident that H_1 and H_2 are parallel and for a perfectly separable training set, no point lies between the two hyperplane. Furthermore, H_1, H_2 and the optimal hyperplane differ only on the threshold b. Formally defining the optimal hyperplane with respect to these two hyperplanes, the separating hyperplane is optimal if the minimum distance between these hyperplane and the optimal hyperplane is maximal. i.e.

$$\text{if} \qquad \min\left\{ \min_{y_i=1}\left\{ \frac{w.x_i + b}{\parallel w \parallel} \right\}, \min_{y_i=-1}\left\{ \frac{-(w.x_i + b)}{\parallel w \parallel} \right\} \right\} \qquad \text{is maximal.}$$

To compute the threshold for a given w, let

$$d_+ = \min_{y_i=1}\{w.x_i\}, \qquad d_- = \min_{y_i=-1}\{-w.x_i\}$$

Substituting this values, then the optimal hyperplane is the one which maximises the equation given below:

$$\frac{1}{\parallel w \parallel} \min\{d_+ + b, d_- - b\}$$

From the above expression, the maximum will be attained when the two expressions inside the bracket are equal. Hence the threshold for the optimal hyperplane will be:

$$b_{opt} = \frac{d_- - d_+}{2}$$

and the respective margin will be:

$$\gamma = \frac{1}{\parallel w \parallel}|d_+ + b_{opt}| = \frac{1}{\parallel w \parallel}|d_- - b_{opt}|$$

$$\implies \gamma = \frac{d_+ + d_-}{2 \parallel w \parallel}$$

Therefore we can find the optimal hyperplane that maximises the margin by minimising $\parallel w \parallel$ subject to equation (B.1). i.e.

$$\begin{aligned} Minimise \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(x_i.w + b) - 1 \geq 0 \quad \forall_i \end{aligned}$$

Note that minimising $\|w\|$ is the same as minimising $\|w\|^2$. Minimising a quadratic function under a linear constraint formulated above is called quadratic program and can be solved to give the solution to the optimal hyperplane using Quadratic Programming (QP) optimisation [Cristianini and Shawe-Taylor 2000]. Solving this problem using the classical Lagrangian multipliers approach have a number of advantages [Cristianini and Shawe-Taylor 2000]. Firstly, this approach gives an alternative formulation of the original problem (dual form) which is easier to solve. Secondly, the dual form is not only easier to solve but also emphasises the importance of some training examples over the other, leading to a minimised but critical sample size and thirdly, the dual form makes generalisation beyond linear separable cases an easy task.

Introduce a dual vector of non-negative Lagrangian multiplier $\Lambda = (\alpha_1, \alpha_2, \ldots, \alpha_l)$ corresponding to each inequality constraint in (B.1) the Lagrangian function (see Appendix B.2) will be defined as:

$$L_P(w, b, \Lambda) = \frac{1}{2} \parallel w \parallel^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i.w + b) + \sum_{i=1} \alpha_i \tag{B.4}$$

The saddle point of the Lagrangian, which will be determined by minimising L_P with respect to w and b and maximise with respect to $\Lambda \geq 0$ is the solution to the optimisation problem. Minimising L_P with respect to w and b we have:

$$\begin{aligned} & \frac{\partial L_P(w, b, \Lambda)}{\partial w} = 0 \\ \implies & w = \sum_{i=1}^{l} \alpha_i y_i x_i \end{aligned} \tag{B.5}$$

$$\begin{aligned} & \frac{\partial L_P(w, b, \Lambda)}{\partial b} = 0 \\ \implies & \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned} \tag{B.6}$$

substituting equation (B.5) and (B.6) into (B.4) we have

$$L_D(w, b, \Lambda) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i.x_j) \tag{B.7}$$

The optimisation problem is now reduced to maximising L_D with respect to Λ constrained to (B.6) with the solution given by (B.5). i.e.

$$\text{Maximise} \qquad L_D(w, b, \Lambda) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i.x_j)$$

$$\text{subject to} \qquad \sum_{i=1}^{l} \alpha_i y_i = 0; \tag{B.8}$$

$$\Lambda \geq 0$$

The Karush-Kuhn-Tucker theorem (see Appendix B.2) of optimisation theory, which guarantees the existence of a solution to the optimisation problem shows that, at the saddle point all points satisfy the constraint (B.1) with strict equality. i.e.

$$\alpha_i (y_i (w.x_i + b) - 1) = 0 \qquad i = 1, \ldots, l \tag{B.9}$$

From this equation, the following two conditions need to be distinguished:

- If $\alpha_i = 0$, then $y_i (w.x_i + b) \geq 1$
- If $\alpha_i > 0$, then $y_i (w.x_i + b) = 1$

Recall that one of the advantages of using the Lagrangian function to solve the optimisation problem is expressing the importance of each pattern in the training set. Consider the value of α_i corresponding to each training pattern. Training patterns with $\alpha_i > 0$ will fall on the hyperplane H_1 or H_2 (see figure 3.5) and hence are critical in defining the decision boundary. Other training patterns with $\alpha_i = 0$ lies to the left or right of H_1 and H_2 respectively. These training patterns have no effect in determining the decision boundary. Therefore if those training patterns with $\alpha_i = 0$ value are removed and the training is repeated, the decision boundary will remain the same. Training patterns with nonzero α_i are called *Support Vectors* (The name of this learning technique follows from this).

Suppose the parameter set Λ^* solves the quadratic optimisation problem given in equation (B.8). Then the orientation of the optimal hyperplane $w*$ will given by equation (B.5). The geometric margin can be redefine in terms of Λ^* as:

$$\gamma = \frac{d_+ + d_-}{2 \parallel w^* \parallel}$$

for maximum margin hyperplane $d_+ = d_- = 1$, hence

$$\gamma = \frac{1}{\parallel w^* \parallel} = \left(\sqrt{(w^*.w^*)} \right)^{-1} \tag{B.10}$$

substituting w in (B.5) in (B.1) we have

$$y_i(\sum_j \alpha_j^* y_j^*(x_i.x_j) + b^*) = 1;$$

and substituting w from (B.5) in (B.10) we have

$$\begin{aligned}(w^*.w^*) &= \sum_{i,j} \alpha_i^* \alpha_j^* y_i y_j (x_i.x_j) \\ &= \sum_i \alpha_i^* y_i \sum_j \alpha_j^* y_j (x_i.x_j) \\ &= \sum_i \alpha_i^* (1 - y_i b^*) \\ &= \sum_i \alpha_i^* - \sum_i (\alpha_i^* y_i) b^*\end{aligned}$$

Using equation (B.6) the second term will be cancelled. Therefore,

$$\gamma = \left(\sqrt{\sum_i \alpha_i^*} \right)^{-1} \tag{B.11}$$

However, the threshold can not be computed directly from Λ^*. To compute the threshold b^* recall the optimal hyperplane with the maximum margin was defines as:

$$\begin{cases} \max_{y_i=-1}(w^*.x_i) + b^* = -1, \\ \max_{y_i=+1}(w^*.x_i) + b^* = +1 \end{cases} \quad \forall i = 1, 2, \ldots, l$$

Adding the two expressions and solving for the threshold of the optimal hyperplane b^* gives:

$$b^* = -\frac{\max_{y_i=-1}(w^*.x_i) + \max_{y_i=+1}(w^*.x_i)}{2} \tag{B.12}$$

With the weight vector w^* and the threshold b^* in place, substituting equation (B.5) into equation (B.2) the mapping function can be redefined as:

$$f(x, \Lambda^*, b^*) = sign(\sum_{i=1}^{l} y_i \alpha_i^* (x.x_i) + b^*) \quad \forall i = 1, \ldots, l \tag{B.13}$$

We have seen that the parameter $\alpha_i^* = 0$ for all training points except for the support vectors, hence the mapping function will have its final form:

$$f(x, \Lambda^*, b^*) = sign(\sum_{i \in \mathcal{SV}} y_i \alpha_i^* (x.x_i) + b^*) \quad \forall i = 1, \ldots, l \tag{B.14}$$

In other words the expression is evaluated in terms of the dot product between the pattern

to be classified and the Support Vectors (x_i), and the sign of the function is used to classify the pattern to their respective class. Summing up the dual approach in the case of linear separable examples the following proposition can be made:

Proposition B.1.1 *Consider a set of training data which can be separated into their respective class by a linear hyperplane:*

$$((x_1, y_1), \ldots, (x_l, y_l)) \in \mathbb{R}^n \times \{\pm 1\}$$

Let $\Lambda \in \mathbb{R}^l$ be a vector that solves the constrained optimisation problem given below:

$$\begin{aligned} \textit{Maximise} \quad & L_D(w, b, \Lambda) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i . x_j) \\ \textit{subject to} \quad & \sum_{i=1}^{l} \alpha_i y_i = 0; \\ & \Lambda \geq 0 \end{aligned}$$

Then the optimal hyperplane is defined by the pair w and b defined as:

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i$$

and

$$b = -\frac{1}{2}[\max_{y_i=-1}(w.x_i) + \max_{y_i=+1}(w.x_i)]$$

with the geometric margin given by:

$$\gamma = \left(\sqrt{\sum_{i \in \mathcal{SV}} \alpha_i} \right)^{-1}$$

And classification of unseen data will be done based on the sign of the function:

$$f(x, \alpha, b) = sign(\sum_{i \in \mathcal{SV}} y_i \alpha_i (x.x_i) + b) \qquad \forall i = 1, \ldots, l$$

B.1.2 Linearly non-separable case - Soft margin classifier

So far we have seen the case where the training data is perfectly separable using linear hyperplane but real-world problems involve non-separable data and the assumption taken in the previous section is too ambitious. To extend the above solution to non-separable data a positive slack variable $\xi_i; i = 1, \ldots, i$ is introduced to associate further cost as a penalty for misclassification whenever necessary (see Figure 3.6).

Using this relaxed separation constraint equation (B.1) becomes:

$$y_i(x_i.w + b) > 1 - \xi_i \quad \xi_i \geq 0 \quad i = 1, \ldots, l \tag{B.15}$$

The problem of finding optimal margin will therefore compromise two part.

- Maximise the margin (the same as the linear separable case) and
- Minimise the slack variable ξ_i which counts for amount of error

One way of combining these two conditions into a single function is given below:

$$\Phi(w, \Xi) = \frac{1}{2}\|w\|^2 + C(\sum_{i=1}^{l} \xi_i)^k$$

The constant C is a parameter to be freely chosen by the user to specify the trade-off between the width of the margin and misclassification penalty. Therefore the optimal hyperplane will be the one that minimises the function $\Phi(w, \Xi)$. i.e.

$$\begin{aligned} \text{Minimise} \quad \Phi(w, \Xi) &= \frac{1}{2}\|w\|^2 + C(\sum_{i=1}^{l} \xi_i)^k; \\ y_i(x_i.w + b) &> 1 - \xi_i \quad i = 1, \ldots, l; \\ \xi_i &> 0 \quad i = 1, \ldots, l \end{aligned} \tag{B.16}$$

If we choose $k = 1$, the above optimisation problem can be solved using QP. Introducing a dual vector of non-negative value $\Lambda = (\alpha_1, \alpha_2, \ldots \alpha_l)$ for of each the first constraint and $\Gamma = (\mu_1, \mu_2, \ldots, \mu_l)$ for each of the second constraint the Lagrangian representation will be:

$$L_P(w, b, \Lambda, \Xi, \Gamma) = \frac{1}{2} \parallel w \parallel^2 - \sum_{i=1}^{l} \alpha_i(y_i(x_i.w + b) - 1 + \xi_i) - \sum_{i=1}^{l} \mu_i \xi_i + C \sum_{i=1}^{l} \xi_i \tag{B.17}$$

Following the same approach as the separable case, the solution to the optimisation problem will be determined by the saddle point of the primal form of the Lagrangian which will be determined by minimising with respect to w, b and Ξ and maximised with respect to Λ and Γ. Minimising L_P with respect to w, b and Ξ gives:

$$\frac{\partial L_P(w, b, \Lambda, \Xi, \Gamma)}{\partial w} = w - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \tag{B.18}$$

$$\frac{\partial L_P(w, b, \Lambda, \Xi, \Gamma)}{\partial b} = \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{B.19}$$

$$\frac{\partial L_P(w, b, \Lambda, \Xi, \Gamma)}{\partial \Gamma} = C - \alpha_i - \mu_i = 0 \tag{B.20}$$

Substituting (B.18),(B.19) and (B.20) into (B.17) the dual from will still be given by:

$$L_D(w, b, \Lambda) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i.x_j) \tag{B.21}$$

The dual form is the same for both linear separable and non-separable cases (see equation (B.7) and (B.21)) respectively. But there are additional constraints to be satisfied in the later case. It is stated above that both the Lagrangian multipliers Λ and Γ should be non-negative. Although Γ does not appears in the dual form as a result of choosing k to be one, equation (B.20) places the constraint that $C = \alpha_i + \mu_i$. This condition limits the value of Λ to be less that C. Otherwise, if $\Lambda > C$, then $\Gamma < 0$ for the condition in equation (B.20) to hold but this a violation of the assumption that the Lagrangian multiplier Γ is non-negative. Therefore, the new optimisation problem will be redefined as:

$$\begin{aligned} \text{Maximise} \quad & L_D(w, b, \Lambda) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i.x_j) \\ \text{subject to} \quad & \sum_{i=1}^{l} \alpha_i y_i = 0; \\ & 0 \leq \Lambda \leq C \end{aligned} \tag{B.22}$$

Applying the KKT condition we have:

$$\alpha_i (y_i (x_i.w + b) - 1 + \xi_i) = 0 \qquad i = 1, 2, \ldots l \tag{B.23}$$

Following the above equation three different cases needs to be distinguished:

- If $\alpha_i = 0$, then $\mu_i = C$ $(\xi_i = 0)$ and $y_i(x_i.w + b) = 1$;
- If $0 \leq \alpha_i \leq C$, then $0 \leq \mu_i \leq C$ $(\xi_i = 0)$ and $y_i(x_i.w + b) = 1$;
- If $\alpha_i = C$, then $\mu_i = 0$ $(\xi_i > 0)$ and $y_i(x_i.w + b) = 1 + \xi_i$.

In the first case the points are on the correct side of the optimal hyperplane and are distant from the hyperplane by more than the margin γ (i.e. these points lie to the left or to the right of the hyperplane H_1 or H_2 respectively). In the second case, the points lie on the hyperplane H_1 or H_2 and are Support Vectors. In the third case these points are also Support Vectors, but does not necessarily lie on the hyperplane H_1 of H_2. These points might be on the wrong side of the hyperplane or on the right side but closer than the hyperplanes H_1 or H_2 (For example: X_1 and X_2 in Figure (3.6)).

Besides the above additional constraints, the solution for the linear separable case holds. Therefore the solution found in the previous section will be generalise by restating proposition B.1.1 as:

Proposition B.1.2 *Given a set of training data*

$$((x_1, y_1), \ldots, (x_l, y_l)) \in \mathbb{R}^n \times \{\pm 1\}$$

Let $\Lambda \in \mathbb{R}^l$ be a vector that solves the constrained optimisation problem given below:

$$\textit{Maximise} \quad L_D(w, b, \Lambda) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i.x_j)$$

$$\textit{subject to} \quad \sum_{i=1}^{l} \alpha_i y_i = 0;$$

$$0 \leq \Lambda \leq C$$

Then the optimal hyperplane is defined by the pair w and b defined as:

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i$$

and

$$b = -\frac{1}{2}[\max_{y_i=-1}(w.x_i) + \max_{y_i=+1}(w.x_i)]$$

with the geometric margin given by:

$$\gamma = \left(\sqrt{\sum_{i \in \mathcal{SV}} \alpha_i} \right)^{-1}$$

And classification of unseen data will be done based on the sign of the function:

$$f(x, \alpha, b) = sign(\sum_{i \in \mathcal{SV}} y_i \alpha_i^* (x.x_i) + b) \qquad \forall i = 1, \ldots, l$$

B.2 Important definitions and theorems

Definition B.2.1 *[Cristianini and Shawe-Taylor 2000] Given an optimisation problem with the objective function $f(w)$, and equality constraint $h_i(w)$, $i = 1, 2, \ldots l$, we define the Lagrangian function as*

$$L(w, \Lambda) = f(w) + \sum_{i}^{l} \alpha_i h_i(w)$$

where the coefficient α_i are called the Lagrangian multipliers.

Theorem B.2.2 *(Kuhn - Tucker) [Cristianini and Shawe-Taylor 2000] The point $\upsilon_0 \in \mathbb{R}$ minimises the function*

$$f : \mathbb{R}^N \to \mathbb{R}$$

subject to

$$g_i(\upsilon) \leq 0, \qquad i = 1, \ldots, l$$

where g_i are convex function and there is at least one point satisfying the constraint with strict inequalities, if and only if there exists a vector $\Lambda_0 \geq 0 \in \mathbb{R}^l$ with

$$L(\upsilon, \Lambda) = f(\upsilon) + \sum_{i}^{l} \alpha_i g_i(\upsilon)$$

has a saddle point at (υ_0, Λ_0), which is the minimum with respect to υ and a maximum with respect to Λ. The condition to be a maximum with respect to Λ is equivalent to the Kuhn-Tucker condition

$$\alpha_i g_i(\upsilon_0) = 0, \qquad \forall \alpha_i \in \Lambda_0$$

i.e. for each i either $\alpha_i = 0$, or $g_i(\upsilon_0) = 0$ The last relation is known as Karush-Kuhn-Tucker (KKT) complementarity condition.

Definition B.2.3 *[Cristianini and Shawe-Taylor 2000] A kernel $K(x, y)$ is a function such that for any two points (x, y) in the input space:*

$$K(x, y) = \phi(x).\phi(y) \tag{B.24}$$

where $x, y \in X$ and ϕ is the mapping $X \rightarrow F$.

Theorem B.2.4 *(Mercer's Theorem) [Burges 1998] There exists a mapping ϕ and an expansion*

$$K(x, y) = \sum_{i} \phi_i(x).\phi_i(y)$$

if and only if, for any $g(x)$ such that

$$\int g(x)^2 dx \qquad \textit{is finite}$$

then

$$\int K(x, y) g(x) g(y) dx dy \geq 0$$

Appendix C

Genetic codes

C.1 RNA codon table

Amino-acid Name	Code	codons coding it
Alanine (Ala)	A	GCU, GCC, GCA, GCG
Arginine(Arg)	R	CGU, CGC, CGA, CGG, AGA, AGG
Asparagine (Asn)	N	AAU, AAC
Aspartic acid (Asp)	D	GAU, GAC
Cysteine (Cys)	C	UGU, UGC
Glutamine (Gln)	Q	CAA, CAG
Glutamic acid (Glu)	E	GAA, GAG
Glycine (Gly)	G	GGU, GGC, GGA, GGG
Histidine (His)	H	CAU, CAC
Isoleucine (Ile)	I	AUU, AUC, AUA
Leucine (Leu)	L	UUA, UUG, CUU, CUC, CUA, CUG
Lysine (Lys)	K	AAA, AAG
Methionine (Met)	M	AUG
Phenylalanine(Phe)	F	UUU, UUC
Proline (Pro)	P	CCU, CCC, CCA, CCG
Serine (Ser)	S	UCU, UCC, UCA, UCG, AGU,AGC
Threonine (Thr)	T	ACU, ACC, ACA, ACG
Tryptophan (Trp)	W	UGG
Tyrosine (Tyr)	Y	UAU, UAC
Valine (Val)	V	GUU, GUC, GUA, GUG
Start		AUG, GUG
Stop		UAG, UGA, UAA

Table C.1: Standard amino acids used in proteins, and the codons that code for each amino acid.

C.2 The IUPAC nucleotides Codes

Nucleotide Code:	Base:
A	Adenine
C	Cytosine
G	Guanine
T (orU)	Thymine (or Uracil)
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base
. or -	gap

Table C.2: The IUPAC nucleotides Codes