University of Alberta

USING SNP DATA TO PREDICT RADIATION TOXICITY FOR PROSTATE CANCER PATIENTS

by

Farzaneh Mirzazadeh

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Spring 2010
Edmonton, Alberta

Examining Committee

Russell Greiner, Department of Computing Science

Rong-Cai Yang, Department of Agricultural, Food and Nutritional Science

Robert Holte, Department of Computing Science

To my loving parents.

Abstract

Radiotherapy is often used to treat prostate cancer. While using high dose of radiation does kill cancer cells, it can cause toxicity in healthy tissues for some patients. It would be best to apply this treatment only to patients who are likely to be immune from such toxicity. This requires a classifier that can predict, before treatment, which patients are likely to exhibit severe toxicity. Here, we explore ways to use certain genetic features, called Single Nucleotide Polymorphisms (SNPs), for this task.

This thesis uses several machine learning methods for learning such classifiers for predicting toxicity. This problem is challenging as there are a large number of features ($164,273$ SNPs) but only 82 samples. We explore an ensemble classification method for this problem, called Mixture Using Variance (MUV), which first learns several different base probabilistic classifiers, then for each query combines the responses of the different base classifiers based on their respective variances.

The original MUV learns the individual classifiers using bootstrap sampling of the training data; we modify this by considering different subsets of the features for each classifier. We derive a new combination rule for base classifiers in the proposed setting and obtain some new theoretical results. Based on characteristics of our task, we propose an approach that involves first clustering the features before selecting only a subset of features from each cluster for each base classifier.

Unfortunately, we were unable to predict radiation toxicity in prostate cancer patients using just the SNP values. However, our further experimental results reveal strong relation between correctness of a classifier in its prediction and the variance of the response to the corresponding classification query, which show that the main idea is promising.

Acknowledgements

First, I would like to sincerely thank my supervisor, Professor Russell Greiner, for his invaluable guidance, support, enthusiasm and patience throughout the period of this research. I am grateful to have the honor of working under his supervision.

I also gratefully acknowledge Professor Peter Hooper, from the department of Mathematical and Statistical Sciences, for his insight and expertise in the domain and the time he dedicated to our discussions. I thank Professors Robert Holte and Rong-Cai Yang for a careful reading of the dissertation, and the helpful comments that improved the thesis.

Next, I would like to thank my colleagues at AICML lab. I appreciate Nasimeh Asgarian and Bret Hoehn for patiently answering my questions in the duration of this thesis. I would like to thank Dr. Barnabas Poczos for his helpful comments and support.

I am very grateful to Drs S. Damaraju, M. Parliament, and B. Sehrawat of the Cross Cancer Institute, Edmonton, Alberta, for generously providing access to the SNP data, initial analysis and patient clinical characteristics, for meeting with us during the project, and for helping to guide the plan of work. Both their data and their insights were essential and invaluable to help shape the thesis objectives.

Finally, I wish to sincerely thank my husband, for his patience and support in the duration of this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The phenomenal rate of production of biological data has made the use of computers and computational methods essential in biological research [LGG01]. As a result, bioinformatics as a multidisciplinary field dealing with the topic has turned into a flourishing area of today's research. Bioinformatics is the application of computational methods in solving problems in the field of molecular biology [LGG01].

Computer scientists use data mining methods to extract the large amount of information embedded in human DNA, protein sequences, etc. and use this knowledge to solve some problems in biology or medicine. One type of problem is to automatically predict some behavior of a patient based on past history of other patients. For example, using machine learning methods, bioinformaticians try to predict if a patient is at high risk for a cancer using his/her genes. This problem is an instance of a classification problem.

Classification is a machine learning method that tries to use past experience to predict some discrete property, like sickness versus healthiness of a new object using a classifier. The property to be predicted is called the "class". For each new object, the goal is to predict the correct label for the class from the set of possible choices.

Past experience is introduced by training data that includes a set of n objects along with their class labels. Each object in a classification task is called a sample or a data point [Lus07]. Samples are typically defined by a set of properties. Each property is called a "feature" or an "attribute". Feature values can be discrete or continuous. Each feature can correspond to some property of the samples like

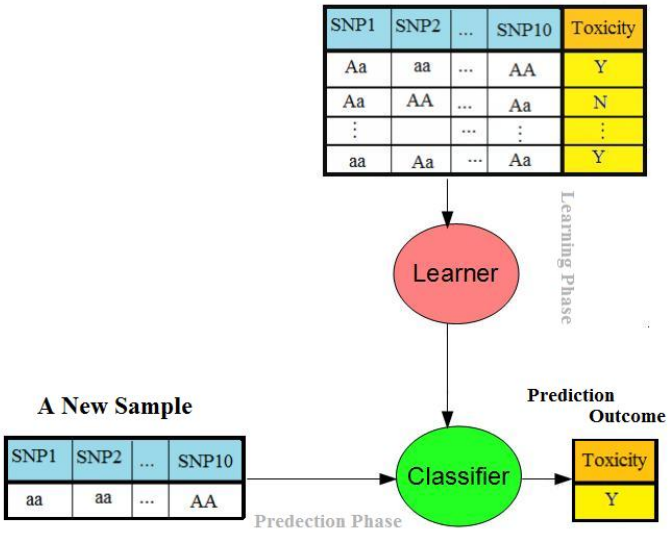

Figure 1.1: Learning and predicting phases in a classification procedure.

weight, height, ethnicity or the value of one of the genes.

We use a learning algorithm to produce a classifier, which can then be used to predict the label for new data based on past data. Therefor, classification is a two phase procedure [Lus07]:

1. Learning the model from training data.

2. Predicting the label of a new sample based on the learned model.

Figure 1.1 shows an example of the process of classification for predicting toxicity as a side effect of some treatment for different patients. In this example, each patient is presented by a feature vector (SNP_1, ..., SNP_{10}). Possible values for each feature are $\{$aa, Aa, AA$\}$. First, a classifier is learned on the labeled training data. Next, in performance time, the label for a new patient is predicted based on its feature values and the learned model.

The learning phase uses a training dataset of n samples and p features viewed as an $n \times p$ matrix X. Each row corresponds to one sample, while each column corresponds to one feature. An $n \times 1$ vector y contains the class labels for X.

1.1 The Problem

The problem we are working on in this thesis is a classification problem. We are dealing with prostate cancer patients. One major treatment for this disease is radiation therapy (aka radiotherapy). Unfortunately, this method can cause severe toxicity in some patients, but not all of them. Our goal is to predict whether a prostate cancer patient will suffer from toxicity after radiation therapy or not. Several researchers believe different reactions against radiotherapy may be related to genetic differences between people (for example, see [DMD$^+$06]). By this we mean that even with almost same conditions of treatment like intensity of radiation and the same clinical features like age, weight, etc., different people react differently to radiation. In this thesis, we study the correctness of this belief using a relevant dataset.

The dataset we received from our collaborators at Cross Cancer Institute (CCI) [1] contains genetic information of 82 prostate cancer patients along with their 70 clinical features. The genetic information we use are Single Nucleotide Polymorphisms (SNPs), which we discuss in Section 2.1. It consists of $164,273$ features. Each feature can have one of the values in $\{AA, Aa, aa\}$. Class labels are binary and show what really happened in the past for these patients after radiotherapy. Label "1" means the patient experienced toxicity — i.e. suffered from extensive bleeding after 90 days in at least one of these visits — , while "0" means he did not. These class labels are generated manually by studying the recorded status of these patients in several physician visits.

In typical classification problems, the number of samples n is much more than the number of features p. However, in many classification problems that originate from bioinfomatics the case is opposite: the number of features is much more than the number of samples. In our case, the reason this happens is that genetic data usually comes in very large amounts. In contrast, it is very expensive to gather data of a number of patients comparable with number of samples. Moreover, unfortunately

[1] Genotype and prostate cancer data were generated through independent grant funding (by Alberta Cancer Board) to Drs S. Damaraju and M. Parliament, University of Alberta and Cross Cancer Institute, Edmonton.

many patients do not consent to releasing their personal data. In our problem, the number of features is around 2000 times the number of samples. This "large p, small n" problem, makes the classification task difficult [HTF01]. Different methods are proposed to reduce dimensionality of data by implicitly or explicitly using a subset of features with a more tractable size. (See Section 3.1.)

1.2 Overview of Dissertation

In this dissertation we study different classification methods, and how well they can solve our problem. The rest of this document continues as follows: Chapter 2 presents the biological background behind our topic. In this chapter, we explain the biological meaning of SNPs and discuss prostate cancer toxicity issues.

The computational background of this thesis comes in Chapter 3, which reviews some feature selection, classification and clustering methods. Section 3.2.5 discusses a classification method called MUV in detail. This method uses a pool of classifiers for classification and is the major classifier that we explored in our thesis, both theoretically and experimentally.

In Chapter 4, we present our new version of MUV called FMUV. This chapter focuses on theoretical analysis of FMUV as well as the algorithm for assigning different features to each of the classifiers in the FMUV pool. In Chapter 5, we present our experimental results. We use different classifiers on our dataset and in this chapter we report their performance. Moreover, we evaluate performance of FMUV classifier and compare it with performance of original version of MUV on benchmark datasets. Also, we perform some experiments to show the correctness of the motivation behind MUV idea. Finally, our discussion and conclusion along with suggestions for future improvements come in Chapter 6.

Chapter 2

Biological Background

A gene is the physical entity transmitted from parent to offspring in reproduction that influences hereditary traits. From biochemical point of view, a gene corresponds to a region along a molecule of DNA (Deoxyribonucleic Acid). DNA is the genetic material in organisms. Genes can exist in different forms or states. For example, a gene for hemoglobin may exist in different forms that result in hemoglobin molecules that are more or less abnormal. These alternative forms of genes are called alleles. A nucleotide is the basic building block of DNAs appearing in four types: Adenine, Cytosine, Guanine, and Thymine denoted by letters A, C, G, and T respectively [HC89].

People differ from each other genetically. Section 2.1 introduces the major source of this difference. As a result of genetic differences, patients react differently to some treatments. In particular, prostate cancer patients react differently to one of their major treatments: radiotherapy. It can have genetic reasons. We discuss this problem in Section 2.2.

2.1 Single-nucleotide Polymorphisms

Single-nucleotide Polymorphisms (SNPs, pronounced snips) are the most common source of genetic diversity in members of different species. It occurs when two alternative nucleotides are possible in a locus in members of species. A locus is a specific location of a gene or DNA sequence on a chromosome. SNP variation occurs when a single nucleotide, such as an A, is replaced by one of the other three

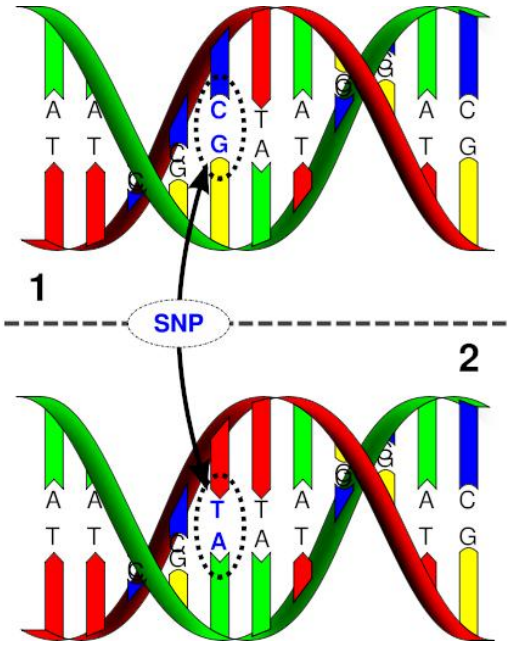

Figure 2.1: A Single Nucleotide Polymorphism (SNP) and two alleles. (Picture taken from [SNP])

nucleotides C, G, or T. Figure 2.1 shows a SNP and its two alleles.

In the cells of most organisms that reproduce sexually, one chromosome is inherited from each of the parents and as a result chromosomes occur in pairs. The two chromosomes of each pair contain genes that correspond to the same inherited traits. Equivalently, for each genetic trait in an organism there are two responsible genes. If both of these genes are the same, the organism is called "homozygous" for that trait [HC89]. Otherwise, the organism is called "heterozygous" for that trait. Figure 2.2 shows a locus for flower color gene along with two different alleles for that locus corresponding to white and purple colors in a flower.

Normally, for each SNP with two alleles, one allele is more frequent than the other in the members of species, which we call the major allele and denote with "A". The minor allele denoted by "a" is the allele with lower frequency in the members of species. Each child inherits one chromosome from each of the parents, which can be major or minor. However, looking at a child's chromosomes, it is not possible to distinguish between chromosomes from each of the parents from the child's chromosome, i.e., mother="A" and father="a" is the same as mother="a" and father="A". So each SNP can have three possible values: "AA" major homozygous,

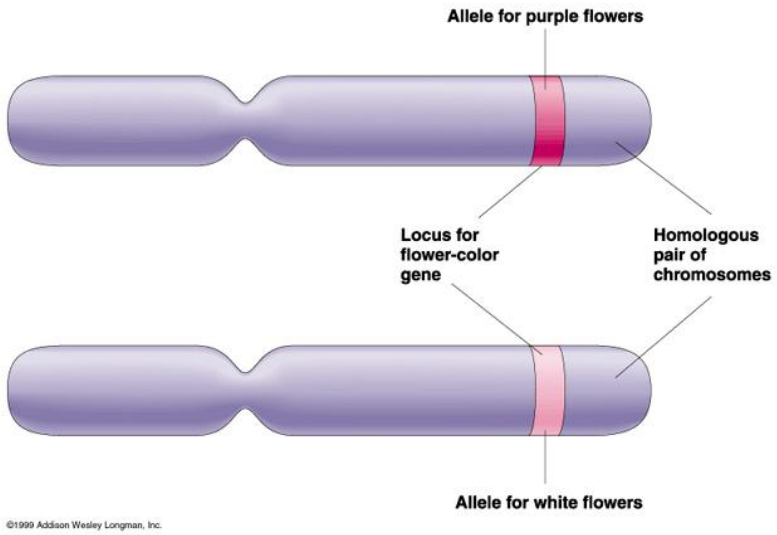

Figure 2.2: Alleles for a Locus Corresponding to a Flower Color. (Picture taken from [All])

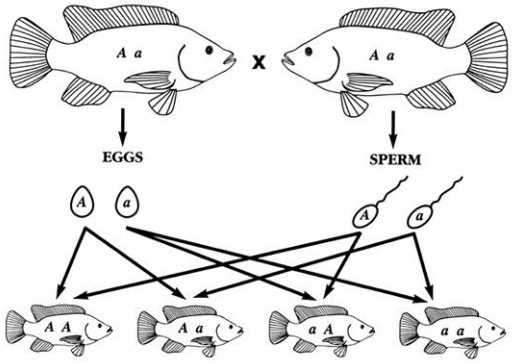

Figure 2.3: Parents with heterozygous alleles can have children with homozygous major, homozygous minor, or heterozygous alleles. (Picture taken from [zyg])

"aa" minor homozygous, and "Aa" heterozygous. If there are only two alleles and the frequency of alleles "A" and "a" in population are p and $1-p$ respectively, then genotype frequencies of "AA", "aa" and "Aa" will be p^2, $(1-p)^2$ and $2p(1-p)$ respectively. Figure 2.3 shows different possible alleles for a child with parents with a heterozygous trait in a fish population.

Sometimes the alleles appearing in different loci in genes are not independent- i.e. some alleles are observed together more than expected. A combination of alleles at multiple loci that are transmitted together on the same chromosome is called a "haplotype" [HC89]. Association of alleles of genes in different loci is called

"linkage disequilibrium". To measure linkage disequilibrium a quantity D is defined as the departure of the real haplotype frequency $P_{a,b}(x,y) = P(SNP_a = x, SNP_b = y)$ from the simple product of their individual component allele frequencies $P_a(x) = P(SNP_a = x)$ and $P_b(x) = P(SNP_b = x)$ [HC89]:

$$D(x,y) = P_{a,b}(x,y) - P_a(x)P_b(y). \tag{2.1}$$

If the alleles x and y are independently transmitted, their haplotype frequency $P_{a,b}(x,y) = P_a(x)P_b(y)$ and consequently $D(x,y) = 0$. Larger values of $D(x,y)$ correspond to stronger linkage between the alleles.

Knowing about this statistical association is helpful in many bioinformatics studies like those that try to find markers for different phenotypes, classification, etc. The reason is that it can help in reducing the dimension of the problem in hand. Dimension reduction can be useful when working with SNPs because the number of SNPs is large enough to make the problem difficult. There are some worldwide efforts to find out and document such useful information. If the plan is to use biological background knowledge in a project dealing with finding biomarkers, these sources are essential.

The "International HapMap Project" is a project that tries to identify linkage disequilibrium in the human genome [Con]. This project is a "multi-country effort to identify and catalog genetic similarities and differences in human beings. Using the information in the HapMap, researchers will be able to find genes that affect health, disease, and individual responses to medications and environmental factors. The Project is a collaboration among scientists and funding agencies from Japan, the United Kingdom, Canada, China, Nigeria, and the United States." [1]

2.2 Prostate Cancer Toxicity

Prostate cancer is one of the most common cancers in men. Particularly,it is the most common cancer in Canadian men [MDD+09]. It occurs in the prostate, a small gland in the male reproductive system. It usually grows slowly and can often

[1] http://hapmap.ncbi.nlm.nih.gov/thehapmap.html.en

be cured or managed successfully. Some treatment options are surgery, radiation therapy, hormone therapy, and chemotherapy.

Radiotherapists use high energy waves to treat this cancer. The use of radiotherapy to cure early prostate cancer can be internal, called "brachytherapy", or external called "external beam radiotherapy". External radiation therapy uses a machine outside the body to send radiation toward the cancer. Internal radiation therapy, on the other hand, uses a sealed radioactive substance placed directly into or near the cancer.

Three-dimensional conformal radiotherapy (3D-CRT) is an image guided method for external beam radiotherapy. This process begins with the creation of three dimensional digital data sets of patient tumors. These data sets are then used to generate 3D computer images and to develop complex plans to deliver highly focused radiation while sparing normal adjacent tissue. In 3D-CRT, different locations for positioning the source of external radiation are used that all concentrate on the cancer cells. In this way, higher doses of radiation can be delivered to cancer cells while the amount of radiation received by surrounding healthy tissues is not increased. This technique should increase the rate of tumor control while decreasing side effects. However, late rectal bleeding is a dose limiting complication of external beam radiotherapy [JSZ$^+$01].

Dose escalation is reported to improve biochemical freedom from relapse after 78 versus 70 Gray dose of radiation for localized prostate cancer [DMD$^+$06]. The worry is that high-dose treatments should be limited by the radiation toxicity capacity of the patient. Identifying people potentially at high risk of late radiation toxicity can be very useful. Damarju et al. [DMD$^+$06] also report from [PZS$^+$02] a decrease in risk of radiation-related rectal complications from 26% to 12% when using 70 Gray instead of 78 Gray dose of radiation. They assume this different reaction has genetic reasons and try to find association between it and some SNPs. They also assume the genes related to "repair" mechanism in human beings should be most related to toxicity and looks at SNPs on those genes. What we do in this thesis is related to the work done in that paper. However, our problem is a classification problem instead of association and we are using a different set of SNPs.

Chapter 3

Computational Background

In this chapter we explain some machine learning terms and methods that we use frequently in the next chapters of this thesis. We discuss feature selection methods, classification methods, and clustering. Since Section 3.2.5 is highly related to our analysis and methods in Chapter 4, we explain it in detail.

3.1 Feature Selection

We use "feature selection" to mean constructing and selecting subsets of features that are useful in building a good classifier [Guy03]. In past, many papers published on feature selection (like [BL97, KJ97]) explore domains with no more than 40 features. However, now many bioinformatics and text categorization problems deal with hundreds of thousands of features [Guy03]. Several methods are proposed for solving this challenging problem. We review some of them here. We follow the categorization used in [Guy03].

3.1.1 Feature Ranking

In this set of methods, a criterion is used to rank each single feature. Then a number of features with highest rank are selected. Many algorithms use this method because it is simple, scalable, has shown good empirical success [Guy03]. Sometimes the number of features is so large that this method is the only practical initial phase. However, with this method, it is possible that many relevant but correlated and redundant features are selected, which is not desired. Two types of criteria often

used for feature selection are correlation-based and information-theoretic criteria.

Correlation-based Feature Selection

For two vectors X and Y, the Pearson correlation coefficient, $Corr(X, Y)$ also denoted by ρ is defined as below [Guy03]:

$$\rho(X, Y) = Corr(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}} \tag{3.1}$$

where the covariance $Cov(X, Y)$ is defined as $Cov(X, Y) = E(XY) - E(X)E(Y)$ and the variance $Var(X)$ is defined as $Var(X) = E(X^2) - E^2(X)$. This coefficient is also equal to the cosine between vectors X and Y after each has been centered (its mean subtracted from each value). This correlation coefficient value is in the interval $[-1, +1]$ that attempts to quantify the linear dependencies between variable and target.

Information Theoretic Feature Selection

"Mutual Information" is a quantity that measures the mutual dependence of the two variables. The mutual information of two discrete random variables X and Y denoted as $MutualInfo(X, Y)$ is computed as below [Guy03]:

$$MutualInfo(X, Y) = \sum_{x \in X} \sum_{y \in Y} P_{XY}(x, y) \log(\frac{P_{XY}(x, y)}{P_X(x)P_Y(y)}) \tag{3.2}$$

where joint probability $P_{XY}(x, y)$ and marginal probabilities $P_X(x)$ and $P_Y(y)$ are estimated by frequencies using the dataset.

"Information Gain" is another information theoretic criterion. It is a measure of the effectiveness of a feature in classifying a dataset that is equal to the expected reduction in entropy caused by partitioning features according to this feature. More precisely, the information gain, $InfoGain(S, F)$ of a feature F relative to a collection of samples S, is defined as:

$$InfoGain(S, F) = Entropy(S) - \sum_{v \in Values(F)} P(v)Entropy(S_v) \tag{3.3}$$

where $Values(F)$ is the set of all possible values for feature F, P_v is estimated from data so that $P(v) = \frac{|S_v|}{|S|}$ and S_v is the subset of S for which feature F has value v and

$$Entropy(S) = \sum_{i=1}^{c} -P_i \log(P_i).$$

P_i is the portion of S belonging to class i.

Information gain is the information provided about target function value given the value of some feature F [Mit97].

Selection Using Single Variable Classifier

Finally, the idea of this method is simply to select features according to their individual predictive power using as criterion the performance (like error rate) of a classifier built with a single variable [Guy03].

3.1.2 Feature Subset Selection

The motivation behind this method of feature selection is that a variable that is completely useless by itself can sometimes provide a significant performance improvement when taken with others. Even two variables that are useless by themselves can be useful together [Guy03], e.g. suppose we want to learn a classifier for binary XOR operator. In this example, the best classifier built on each of the features has 50% accuracy, while a classifier built on both features can have 100% accuracy. By feature subset selection, complementary useful features can be selected instead of redundant useful features. We explore some methods below:

Wrapper and Embedded Methods

The wrapper methodology consists of using prediction performance of a classifier to assess the relative usefulness of subsets of variables. In practice, a method for searching the space of all possible features is needed. Exhaustive search is not effective because the problem is known to be NP-hard. Seeking suboptimal subsets of features, a wide range of search strategies can be used including best-first,

branch and bound, simulated annealing, and genetic algorithms. Some classification methods like CART decision tree learning [BFOS84] have a built-in feature selection mechanism inside. We call them "Embedded Feature Selection Methods" after [Guy03]. These methods may be more efficient than wrapper methods in several respects: (1) They make better use of available data by not needing to split the training data into a training and validation set. (2) They reach a solution faster as they do not need to retrain a classifier for every subset of features investigated [Guy03].

Direct Objective Optimization

Many researchers have attempted to formalize an objective function for variable selection and find algorithms to optimize it. Generally the objective function consists of two terms:

1. the goodness of fit
2. the number of features

where the goal is to maximize the former and minimize the latter.

These methods also embed a feature selection procedure inside. Lasso is an example of these methods [Tib96].

3.2 Classification Methods

In this section, we explain popular classification techniques that we use in our experimental study. Moreover, in Section 3.2.5 we describe a classification method called Mixture Using Variance [LGW06] and the method of computing the variance of response of a classifier to a query [AGH01] in detail. All our analysis and methods in Chapter 4 is related to this section.

One term that we use a lot in the rest of this chapter is overfitting. Overfitting happens when a learned model classifies the training data accurately, but not the test data. This usually means that the model is fitted too much to the small perturbations in data, which is typically related to noise rather the general pattern of data [HTF01].

In all methods below, we represent the training dataset with n samples and p features as an $n \times p$ matrix X. Each row corresponds to one sample, while each column corresponds to one feature. To refer to the i^{th} sample, we use the notation x_i, a $p \times 1$ vector defined as the transpose of i^{th} row of matrix X. An $n \times 1$ vector y contains the class labels for X. Features are represented by $\{F_1, ..., F_p\}$.

3.2.1 Support Vector Machines

Support vector machines [Bur98] are one of today's most powerful classifiers. In their simplest form, they can be used for linear classification. These classifiers try to find a separating hyperplane with maximum margin between the classes. Moreover, using kernel functions they can perform nonlinear classification as well.

Assume each data point $x_i \in \Re^n$ is an $n \times 1$ vector with binary class label $y_i \in \{-1, 1\}$. The points belonging to two classes are linearly separable then a separating hyperplane can be presented as $w^T x + b = 0$ where w is an $n \times 1$ vector normal to the hyperplane and b is a scalar. Let d^+ and d^- be the shortest distance from the separating hyperplane to the closest positive and negative samples respectively. Our goal is to minimize $d = d^+ + d^-$, which is called the margin. If we represent margin boundaries with equations

$$B_1 = \{x| \quad w^T x + b = +1\} \tag{3.4}$$

$$B_2 = \{x| \quad w^T x + b = -1\} \tag{3.5}$$

the margin would be equal to the distance between these parallel hyperplanes:

$$d = \frac{2}{\|w\|}$$

Hence to maximize the margin, we need to minimize the Euclidean norm $\|w\|$ or an increasing function of it such as $\frac{\|w\|^2}{2} = \frac{w^T w}{2}$.

If the line is to be the separator of the members of two classes, any training point x_i with class value $y_i = +1$ should have $w^T x_i + b \geq 1$ and any training point with $y_i = -1$ should have $w^T x_i + b \leq -1$, (see Figure 3.1).

In summary, the goal is to minimize

$$\frac{w^T w}{2} \tag{3.6}$$

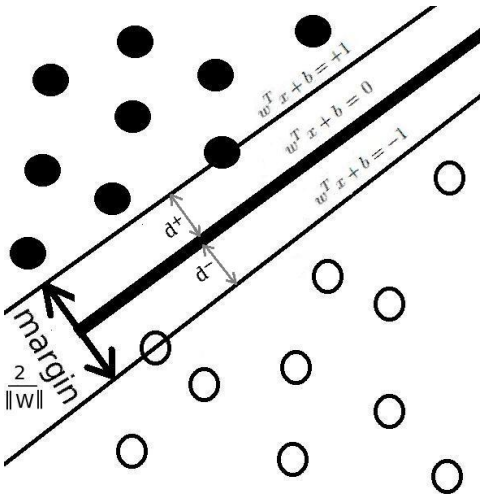

Figure 3.1: Separating Hyperplane in SVM. Positive samples are shown filled in, while negative samples are empty. The width of margin is $\frac{2}{\|w\|}$

subject to

$$\forall i, y_i(w^T x_i + b) \geq 1$$

This constrained optimization problem can be solved using Lagrange multipliers. The solution is the classifier we are seeking.

A version of SVM with soft margins addresses the case when samples are not linearly separable. In this version, misclassification for some of the samples is allowed. However, we change the optimization function so that it minimizes a linear combination of misclassification and margin width simultaneously. The degree of misclassification for each sample is defined as

$$\xi_i = 1 - y_i(w^T x_i + b) \tag{3.7}$$

Here the goal is to minimize

$$\frac{w^T w}{2} + C \sum_i \xi_i \tag{3.8}$$

subject to constraints

$$\forall i, y_i(w^T x_i + b) > 1$$

$$\forall i, \xi_i \geq 0$$

In Equation 3.8, C is a user-specified constant. Again the solution is the classifier we are seeking.

3.2.2 Decision Trees

Decision trees are tree structures successfully used in solving classification and regression problems. In a decision tree used for classification, features appear in interior nodes and target class values appear in leaves. The edges leaving each node are associated with feature values corresponding to that variable. A decision tree classifies a data point by sorting it out through the tree from root to the appropriate leaf node, then returning the class associated with that leaf [Mit97].

There are different machine learning methods for learning a decision tree from data. Based on the algorithms used for training, the decision tree will have different names like ID3, C4.5 and CART trees. These algorithms work by recursively choosing the best feature among available features and putting it in the root of the tree. Then by omitting this feature from available set, a new subset of features is composed. For each value of the selected feature, a branch is added to this root and the subset of samples with the corresponding feature value is selected for the next step. Then, the process recursively continues for each subtree and the new subset of features and samples.

All methods try to choose features that best split the set of data samples. Different algorithms use different criteria for this selection. ID3 and C4.5 use the information gain criterion (See Section 3.1). Algorithm 1, adapted from [Mit97], describes the ID3 learning. In this algorithm, *trainingSamples* is the set of all training samples, *targetFeature* is the class feature whose value is to be predicted, and *Features* is the list of other features that maybe tested by the learned decision tree. This algorithm returns a decision tree *Root* that correctly classifies the given *trainingSamples*.

Figure 3.2 shows a well known example of a decision tree. This tree classifies the possibility of playing tennis in different weather conditions. In a decision tree like this, an equivalent way of representing decision trees is by a set of rules. Each path from root of the tree to a leaf corresponds to a rule. For example in Figure 3.2, "If the outlook is rainy and the wind is weak, it is possible to play tennis" is a rule.

C4.5 is an improved version of ID3, proposed by the same author. C4.5 performs a pruning procedure after creating the tree, which attempts to remove branches

Algorithm 1 $Root = ID3(trainingSamples, targetFeature, Features)$

Create a node for the tree. Call it $Root$
if all trainingSamples are positive **then**
  Return the single-node tree $Root$, with label "+" .
else if all trainingSamples are negative **then**
  Return the single-node tree $Root$, with label "-" .
else if $Features = \emptyset$ **then**
  Return the single-node tree $Root$, with label = most common value of $targetFeature$ in $trainingSamples$
else
  F $\leftarrow$ the feature in $Features$, that best classifies $trainingSamples$
  The decision feature for $Root \leftarrow$ F
  for each possible value, v_i of F **do**
    Add a new tree branch below $Root$, corresponding to the test $F = v_i$.
    Let $Samples_{v_i}$ be the subset of $trainingSamples$ that have value v_i for F
    if $Samples_{v_i} = \emptyset$ **then**
      below this new branch add a leaf node, with label = most common value of $targetFeature$ in $trainingSamples$
    else
      below this new branch add the subtree:
      $ID3(Samples_{v_i}, targetFeature, Features - \{F\})$
    end if
  end for
end if
Return $Root$

that do not help in classification. This procedure makes the tree simpler and helps to prevent overfitting. In addition, in C4.5 both continuous and discrete features can be handled while ID3 can only handle discrete features. C4.5 allows missing values and different cost functions for attributes as well.

3.2.3 Bayesian Networks

Bayesian networks (aka belief networks) [Pea88] are directed acyclic graphs (DAGs) that allow efficient representation of the probability distribution over a finite set of random variables or a set of parameters. Each node v in the graph represents a random variable, and each directed edge e represents dependencies between the two variables it connects. This dependency states that each variable X_v is independent of all its nondescendants in the graph given the state of its set of parents $Pa(X_v)$. Probabilistic parameters are encoded in a set of tables called CPtables, one for each

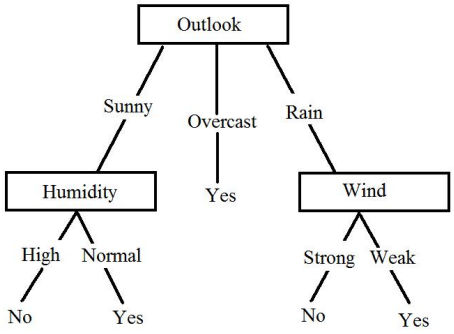

Figure 3.2: A Well-known example of a decision tree. This tree shows the possibility of playing tennis in different weather conditions. (Picture from [Mit97] adapted from [Qui86].)

variable, that represents local conditional distribution of a variable given its parents [FGG+97]. Figure 3.3, adapted from [AGH01], shows an example of a Bayesian network and its CPtables, where CPtable entries are constant.

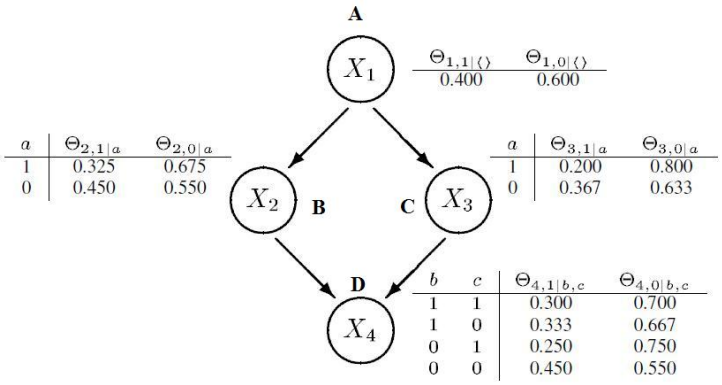

Figure 3.3: An example of Bayesian network and its CPtables with constant entries (from [AGH01])

There are many different methods proposed for learning a Bayesian network from data. The objective is to induce a network that best describes the probability distribution over the training data. As usual, there are different criteria for determining the best network. Usually this is a two-step procedure; first learning the structure of the network, then learning the parameters for this structure.

After learning the network structure and CPtable entries, we can use the resulting model for classification of new data. Given a set of features F_1,..., F_n

and the corresponding valid feature values f_1,..., f_n a classification algorithm returns the class label c that maximizes the posterior probability $P(Class = c|F_1 = f_1, ..., F_n = f_n)$.

A special case of Bayesian networks is naive Bayes classifier. Naive Bayes Classifiers use a two level DAG structure, with the class variable at the root and all features in the next level (Figure 3.4(a)). Directed arcs are connected from class feature to other features. This structure encodes a strong independence assumption that all the features are independent given the value of the class. Although this conditional independence assumption is unrealistic, naive Bayes classifier performs surprisingly well.

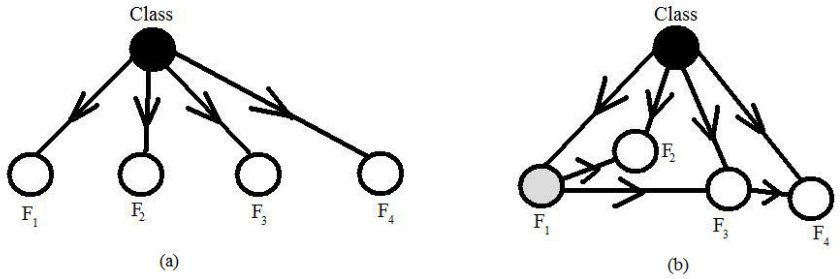

Figure 3.4: (a) A Sample Naive Bayes Classifier. f_1 to f_4 are the features. (b) A Sample TAN. f_1 to f_4 are the features. Tree structure can be seen among the features. The Root of the feature tree, f_1 ,is shown in Grey.

To relax this unrealistic assumption, other structures with some modifications are proposed [FGG$^+$97], such as Tree Augmented Naive Bayes (TAN). Like naive Bayes,the TAN class variable has no parents. However, each feature has as parents the class variable and at most one other feature. In TAN after omitting the class variable and all of its edges, the remaining structure is a directed tree (Figure 3.4(b)).

3.2.4 Lasso

Lasso [Tib96] fits a linear model

$$\hat{y_i} = b_0 + b_1 X_{i1} + ... + b_p X_{ip} \tag{3.9}$$

to data where b_0 to b_p are scalars.

The goal here is to minimize

$$\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{3.10}$$

subject to

$$\sum_{j=0}^{p}|b_j| \leq s$$

where s is a tuning parameter that can be found using cross-validation.

We can center data points by subtracting their means, so that they have zero mean. Then instead of Equation 3.10 we can minimize:

$$\sum_{i=1}^{n}(y_i - \hat{y_i})^2 + \lambda\sum_{j=1}^{p}|b_j| \tag{3.11}$$

Lasso is originally proposed for regression where the y_is can have any real value. However, we use it for classification, where $y_i \in \{-1, +1\}$. Any positive prediction for a sample is considered as "+1" and any negative prediction as "-1".

The solution to Lasso can be found using quadratic programming. However, there is no analytic solution for it. Efron et al. [EHJT04] proposed an efficient algorithm Least Angle Regression (LARS) to solve the problem. Algorithm 2, presents the LARS algorithm (from [EHJT04]). This algorithms inputs feature vectors $X_1, ..., X_p$ and target vector y and returns a linear model b.

3.2.5 Mixture Using Variance

Mixture using variance (MUV) is a query-specific ensemble classification method [LGW06]. For each classification query (like $P(Class = c_1|F_1 = f_1, ..., F_n = f_n)$ which asks about the probability that the class is c_1 given the feature vector $(f_1, ..., f_n)$), each classifier provides its response μ_i along with the variance σ_i^2 around this response. The variance is interpreted as a measure of uncertainty in the classifier response. The final response of the ensemble classifier is a weighted sum of these individual responses using weights in $[0, 1]$ that are inversely proportional

Algorithm 2 b = LARS ($X_1, ...X_p, y$)

for all j= 0:p **do**
  $b_j \leftarrow 0$.
end for
Find the predictor x_j most correlated with y
Repeat
  Repeat
    Increase the coefficient b_j in the direction of the sign of its correlation with y.
    Take residuals $r = y - \hat{y}$.
  Until some other predictor x_k has as much correlation with r as x_j has.
  Repeat
    Increase (b_j, b_k) in their joint least squares direction,
  Until some other predictor x_m has as much correlation with the residual r.
Until all predictors are in the model

to their respective variance around the response:

$$P(c|e) = \frac{1}{\sum_i \frac{1}{\sigma_i^2(c|e)}} \sum_i \frac{\mu_i(c|e)}{\sigma_i^2(c|e)} \tag{3.12}$$

where $\mu_i(c|e)$ is the response of the ith classifier to the query $p(c|e)$ and $\sigma_i(c|e)$ is the variance of response of the i^{th} classifier to this query. c is the class and e is the evidence, which here means some valid assignment of feature values to features.

Although MUV can be used for combining any type of classifiers whose variance is computable, it is originally proposed for Bayesian networks and specifically Naive Bayes and Tree Augmented Naive Bayes (TAN) classifiers. When the parameters of a network are assumed to be random variables, the randomness in parameters induces randomness in the response of the network that is a function of those parameters. Using the Delta method [BFH75], Van Allen et al. [AGH01] computed the asymptotic variance the around response to a query of a Bayesian network assuming the structure of network is correct and network parameters follow independent Dirichlet distribution. We will discuss it later in this section. Here we only focus on the main derivation of the MUV combination rule.

The original MUV learns the individual classifiers using bootstrap sampling of the training data. The basic idea behind bootstrap sampling is to randomly draw datasets with replacement from the training data, each sample the same size as the original training set [HTF01]. To derive the combination rule for this version

of MUV, let $\mu = P(c|e)$ be the true value for a response to a query. Each base classifier i responds to this query with a noisy version of μ called μ_i. The noise is assumed to be normal with mean 0 and variance σ_i^2:

$$\mu_i = \mu + \epsilon_i \tag{3.13}$$

where

$$\epsilon_i \sim N(0, \sigma_i^2).$$

The goal is to estimate μ using a linear combination of μ_is. Suppose we have a pool of m classifiers. We estimate $\hat{\mu}$ as:

$$\hat{\mu} = \sum_{i=1}^{m} \alpha_i \mu_i \tag{3.14}$$

subject to

$$\sum_{i=1}^{m} \alpha_i = 1$$

We pick α_is so that they minimize mean square classification error. This error is computed as:

$$\begin{aligned} error(\alpha) &= E[(\mu - \sum_{i=1}^{m} \alpha_i P_i)^2] \\ &= E[(\mu - \sum_{i=1}^{m} \alpha_i (\mu + \epsilon_i))^2] \\ &= E[(\mu(1 - \sum_{i=1}^{m} \alpha_i) - \sum_{i=1}^{m} \alpha_i \epsilon_i)^2] \\ &= E[(\sum_{i=1}^{m} \alpha_i \epsilon_i)^2] \end{aligned} \tag{3.15}$$

To minimize classification error we need to minimize $E[(\sum_{i=1}^{m} \alpha_i \epsilon_i)^2]$ subject to $\sum_{i=1}^{m} \alpha_i = 1$. We solve this constraint optimization problem using Lagrange Multipliers. The Lagrangian function is

$$f(\alpha) = error(\alpha) + \lambda(1 - \sum_{i=1}^{m} \alpha_i) \tag{3.16}$$

We find the stationary points of this equation

$$\frac{\partial f}{\partial \lambda} = 1 - \sum_{i=1}^{m} \alpha_i = 0$$

and

$$\begin{aligned}\frac{\partial f}{\partial \alpha_i} &= E[2\epsilon_i \sum_{j=1}^{m} \alpha_j \epsilon_j] - \lambda \\ &= \sum_{j=1}^{m} 2\alpha_j E(\epsilon_i \epsilon_j) - \lambda = 0\end{aligned}$$

In this method, we assume that the noise of different classifiers of the pool are independent from each other. This important assumption implies that $E[\epsilon_i \epsilon_j] = E[\epsilon_i]E[\epsilon_j]$. Since we assumed noise to have mean 0 we will have $E[\epsilon_i \epsilon_j] = E[\epsilon_i]E[\epsilon_j] = 0 \times 0 = 0$, for $i \neq j$.

So we will have:

$$\begin{aligned}\frac{\partial f}{\partial \alpha_i} &= \sum_{j=1, j\neq i}^{m} 2\alpha_j E[\epsilon_i]E[\epsilon_j] + 2\alpha_i E[\epsilon_i^2] - \lambda \\ &= 2\alpha_i E[\epsilon_i^2] - \lambda = 0\end{aligned}$$

The final solution is:

$$\alpha_i = \frac{\frac{1}{\sigma_i}}{\sum_{i=1}^{m} \frac{1}{\sigma_i^2}},$$

which is equal to the weights in the Expression 3.12.

Originally, each of classifiers in the MUV pool is trained using different data samples. Here, we expect that a classifier that is learned from data samples that resemble the current query will be fairly certain of its response - i.e. will have a small variance. By contrast, we expect this variance to be large from a classifier that was based on instances that were different from the query. However, when the number of samples is very small compared to the number of features, losing a portion of the samples is not desirable. The reason is that the reduction can increase the chance of overfitting and lead to less accurate classifiers. It can be more reasonable to use the MUV combination rule when each classifier uses a different subset of features. In this way each classifier will work with smaller number of features. A classifier

working with a more relevant subset of features should perform better than others. Some of the assumptions above are not valid for this new problem. We study this in Chapter 3.

Variance of a Bayesian Network

In this section, we explain the method of computing the asymptotic variance of a Bayesian network used in [AGH01, ASGH08]. Later, Hooper et al. in [HAYGH09] presented an improved approach for this problem.

At a high level, Van Allen et al. [AGH01] assume independent Dirichlet distributions for the parameters of rows of CPtables in a Bayesian network. Then using the fact that it is easy to compute both the variance of the components of a Dirichlet random vector, and the covariance of two components in such a vector, they compute the variances of parameters and covariances of parameters in the same row of a CPtable. Independence of parameters of rows implies that the covariance of parameters in different rows is zero. Then this paper computes the asymptotic variance of a query in terms of these computed variance and covariance values. For this purpose it uses the Delta method that approximates the query function using its first order Taylor expansion around a point and computes the variance of this term instead. This term is linear in parameters of the network hence calculating its variance in terms of variances and covariances of parameters is straightforward.

Now for the details, let v be a variable in a Bayesian network with k possible values $\{x_1, ..., x_k\}$. Then each row of the CPtable corresponding to this variable has k columns. Let $\Theta = \langle \theta_{v,x_1|f}, ..., \theta_{v,x_k|f} \rangle$ be one such row where $\theta_{v,x_i|f}$ is the CPtable entry related to variable v having the value x_i and its set of parents having value equal to vector f. Van Allen et al. [AGH01] assume Dirichlet priors for this row and show the posterior values of the these parameters a follow Dirichlet distribution as well. Moreover, they assume different rows are independent. Then they compute mean, variance, and covariances of components of Θ using the properties of Dirichlet distribution. Let the parameters of posterior distribution of Θ be $\langle \alpha_1, ..., \alpha_k \rangle$.

For each row of CPtable $\Theta \sim Dir(\alpha_1, ..., \alpha_k)$ we can calculate the mean value

of each parameter and the covariance between parameters in a row using properties of the Dirichlet distribution as below [BFH75]:

Suppose $\alpha_0 = \sum_{i=1}^{k} \alpha_i$, then:

$$E[\theta_{v,x_i|f}] = \frac{\alpha_i}{\alpha_0} = \mu_{v,x_i|f} \tag{3.17}$$

$$Var[\theta_{v,x_i|f}] = \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0 + 1)} = \frac{\mu_{v,x_i|f}(1 - \mu_{v,x_i|f})}{\alpha_0 + 1} \tag{3.18}$$

$$Cov[\theta_{v,x_i|f}, \theta_{v,x_j|f}] = -\frac{\alpha_i \alpha_j}{\alpha_0^2(\alpha_0 + 1)} = -\frac{\mu_{v,x_i|f}\mu_{v,x_i|f}}{\alpha_0 + 1} \tag{3.19}$$

Let $\mathbf{\Theta}$ be the set of all parameters of a network and $q(\mathbf{\Theta})$ be a query expressed as a conditional probability of the form

$$q(\mathbf{\Theta}) = P(H = h|E = e, \mathbf{\Theta}) \tag{3.20}$$

where H and E are subvectors of X and h and e are legal assignments to these subvectors. The Taylor expansion for $q(\mathbf{\Theta})$ is

$$q(\Theta) = q(\mu) + D + R \tag{3.21}$$

where

$$D = \sum_{v \in V} \sum_{f \in F_v} \sum_{x \in X_v} q'_{v,x|f}(\theta_{v,x|f} - \mu_{v,x|f}) \tag{3.22}$$

is the first order Taylor expansion of $q(\theta)$ about $q(\mu)$ and R is the remainder. V is the set of variables, X_v is the set of values of variables, and F_v is the set of parents of variable v.

Van Allen et al. [AGH01] show that remainder R is asymptotically negligible compared to D and as a result $q(\mathbf{\Theta})$ can be approximated using $D + q(\mu)$. This result is first used to prove that D is asymptotically normal. This normality follows from asymptotic multivariate normality of components of $\mathbf{\Theta}$ and the fact that D is a linear function of components of θ. Second, it is used to approximate the variance of $q(\mathbf{\Theta})$ with variance of D.

$$Var(q(\mathbf{\Theta})) \approx Var(D) = \sum_{v \in V} \sum_{f \in F_v} \sum_{x \in X_v} \sum_{y \in X_v} q'_{v,x|f} q'_{v,y|f} Cov(\theta_{v,x|f}, \theta_{v,y|f}) \tag{3.23}$$

It should be noted that when the arguments of covariance $Cov(.,.)$ correspond to parameters in different rows of a CPtable, the covariance is zero. In addition, when arguments of covariance above are the same the value is equal to variance which is calculated using Equation 3.18. Finally, if the arguments of covariance are different but they correspond to parameters in the same row of a CPtable the covariance is calculated in Equation 3.19. The first derivative $q'_{v,x|f}$ derived in [GGS97, Dar00] is equal to

$$q'_{v,x|f} = \frac{P(H=h, X_v=x, F_v=f|E=e)}{\mu_{v,x|f}} - \frac{P(H=h|E=e)P(X_v=x, F_v=f|E=e)}{\mu_{v,x|f}} \quad (3.24)$$

We use a short notation for this equation:

$$q'_{v,x|f} = \frac{p_v(h,x,f|e) - P(h|e)p_v(x,f|e)}{\mu_{v,x|f}} \quad (3.25)$$

After simplification, the variance of response of a Bayesian network to a query is asymptotically equal to:

$$\hat{\sigma}^2[P(c|e)] = \sum_{\theta_{D|f} \in \Theta} \frac{1}{1+\alpha_0} \bar{v}_{c|e}(D|f) \quad (3.26)$$

where

$$\bar{v}_{c|e}(D|f) = \sum_{v \in V} \sum_{f \in F_V} \frac{1}{\bar{\mu}_{v,x|f}} [P_v(h,x,f|e) - P(h|e)P_v(x,f|e)]^2 - [P_v(h,f|e) - P(c|e)P_v(f|e)]^2$$

In our theoretical analysis in Section 4.1.2, we use a similar approach to the method discussed in this section to compute the covariance of responses of two naive Bayes classifiers. There the classifiers however, are formed differently.

3.2.6 Other Ensemble Methods

Here, we briefly introduce two other ensemble methods: bagging and boosting. Bagging is an iterative ensemble classification method. In each iteration, it draws a bootstrap sample S_m of the training data and learns a classifier on this data. Then, at performance time, for each sample it predicts the class with majority vote [Bre96].

The idea behind boosting is to weight the predictions of the classifiers in the pool with their error. Boosting performs multiple iteration. In each iteration, the weight of one classifier is determined in a way that later classifiers focus on samples that were misclassified by earlier classifiers [FS96].

3.3 Clustering

Clustering is an unsupervised method of organizing data. The goal is to group data in a way that elements in the same group are as close as possible together whereas elements on different groups are as far as possible from each other. As usual different distance measures can be used for this purpose. Euclidean distance, and correlation are some examples. Using the same assumptions and notations as Section 3.2, the Euclidean distance between two datapoints x_i and x_j is defined as:

$$EuclideanDist(x_i, x_j) = (\sum_{k=1}^{p}(X_{ik} - X_{jk})^2)^{(1/2)} \tag{3.27}$$

Also, the correlation between them is defined as:

$$Corr(x_i, x_j) = \frac{Cov(x_i, x_j)}{\sqrt{Var(x_i)Var(x_j)}} \tag{3.28}$$

In clustering algorithms we usually need to input a parameter like the number of clusters or minimum permitted distance for clusters. The algorithm then identifies the partition that optimizes a clustering criterion.

The output of clustering can be hard or soft (fuzzy). In hard clustering each data point membership is binary, i.e. each cluster either include a data point or not and each data point belongs in at most one cluster. In fuzzy clustering each data point has a degree of membership to each cluster. Clustering algorithms can be divided to hierarchical, partitional, probabilistic, and graph-theoretic approaches. Jain et al. provide information about these methods [JMF99].

Later, in Section 4.2, in the feature clustering phase, we use k-means partitional clustering that given the number of clusters, returns a hard clustering of data points. We use clustering to group the similar features, and then only use one from each

cluster for classification. This attempts to prevent our classifiers from using too many similar features, which might lead to overfitting.

Chapter 4

FMUV: A Variant of MUV Method

The original version of MUV discussed in Section 3.2.5 assumes each classifier uses all features and a subset of samples, which is generated by bootstrap sampling. Another approach is to use a pool of classifiers that each uses a different subset of features but over the complete training data. For distinction, we call the former, which deals with samples, SMUV and the latter, which deals with features, FMUV. In SMUV, as we explained in Section 3.2.5, we assume that errors of different classifiers are independent from each other. Moreover, SMUV implicitly assumes that each classifier is unbiased. Neither of these assumptions is valid for FMUV.

We think the FMUV classifier with naive Bayes or TAN base classifiers is appropriate for solving our problem because it uses many weak classifiers and merges them to achieve a strong classifier. Naive Bayes assumes that the features it is using are independent. We try to satisfy this assumption by assigning uncorrelated features to each classifier. Specifically, we expect FMUV to perform better than SMUV in this problem, because FMUV uses all training data and so does not loose any part of it. This is particularly helpful in our problem, because we have only a few samples in our training data. Also it embeds an automatic feature selection inside, which is greatly useful for our problem with this huge number of features. It should be noted that biologically we know that some SNP are related to each other using the notion of linkage disequilibrium. To embed this relation in our method, we also have used TAN classifier as the base classifier in both FMUV and SMUV. TAN also assumes a simple dependence between features.

In this chapter, we extract the combination rule for FMUV and explore some of

its theoretical properties. Moreover, we propose an algorithm for assigning features to the classifiers in the pool of FMUV.

4.1 Theoretical Analysis

In this section we present our theoretical analysis.

4.1.1 FMUV: Feature Mixture Using Variance

Assume there exist n different features in a classification task, some of which are really effective in the classification and some not. Let

$$\mu = P(Class = c|F_1 = f_1, ..., F_n = f_n) \tag{4.1}$$

be the true probability of response to a query given all feature values.

We want to estimate μ using a pool of m classifiers. Suppose each of the classifiers in the pool is a naive Bayes classifier that uses a subset of these features. Let f_{kC_m} be the m^{th} feature used in the k^{th} classifier. The query given to the k^{th} classifier is the probability $P(C = class_j|F_{kC_1} = f_{kC_1}, ..., F_{l_1C_1} = f_{l_1C_1})$ where $\{F_{1C_k}, ..., F_{l_1C_k}\}$ is the subset of features used by the k^{th} classifier.

We assume each classifier in the pool responds to this query with a value that is the true value μ plus an additive Gaussian noise

$$\begin{bmatrix} p_1 \\ . \\ . \\ . \\ p_m \end{bmatrix} = \mu + \begin{bmatrix} \epsilon_1 \\ . \\ . \\ . \\ \epsilon_m \end{bmatrix}$$
$$\epsilon = [\epsilon_1, ..., \epsilon_m]^T \sim N(b, \Sigma)$$

in which b is an $m \times 1$ vector of bias values of each classifier in its response to this query, due to losing some important features and $\Sigma = [\sigma_{i,j}]$ is the covariance matrix of such error between classifier responses to this query. Since the true value of a query is fixed, the covariance of errors of classifiers is equal to covariance of responses of classifiers to that query.

We again estimate μ with a linear combination of responses of classifiers: $\hat{p} = \sum_{i=1}^{m} \alpha_i p_i$ where $\sum_{i=1}^{m} \alpha_i = 1$. Our goal is to find αs that minimize mean square

classification error. Using the same steps in Equation 3.15, $error(\alpha) = E[(\sum_{i=1}^{m} \alpha_i \epsilon_i)^2]$. So our goal is to minimize $E[(\sum_{i=1}^{m} \alpha_i \epsilon_i)^2]$ subject to $\sum_{i=1}^{m} \alpha_i = 1$. Using Lagrange multipliers we will have:

$$
\begin{aligned}
f(\alpha) &= error(\alpha) + \lambda(1 - \sum_{i=1}^{m} \alpha_i) \\
&= E[(\sum_{i=1}^{m} \alpha_i \epsilon)^2] + \lambda(1 - \sum_{i=1}^{m} \alpha_i)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial f}{\partial \alpha_i} &= E[2\epsilon_i \sum_{j=1}^{m} \alpha_j \epsilon_j] - \lambda \\
&= E[2 \sum_{j=1}^{m} \epsilon_i \alpha_j \epsilon_j] - \lambda \\
&= 2 \sum_{j=1}^{m} \alpha_j E[\epsilon_i \epsilon_j] - \lambda \\
&= 2 \sum_{j=1}^{m} \alpha_j (\Sigma_{ij} + E[\epsilon_i] E[\epsilon_j]) - \lambda \qquad \text{(From the definition of covariance)} \\
&= 2(\sum_{j=1}^{m} \alpha_j \sigma_{ij} + b_i b_j) - \lambda
\end{aligned}
$$

Setting this derivative to be equal to 0 means we should solve, for each i, $\frac{\lambda}{2} = \sum_{j=1}^{m} \alpha_j \sigma_{ij} + b_i b_j$.

The final solution to this equation is

$$
\alpha_k = \frac{\sum_{i=1}^{m} (C^{-1})_{ik}}{\sum_{i=1}^{n} \sum_{j=1}^{m} (C^{-1})_{ij}} \tag{4.2}
$$

where $C = \Sigma + bb^T$.

This means that once we can estimate the covariance matrix of errors in responses of different classifiers to a classification query and bias of each classifier, the weight of each classifier in the ensemble is simply calculated using Equation 4.2.

4.1.2 Result 1: Computing Covariance of Naive Bayes Classifiers with Overlapping Features

The focus of this section is how to calculate the covariance of query responses of different classifiers for a given query.

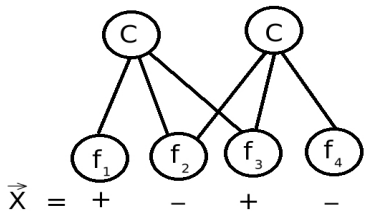

Figure 4.1: Two simple naive bayes classifiers with overlapping features

Our goal is to calculate the covariance between the responses of two Naive Bayes classifiers to a certain query. Our approach is similar to the one used in [AGH01] and explained in Section 3.2.5. Similarly, we adapt independent Dirichlet priors for the parameters in each row of a CPtable.

Assume we have two Naive Bayes that use the same training samples but different subsets of features with possible overlap. (As an example look at Figure 4.1.) Let q_1 be the query response $P(Class = c|E_1 = e_1, \Theta_1 = \mu_1)$ in the first network where E_1 is the subset of variables in the first network, e_1 is a legal value assignment to this subset, Θ_1 is the set of CPtables related to this network, and μ_1 is a valid assignment to parameters of CPtables derived from training data. Similarly we define q_2 to be $P(Class = c|E_2 = e_2, \Theta_2 = \mu_2)$, which is the response in the second network . Now we consider the first order Taylor expansion of expressions $q_1(\Theta)$ around $q_1(\mu)$ and call it D_1. Similarly, we call the first order Taylor expansion of expressions $q_2(\Theta)$ around $q_2(\mu)$ as D_2:

$$q_1(\Theta) = q_1(\mu) + D_1 + R_1$$
$$q_2(\Theta) = q_2(\mu) + D_2 + R_2$$

where

$$D_1 = \sum_{v \in V_1} \sum_{f \in F_{v1}} \sum_{x \in X_{v1}} q'_{v,x|f}(\theta_{v,x|f} - \mu_{v,x|f})$$
$$D_2 = \sum_{v \in V_2} \sum_{f \in F_{v2}} \sum_{x \in X_{v2}} q'_{v,x|f}(\theta_{v,x|f} - \mu_{v,x|f})$$

The remainder terms R_1 and R_2 can be expressed in terms of the matrix of second derivatives of q_1 and q_2 respectively.

The covariance of $q_1(\theta)$ and $q_2(\theta)$, after ignoring the remainder terms, is equal to the covariance of D_1 and D_2, which we compute below:

$$Cov(D_1, D_2) = \sum_{v_1 \in V_1} \sum_{f_1 \in F_{v1}} \sum_{x \in X_{v1}} \sum_{v_2 \in V_2} \sum_{f_2 \in F_{v2}} \sum_{y \in X_{v_2}} q'_{v_1,x|f_1} q'_{v_2,y|f_2} Cov(\theta_{v_1,x|f1}, \theta_{v_2,y|f2}) \tag{4.3}$$

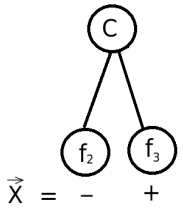

Figure 4.2: Subnetwork including only the common nodes.

The asymptotic variance of response to a query of this network computed using Equation 3.23 has exactly the same terms. We can conclude that $Cov(D_1, D_2)$ is equal to the variance of query response of the subnetwork consisting only of shared nodes.

Theorem 1 *The covariance of the responses to a query of two naive Bayes classifiers that use the same training data is equal to the variance of the response of the shared subnetwork to that query.*

4.1.3 Result 2 : Asymptotic Normality of Response of MUV Classifiers

In this section we show that the response to a query of both FMUV and SMUV classifiers asymptotically follow normal distribution. In addition, we derive expressions for the asymptotic mean and variance of this distribution.

As we mentioned in Section 3.2.5, Van Allen et al. [AGH01] prove that under the assumption of independent Dirichlet priors for the various CPtable rows Θ, the response Q of a Bayesian network is asymptotically normal. This result follows from the asymptotic multivariate normality of the components of Θ and the fact that a query $q(\Theta)$ can be approximated using a linear function of Θ. Let Q_j be the response of the j^{th} Bayesian network classifier to a specific query and $Q_S = \sum_{j=1}^{n} \alpha_j^S Q_j$ and $Q_F = \sum_{j=1}^{n} \alpha_j^F Q_j$ be the responses of SMUV and FMUV respectively. Since SMUV and FMUV are linear ensembles of classifiers, we can approximate Q_S and Q_F in turn as linear functions of Θ. Thus with the same reasoning that the response of a Bayesian network to a query is asymptotically normal, we see that:

Theorem 2 *The distribution of response to a query of an MUV ensemble classifier is asymptotically normal.*

The next step is to find means and variances of Q_S and Q_F. Using the linear property of expectation, it is easy to compute the means of Q_S and Q_F:

$$E(Q_S) = E(\sum_{j=1}^{n} \alpha_j^S Q_j) = \sum_{j=1}^{n} \alpha_j^S E(Q_j(\Theta)) \tag{4.4}$$

$$E(Q_F) = E(\sum_{j=1}^{n} \alpha_j^F Q_j) = \sum_{j=1}^{n} \alpha_j^F E(Q_j(\Theta)) \tag{4.5}$$

Cooper et al. [CH92] shows that in some conditions asymptotically $E(q(\Theta)) = q(E(\Theta))$. This property is used in [AGH01] in computing the variance. Here we

use it and conclude that asymptotically:

$$E(Q_S) = \sum_{j=1}^{n} \alpha_j^S E(Q_j(\Theta)) = \sum_{j=1}^{n} \alpha_j^S Q_j(E(\Theta))) \tag{4.6}$$

$$E(Q_F) = \sum_{j=1}^{n} \alpha_j^F E(Q_j(\Theta)) = \sum_{j=1}^{n} \alpha_j^F Q_j(E(\Theta))) \tag{4.7}$$

For computing the variances we use a generalized form of the following property of variance.

$$Var(a_1x_1+a_2x_2) = {a_1}^2 Var(x_1)+{a_2}^2 Var(x_2)+2a_1a_2Cov(x_1,x_2) = \sum_{i=1}^{2}\sum_{j=1}^{2} a_ia_jCov(x_1,x_2) \tag{4.8}$$

In the same way we will have

$$Var(Q_S) = Var(\sum_{j=1}^{n} \alpha_j^S Q_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i^S\alpha_j^S Cov(Q_i,Q_j) \tag{4.9}$$

and

$$Var(Q_F) = Var(\sum_{j=1}^{n} \alpha_j^F Q_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i^F\alpha_j^S Cov(Q_i,Q_j) \tag{4.10}$$

So once we compute the covariance matrix of responses of members of the classification pool, computing the variance of the distribution is straightforward. In Section 4.1.2, we showed how to compute the necessary covariance matrix.

4.1.4 Estimating Query-based Bias for a Naive Bayes Classifier

Up to now in this chapter, we derived a formula for FMUV combination of classifiers in Section 4.1.1 (Equation 4.2). In later sections, we could estimate all terms of this formula except for the bias terms. So what we need is a query based estimation of bias in a naive Bayes classifier. Unfortunately, in the duration of this thesis, we did not manage to solve this problem. So we need to postpone the study of performance of our proposed method until we can estimate the bias term.

4.2 Feature Allocation Algorithm

In this section we propose an algorithm for FMUV ensemble classification in a dataset with large number of features. As mentioned in Section 4.1.1, FMUV uses

a pool of classifiers. Each classifier in the pool uses a subset of all features as its variable set, then produces the final result by combining the results of these classifiers. It should be noted that in our method, each classifier uses all data in corresponding training folds. In contrast, in SMUV algorithm each classifier uses a subset of training data generated by bootstrap sampling.

To be able to use FMUV combination rule in practice, we need an effective algorithm for assigning a subset of features to each of the classifiers in the pool. We need to select good features and assign them to classifiers. Using too many similar features in a classifier is not desired. Our goal here is to allocate features to classifiers in a way that maximizes classification accuracy.

Our algorithm for assigning features produces clusters of similar features. Then it forms separate classifiers by selecting one feature from each cluster.

4.2.1 Feature Filtering

As $164,273$ is a large number of features, in the first stage we only select a subset of features. We call this phase "feature filtering", which is an *infold* feature selection procedure, i.e. in this phase, we select the 2000 most informative features from training folds only. By most informative, we mean the features that have highest information gain with respect to class value. Previously, we defined feature selection based on information gain in Section 3.1. Information gain is defined in Equation 3.3. For evaluation, we use 10-fold cross-validation. In 10-fold cross-validation, for each test fold, we train on the remaining 9 folds.

It is important that feature filtering phase is performed infold. Otherwise, observation of the labels of test samples can lead to optimistic classification results. Our evidence for this importance is that we have been able to reach to a 100% test accuracy using 10-fold cross-validation, when $10,000$ features are selected from the complete dataset (not only training folds). Because $10,000$ is a very large number, one might think it is acceptable to pick this number of features from the complete train and test folds. However, 100% is not our real classification accuracy and it is far beyond the accuracy we could reach when feature selection phase only uses train folds.

4.2.2 Feature Clustering

We cluster the 2000 features produced by feature filtering phase using k-means algorithm [Mac67] with correlation as our distance measure. Section 3.3 provides information about clustering. Specifically correlation criterion is defined in Equation 3.28. In our experiments on our prostate cancer dataset, we picked $k \in \{5, 50\}$ as the number of clusters.

4.2.3 Feature Merging

Once clusters are generated, we pick one feature from each cluster for each classifier in the pool. Our pool consists of m naive Bayes classifiers. In one trial, we select one feature from each cluster with replacement, in others without. When we select features without replacement, the resulting feature sets will not have overlap. In contrast, when we select features with replacement, the resulting feature sets can have overlap. In the non-replacing case, if we use all members of one cluster while other clusters are still remaining, we simply ignore the empty clusters. We used three methods for selecting the next feature from cluster i for the classifier j:

1. Select the next feature from cluster i randomly.

2. Select the next remaining feature from cluster i with highest infold correlation with respect to class variable.

3. Select the next remaining feature from cluster i with the lowest sum of correlations with respect to features already selected for classifier j.

The goal of method 2 is to select relevant features with respect to class. The goal of method 3 is to select uncorrelated features, so that the feature independence assumption behind naive Bayes model, which is our base classifier, is partially satisfied. In our experiments on our prostate cancer dataset, we picked $m = 5$ as the number of naive Bayes classifiers. We find this number reasonable for the 2000 total features, which is output of our feature filtering phase. However, we have also tested other numbers, which we do not report here.

4.2.4 Complete Algorithm

Algorithm 3 shows the complete algorithm. For evaluation, we use 10-fold cross-validation. Our algorithm is run infold. It means when we use the i^{th} fold as a test fold, we give the other 9 folds, which are the training folds, to this algorithm. In this way we prevent our algorithm from implicitly observing the class labels of test samples. In summary, this algorithm receives the training folds of data, number of clusters k, number of classifiers m, the feature merging method $meth$ as defined in Section 4.2.3 and a Boolean variable called $replace$ that shows whether we want a selection of features from clusters with replacement or without replacement. The algorithm returns the feature set to be used by the i^{th} classifier in the pool.

Algorithm 3 $\{FS_1, ..., FS_m\}$ = FeatureAlloc($trainData, k, m, meth, replace$)

Cluster features in k clusters using k-means and correlation distance measure.
for $i = 1 : m$ **do**
  for all Nonempty clusters C **do**
    F← Select a feature from cluster using selection method $meth$
    $FS_i \leftarrow FS_i \cup F$
    if $replace == FALSE$ **then**
      $C \leftarrow C - F$
    end if
  end for
  Return $\{FS_1, ..., FS_m\}$.
end for

Chapter 5

Experimental Results

In this chapter, we report the results of our experimental studies. We divide this chapter into three sections. In the first, we describe our experiments on the Prostate Cancer Radiotherapy Toxicity (PCRT) dataset. In this set of experiments, we use a group of well-known previously presented classifiers to evaluate the accuracy of different classifiers on this dataset. In addition, we use SMUV and FMUV classifiers for this purpose. As mentioned before, SMUV is the original version of MUV classifier and FMUV is our proposed version. In the second section, we evaluate the performance of different versions of MUV classification method on benchmark datasets. In the third section of this chapter, we develop a set of experiments testing the MUV idea in general.

5.1 Experiments on PCRT Dataset

PCRT is a dataset provided by the Cross Cancer Institute (CCI) containing SNP features of prostate cancer patients who have undergone radiation therapy, and toxicity reaction follow-up. As we mentioned in Section 2.1, each SNP is a genetic attribute of a patient that can have one value from the set $\{AA, Aa, aa\}$. Patients react differently to radiotherapy. Some accept it with satisfactory side effects, while the others show unpleasant toxicity responses. Based on the reaction they have shown after radiotherapy, each patient in this dataset is labeled with $\{0, 1\}$. Our goal is to build a classifier with maximum classification accuracy from this dataset.

5.1.1 Further Information about PCRT Dataset

In this dataset the number of patients is only 82, while the number of SNP features is $164,273$. This means the number of features is around 2000 times the number of samples. This ratio makes a classification problem difficult. We do not have any missing values in our dataset.

The labels are generated by studying the recorded status of these patients in several physician visits. A patient who suffered from extensive bleeding in at least one visit after 90 days is labeled as a "1" ; otherwise as "0". Among these 82 patients, 31 are labeled with "1" and the remaining 51 with "0". Hence a majority classifier, which classifies all samples as majority class "0", has 62.20% accuracy. We use this accuracy as a baseline for our study.

To estimate how much each feature can help in the classification process, we computed the correlation between each feature and the class, after converting categorical features to binary. The statistical distribution of correlation scores reveals some information about usefulness of features. To visualize this distribution, we draw a histogram of correlations; see Figure 5.1. The maximum absolute correlation of any feature with the class is 0.4982, which is very low. Such a histogram and such low correlations suggest the possibility that no single feature is closely related to the class variable.

5.1.2 Data Preprocessing

While many classifiers can use nominal features, some only work with numerical features. To be able to use the latter group of classifiers on our dataset, we need to transfer our nominal feature values to numerical equivalents. One method is to choose arbitrary mappings — like "aa" to 1, "Aa" to 2, and "AA" to 3. The disadvantage is that assigning a bigger or smaller number to a nominal feature value compared to other features can affect the classification accuracy. Another suggestion is to convert each feature with nominal values to a set of binary features, each of which corresponding to one of the possible feature values. In the second method a SNP feature with possible values $\{AA, Aa, aa\}$ will be converted to three binary

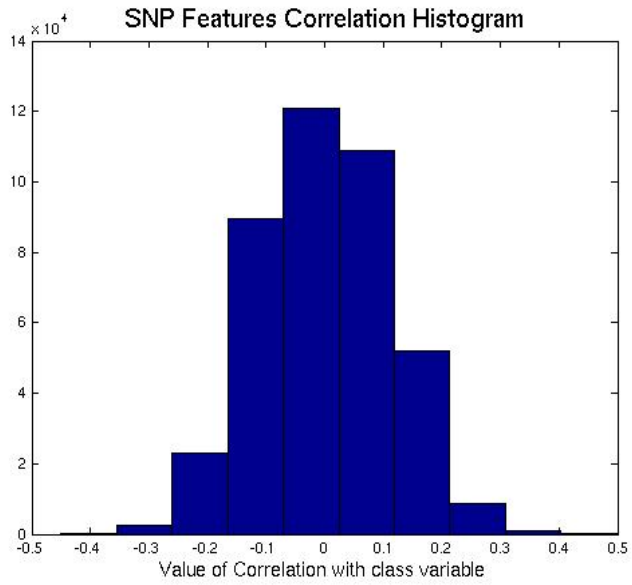

Figure 5.1: Histogram of Correlation Between Class and Features

features with "AA" mapped to 100, "Aa" mapped to 010, and "aa" mapped to 001. The disadvantage here is that the number of features, which is originally too big, will now grow to 3 times its size. In addition, there will be redundant information in the features, i.e. the number of bits we use in this way is more than its minimum possible number. While both methods have disadvantages, we have empirically found the latter more promising and use it when we need numerical features.

Given the large number of features in this dataset, it makes sense to try dimensionality reduction techniques hoping to improve classification performance and avoid overfitting. Some classification methods like Lasso [Tib96] automatically achieve this as they include an implicit feature selection mechanism. For the rest, we explicitly run a feature selection procedure. For this purpose, we select only k features with highest information gain (for definition of information gain, see Section 3.1.1), where $k \in \{5, 10, 100, 500, 1000\}$. We are careful to use only the training data for feature selection, as including test data here can lead to optimistic estimation of classification accuracy.

5.1.3 Results

In our experiments, we examined the performance of a group of classifiers on this dataset. We included strong classification methods like support vector machines (SVM), decision trees, and Lasso as well as other classifiers like MUV. For evaluation, we used 10-fold cross-validation. Here, we divide our set of samples to ten nearly equal portions and each time one part is used as the test data and others as training data. The final result is the average of accuracy in all 10 test folds. Table 5.1 shows a summary of these results. We present the details of our experiments for each classifier in Appendix A.

Type of Classifier	Classification Accuracy + std (%)	# of features	Training Accuracy (%)	p
Majority Class	62.20± 0	0	62.20	-
Decision Tree	58.54± 24.76	100	96.34	-
Naive Bayes	64.63± 8.44	500	100	0.1932
Simple Logistic	67.07± 17.91	500	100	0.2061
SVM	65.85± 12.56	100	100	0.1910
Decision Table	63.41± 16.06	500	91.46	0.4085
Lasso	48.50± 13.50	All	96.34	-
PCA + SVM	62.20± 2.68	5	62.20	0.50
Bagging	58.54± 10.78	1000	95.12	-
Boosting	67.07± 12.49	100	100	0.1244
SMUV	62.20± 7.29	500	100	0.50
FMUV	62.20± 9.51	2000	100	0.50

Table 5.1: Best Results Achieved Using Well-known Classifiers

In Table 5.1, the first column shows the type of classifier used. Column 2 represents the best 10-fold classification accuracy we could reach using this type of classifier. If there are more than one experimental setting leading to the best result we arbitrarily list one of them here. We have also included the standard deviation of accuracy in this column. Column 3 shows the number of features $k \in \{5, 10, 100, 500, 1000\}$ leading to this best result. Here "All" means we did not use a feature selection method. As the majority classifier does not use any features for classification, we used "0" for the number of features for this classifier. Column 4 shows the classification accuracy on the training data.

The second column of Table 5.1 shows the baseline here is 62.20%, correspond-

ing to majority classifier. Other classifiers do not perform significantly better than the baseline. The best accuracy found is 67.07% using simple logistic regressor and boosting methods, which is only 4.73% above baseline. The standard deviation of error is smaller for boosting, 12.49, than for simple logistic regression, 17.91. The classification mean and standard deviation of classification accuracy for each classifier is shown in Figure 5.2.

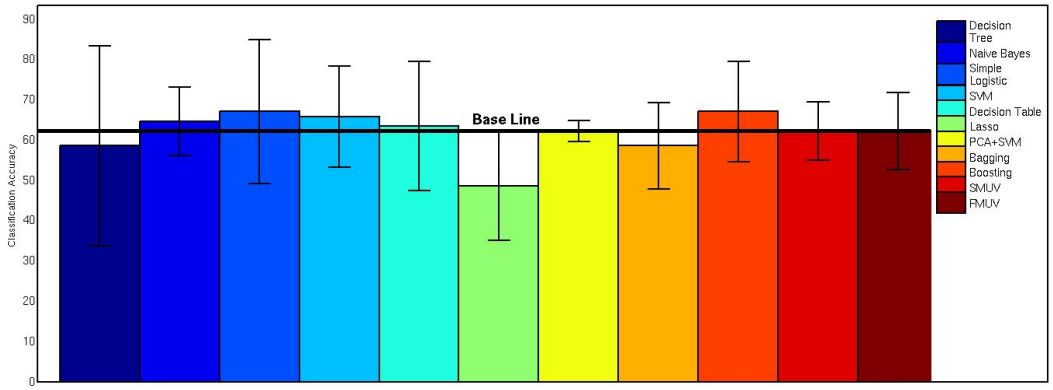

Figure 5.2: Prediction errors and error bars for different classifiers.

To compare the methods more precisely, for the classifiers not less accurate than baseline, we compute the statistical significance of the null hypothesis that their accuracy is from normal distribution with the same mean as baseline, using unpaired t-test. The last column of Table 5.1 shows the p-values for these tests. Using 90% significance level, the hypothesis is not rejected for any of the classifiers in the test. This implies that none of the methods are performing significantly better than the baseline.

Looking back at Table 5.1, we see that despite the low accuracy on test data, the classification accuracy on training data is usually high, often near 100%. This high training accuracy and low test accuracy suggest that the classifiers are overfitting. Low number of samples in comparison with number of features causes our models to overfit. To prevent overfitting we have included feature numbers as low as 5 and 10 in our experiments. However, as we report in Appendix A, unfortunately we observed that, while training accuracy is still high, this reduction in number of features does not improve test accuracy.

We let SMUV refer to the original version of MUV classification method, which uses an ensemble of classifiers that each use all features but only a subset of samples generated by bootstrap sampling. FMUV is our proposed version of MUV, whose base classifiers each use a subset of features, but all samples. We derived the combination rule for this classifier in Section 4.1. Table 5.1 presents the results, which show that neither version of MUV performed better than baseline for the PCRT dataset.

Finally, we show the confusion matrix for each classifier in Table 5.2. Each each row of the matrix represents the number of samples in an actual class, while each column of the matrix represents the number of samples in a predicted class. The number of samples in actual class "0" is 51, while the number of samples in actual class "1" is 31. While the methods PCA+ SVM, SMUV, and FMUV, all have 62.20% accuracy, the confusion matrices are not the same for all of them. PCA+ SVM and FMUV act like a majority classifier and classify all samples as "0". While SMUV classifies, 3 samples with label "1" and 48 samples with label "0" correctly.

5.1.4 Summary of FMUV on PCRT data

In our experiments on PCRT data, we used many well-known and strong classification methods, but none of them gave us a satisfactory result. Therefore, we come to conclusion that given this dataset , it is unlikely that these techniques will be able to learn an accurate classifier for our goal. This conclusion may have two reasons. First, there is not any relation between these SNPs (only a subset of human SNPs are in the dataset) and late radiation toxicity. Second, the dataset is not appropriate for the task of finding an accurate classifier (i.e., even if there is a relation between SNPs and toxicity, this relation is not inferable from this dataset). In addition, it should be noted that our experiments was not limited to what we presented here and in the appendix. We considered a large number of different feature selection methods e.g. using correlation, gain ratio, etc. criteria and also several classification methods, e.g. TAN classifiers, MUV ensemble with TAN base classifiers, nearest shrunken centroids classification method [WZ07]. In addition, we

attempted to learn from clinical features, in addition to the SNP value. Unfortunately, none yielded an improvement in accuracy, which is why we just presented a subset of them in this dissertation.

5.2 Experiments on Benchmark Datasets

The goal of this section is to evaluate the performance of different versions of MUV on some benchmark datasets like datasets from the UCI machine learning repository [UCI]. We present the classification results of SMUV and FMUV with and without feature overlap between base classifiers in Table 5.2. As we mentioned in Section 4.1.4, we need an estimate of the bias of response of a naive Bayes classifier to a query in FMUV combination rule. Since we could not come up with a theoretically sound estimation in this thesis, we set the biases equal to 0 — i.e., assume a pool of unbiased classifiers.

In Table 5.2, column 1 shows the name of dataset. The first three datasets, Spect Heart, Chess, and Vote, are from the UCI repository, and the last two, Corral and Mofn-3-7-10, are from [KJ97]. Column 2 shows the classification accuracy of the SMUV algorithm. Columns 3 and 4 show classification accuracy of FMUV with and without overlap respectively.

Since our method, FMUV, is a variation of SMUV, it makes sense to compare its result with the result of SMUV. Table 5.2 shows that in the first three datasets, FMUV without feature overlap could reach better classification accuracy than SMUV — i.e., our proposed improvement worked for these datasets. Secondly, our results show that in most of these data sets imposing overlap in features of base classifiers and then using the combination rule proposed in Equation 4.2 with bias terms set to 0 did not help in improving the classification accuracy. The only improvement, which is as small as 2.34%, occurred in the last dataset. Therefore, it shows that this estimation of bias with feature overlap, (i.e. setting it to 0) is not effective. The best results ever on these datasets appear in Appendix A.11.

5.3 Variance of Naive Bayes Classifiers

The goal of this set of experiences is to see if the variance of the response to a query is different when a classifier predicts the response *correctly* from when it predicts *incorrectly*. Here, we use the UCI and other datasets used before in Section 5.2. For each test sample, if the prediction is correct we include its variance to one group and if the prediction is incorrect we include it in another group. The method for computing variance of response of a Bayesian network to a query is proposed by [AGH01], as we explained in Section 3.2.5.

To evaluate the result, we draw the histogram of variances of the response to the queries when classifier is correct versus when it is incorrect. Since the values of variances are small, we also show the histogram of logarithm of variances. To be able to compare, we set the size of populations of correct and incorrect responses to be equal; e.g. if there are 1000 correct predictions, we also have exactly 1000 incorrect predictions. In this way, it is easier to compare the histograms. In this set of experiments we use a naive Bayes classifier. Figure 5.3, 5.4, 5.5, 5.6, and 5.7 show the histograms related to Corral, Mofn3-7-10, Vote, Spect and Chess datasets respectively.

Studying these histograms, we find out that the histograms of correct and incorrect variances do not look completely alike. When the classifiers are predicting correctly, the values of variances are more concentrated closely around zero and when they are predicting incorrectly they are farther from zero. The experiment on mofn-3-7-10 data with histograms in Figure 5.4 shows this difference better than other experiments. Since the values of variances are small, a histogram of the logarithm of the variances shows the distinction between the two distribution more clearly. As all histograms show, the logarithm of variances are smaller when a naive Bayes classifier is right versus when it is wrong in answering to a classification query.

Moreover, we use a t-test to test the null hypothesis that both variances come from distributions with equal mean values. Columns 2 and 3 of Table 5.4 shows the expected value and standard deviation of variances when a naive Bayes classifier is right and when it is wrong for different datasets. In addition, column 4 shows

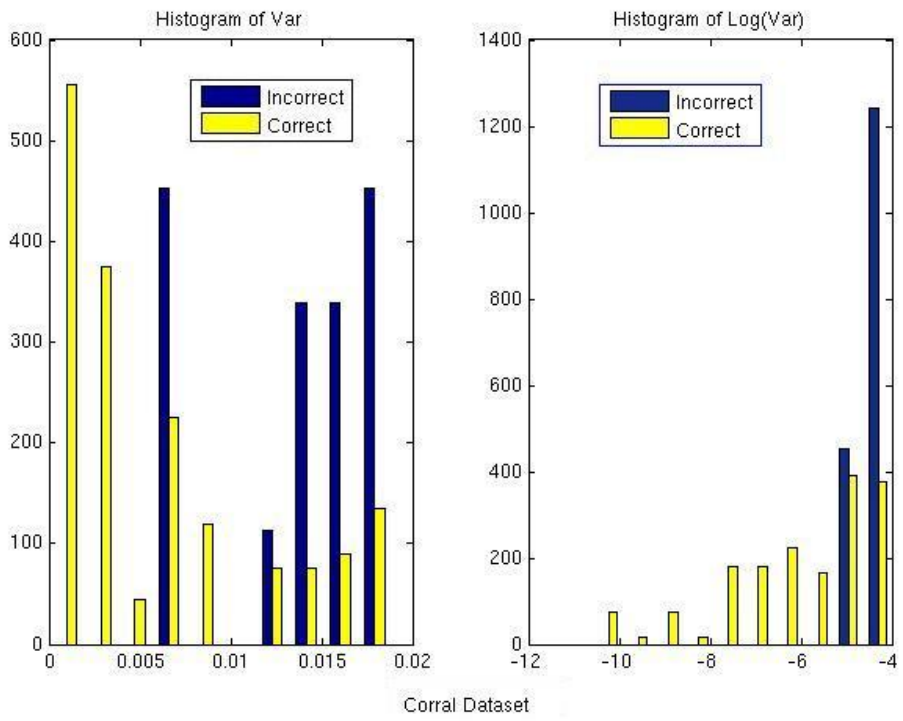

Figure 5.3: Comparison of variances and logarithm of variances of response of a naive Bayes classifier, when classifier is correct versus when it is incorrect in its prediction for Corral Data.

the rejection or non-rejection of the null hypothesis that both variances are from Normal distribution with equal means using two sample t-test.

As we can see in all our experiments, the expected value of variances are smaller when classifier is right that when it is wrong. Moreover, every time the null hypothesis is rejected at the 95% significance level — in fact, this rejection is typically considerably stronger. The results we see here are promising and strengthen the idea that correctness of response of a naive Bayes classifier to a query is related to its variance in responding to the query. Smaller values of variance seem to be more reliable.

It should be noted that we also explored the behavior of TAN classifiers in the same experimental setting. We do not present the details of results related to TAN classifier here. However, in summary, TAN also showed the same behavior; i.e. the variance of response to the queries is smaller when TAN is predicting correctly versus when it is predicting incorrectly.

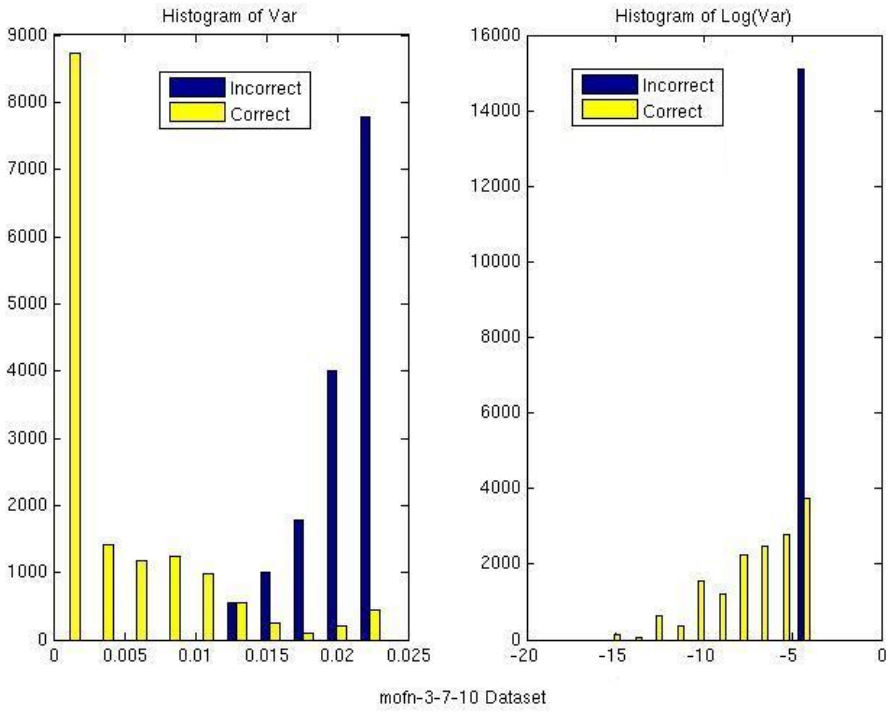

Figure 5.4: Comparison of variances and logarithm of variances of response of a naive Bayes classifier, when classifier is correct versus when it is incorrect in its prediction for Mofn-3-7-10 data.

			Predicted	
Decision Tree(C4.5)			0	1
	Actual	0	29	22
		1	12	19
Naive Bayes			0	1
	Actual	0	50	1
		1	28	3
Simple Logistic			0	1
	Actual	0	43	8
		1	19	12
SVM			0	1
	Actual	0	47	4
		1	24	7
Decision Table			0	1
	Actual	0	29	22
		1	12	19
Lasso			0	1
	Actual	0	22	29
		1	13	18
PCA + SVM			0	1
	Actual	0	51	0
		1	31	0
Bagging			0	1
	Actual	0	40	11
		1	23	9
Boosting			0	1
	Actual	0	36	15
		1	22	19
SMUV			0	1
	Actual	0	48	3
		1	28	3
FMUV			0	1
	Actual	0	51	0
		1	31	0

Table 5.2: Confusion Matrix for Different Classifiers on PCRT Dataset

Dataset Name	SMUV Accuracy (%)	FMUV Accuracy (%) without Feature Overlap	FMUV Accuracy (%) with Feature Overlap
Spect Heart	70.59	72.19	67.91
Chess	87.52	91.37	72.23
Vote	90.11	93.77	88.52
Corral	89.06	88.28	78.12
mofn-3-7-10	87.89	79.69	82.03

Table 5.3: Classification Accuracy of MUV Algorithm on Benchmark datasets

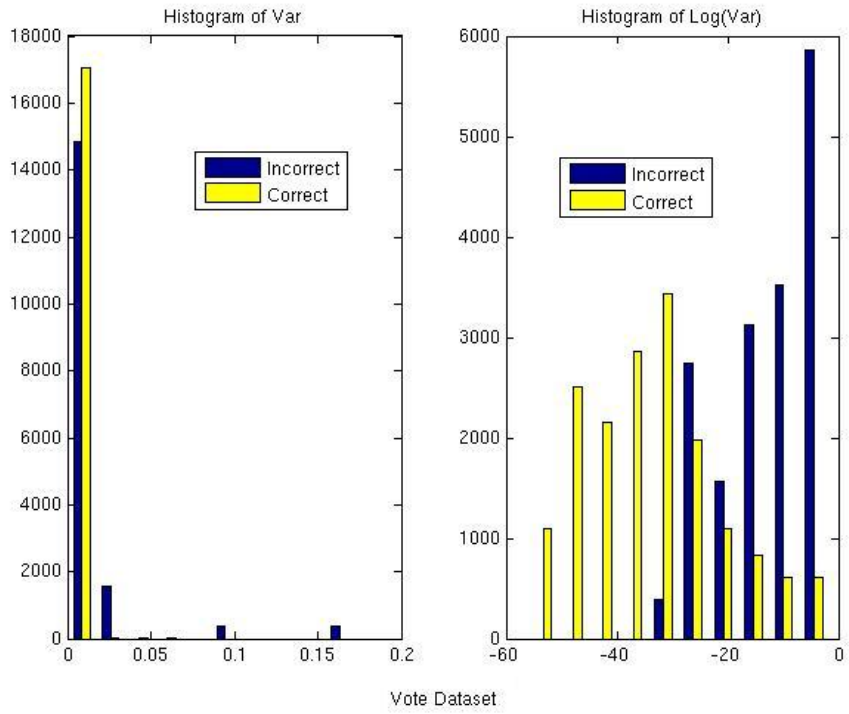

Figure 5.5: Comparison of variances and logarithm of variances of response of a naive Bayes classifier, when classifier is correct versus when it is incorrect in its prediction for Vote data.

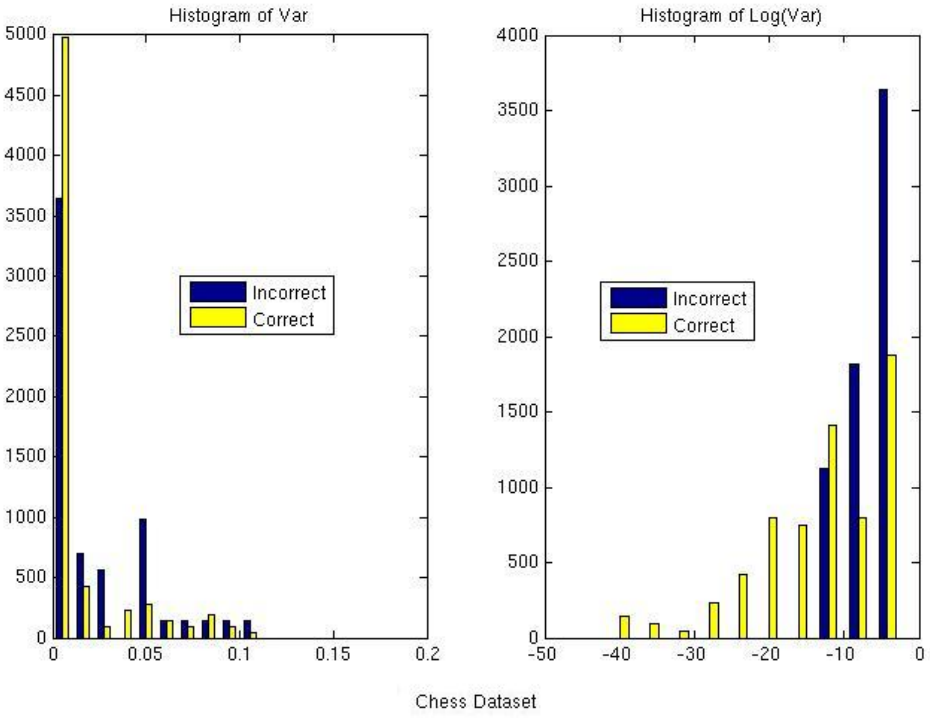

Figure 5.6: Comparison of variances and logarithm of variances of response of a naive Bayes classifier, when classifier is correct versus when it is incorrect in its prediction for Spect data.

Dataset	mean$\pm$ std for V_F	mean$\pm$ std for V_T	p
Spect	0.0213$\pm$ 0.0280	0.0122 $\pm$ 0.0247	3.46×10^{-2}
mofn-3-7-10	0.0204$\pm$ 0.0025	0.0042 $\pm$ 0.0056	2.97×10^{-165}
Chess	0.0039$\pm$ 0.0064	0.0015 $\pm$ 0.0038	5.66×10^{-10}
Corral	0.0136$\pm$ 0.0046	0.0058 $\pm$ 0.0058	1.97×10^{-6}
Vote	0.0095$\pm$ 0.0293	0.00053 $\pm$ 0.0042	4.10×10^{-8}

Table 5.4: Mean and standard deviation of variance of response to a query. V_F shows variances when prediction of a naive Bayes Classifier is incorrect, V_T shows variances when prediction is correct. We also show p-values that this two sample t-test rejects the null hypothesis.

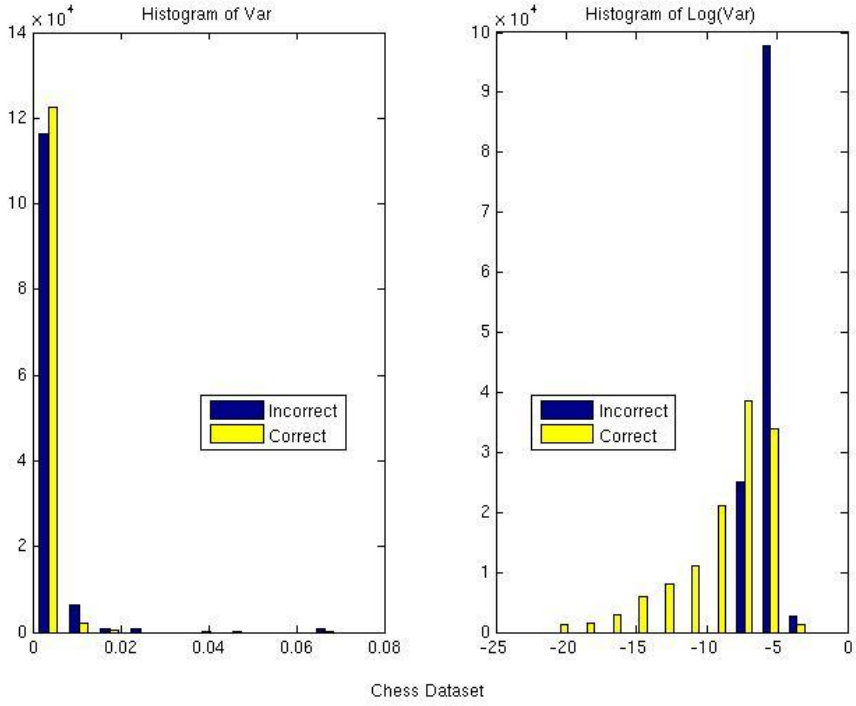

Figure 5.7: Comparison of variances and logarithm of variances of response of a naive Bayes classifier, when classifier is correct versus when it is incorrect in its prediction for Chess data.

Chapter 6

Conclusions

Our studies in this dissertation was focused on two different topics. The first was to find the most accurate classifier for PCRT data. The second was to explore and improve the MUV classification method. In this section, we summarize the attempts and results achieved in both parts and present our suggestions for future works.

In the first problem, we dealt with a high dimensional dataset consisting of $164,273$ SNP features of 82 patients for classification. Some of these patients experienced late toxicity after radiation and some did not. With an assumption that different reaction to radiotherapy has genetic reasons, we tried to find an efficient classifier using only these SNP features. Finding such a classifier has practical importance for radiation oncologists as it would help them to identify which cancer patients will suffer from toxicity from radiotherapy and so avoid giving a high dose of radiation to these patients.

In the experiments related to this part we used several feature selection and classification methods. Unfortunately, most of these methods performed worse than baseline, where the baseline was a majority classifier. Based on these results we conclude that it is unlikely that this class of techniques can learn an accurate classifier on PCRT dataset. This conclusion brings up two possibilities. First, perhaps late toxicity is not related to SNPs or at least not the set of SNPs we are using (notice that the dataset contains only a subset of human SNPs). This means that either the assumption that late toxicity has genetic reasons is wrong or some related SNPs are missed from our dataset. This can be an interpretation of what the majority classifier suggests, i.e. “no patient will suffer from toxicity after radiotherapy with

genetic reasons".

Another possibility is that there are relations between the trait and the set of features we are using, but we could not find it. One reason this can happen is the small number of samples in our dataset. We do not know the minimum number of samples needed for an efficient classification. The number of samples in our problem can be below a lower bound on the number of samples. In any case, if data providers can add to the number of samples, it will be helpful in improving or verifying our results.

In the MUV studies, our results show that we can compute the covariance of the responses to a query of naive Bayes classifiers. Moreover, we showed that the variance of response of an MUV classifier, when base classifiers are naive Bayes, asymptotically follows normal distribution. Then we showed that how the mean and variance can be computed analytically. We computed the two latter results using the covariance we found in our first result. We proposed FMUV as a version of MUV in which each base classifier in the pool uses all samples but a subset of features. We also derived the FMUV combination rule. However, to be able to use this combination rule, we needed an estimate of bias of a naive Bayes classifier. In this project, we used the value of zero for bias — i.e., we assumed the classifiers are unbiased. We empirically studied the performance of our new classifier on benchmark datasets, which shows FMUV method can be superior on some datasets. In addition, using overlapping feature sets and then setting bias to zero did not work. Finally, our empirical studies show with strong evidence that the expected value of variance of response of a naive Bayes classifier is lower when the classifier predicts the response of the query correctly than when it predicts incorrectly. We found this result promising and motivating for future further studies on variance of response of a classifier.

Our suggestions for the PCRT dataset is for the data providers to enhance dataset. It will be useful to add to the number of samples. It is also possible to use biological background knowledge like Hapmap database [Con] for clustering the feature and dimension reduction instead of or accompanied with computational dimension reduction methods. Our future works for MUV part is to estimate the analytical

bias of a naive Bayes classifier. To be able to use FMUV classifier, one suggestion is to treat it as a parameter for a certain problem and numerically try to set the best value for this parameter. One other improvement is to try the improved version of variance computed in [HAYGH09] and study the performance of the resulting classifier.

Bibliography

[AGH01] T. Van Allen, R. Greiner, and P. Hooper. Bayesian error-bars for belief net inference. In *UAI*, pages 522–529, 2001.

[All] Allele picture. Retrieved Dec. 25 2009, from `http://www.csulb.edu/~kmacd/361-6-Ch1_files/allele.jpg`.

[ASGH08] T. Van Allen, A. Singh, R. Greiner, and P. Hooper. Quantifying the uncertainty of a belief net response: Bayesian error-bars for belief net inference. *Artif. Intell.*, 172(4-5):483–513, 2008.

[BFH75] Y.M.M. Bishop, S.E. Fienberg, and P.W. Holland. *Discrete Multivariate Analysis: Theory and Practice*. The MIT Press, Cambridge MA, 1975.

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[BL97] A.L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Bur98] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[CH92] G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[Con] The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437(7063):1299–1320.

[Dar00] A. Darwiche. A differential approach to inference in Bayesian networks. In *UAI*, pages 123–132, 2000.

[DMD+06] S. Damaraju, D. Murray, J. Dufour, D. Carandang, S. Myrehaug, G. Fallone, C. Field, R. Greiner, J. Hanson, C.E. Cass, and M. Parliament. Association of DNA repair and steroid metabolism gene polymorphisms with clinical late toxicity in patients treated with conformal radiotherapy for prostate cancer. *Clinical Cancer Research*, 12(8):2545–2554, April 2006.

[EHJT04] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[FGG+97] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth. Bayesian network classifiers. In *Machine Learning*, pages 131–163, 1997.

[FS96] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *ICML*, pages 148–156. Morgan Kaufmann, 1996.

[GGS97] R. Greiner, A.J. Grove, and D. Schuurmans. Learning Bayesian nets that perform well. In *UAI*, pages 198–207, 1997.

[Guy03] I. Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[GZ02] R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. In *AAAI/IAAI*, pages 167–173, 2002.

[HAYGH09] P. Hooper, Y. Abbasi-Yadkori, R. Greiner, and B. Hoehn. Improved mean and variance approximations for belief net responses via network doubling. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 232–239, 2009.

[HC89] D.L. Hartl and A.G. Clark. *Principles of Population Genetics*. SInaer Associates, INC., 2nd edition, 1989.

[Hol93] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[HTF01] T. Hastie, R. Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.

[JMF99] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review, 1999.

[Jol02] I. T. Jolliffe. *Principal Component Analysis*. Springer, New York, 2002.

[JSZ+01] A. Jackson, M.W. Skwarchuk, M.J. Zelefsky, E.S. Venkatraman D.M. Cowen, S. Levegrun, C.M. Burman, G.J. Kutcher, Z. Fuks, S.A. Liebel, and C.C. Ling. Late rectal bleeding after conformal radiotherapy of prostate cancer. II. volume effects and dose-volume histograms. *Int. Journal Radiation Oncology Biol. Phys*, 49(3):685–698, 2001.

[KJ97] R. Kohavi and G.H. John. Wrappers for feature subset selection, 1997.

[LGG01] N. M. Luscombe, D. Greenbaum, and M. Gerstein. What is bioinformatics? A proposed definition and overview of the field. *Methods Inf Med*, 40(4):346–358, 2001.

[LGW06] C.H. Lee, R. Greiner, and S. Wang. Using query-specific variance estimates to combine Bayesian classifiers, 2006.

[LHF05] N. Landwehr, M. Hall, and E. Frank. Logistic model trees. *Machine Learning*, 59(1-2):161–205, 2005.

[LK02] K. Lukasz and J.C. Krzysztof. Ensemble of classifiers to improve accuracy of the clip4 machine-learning algorithm. In Belur V. Dasarathy, editor, *Sensor Fusion: Architectures, Algorithms, and Applications VI*, volume 4731, pages 22–31. SPIE, 2002.

[Lus07] C.M. Luscombe. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, October 2007.

[Mac67] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[MDD+09] L. Marret, P. De, D. Dryer, L. Ellison, E. Grunfeld, H. Logan, M. MacIntyre, L. Mery, H. Morrison, and H.K. Weir. Canadian cancer statistics 2009. Technical Report ISSN 0835-2976, Canadian Cancer Society, www.cancer.ca, April 2009. Toronto.

[Mit97] T. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[PZS+02] A. Pollac, G.K. Zagars, G. Starschall, J.A. Antolak, J.J. Lee, E. Huang, A.C. von Eschenbach, D.A.Kuban, and I. Rosen. Prostate cancer radiation dose response: results of M.D. Anderson phase III randomized trials. *Int. Journal Radiation Oncology Biol. Phys*, 53:1097–105, 2002.

[QSY07] A. Quinn, A. Stranieri, and J. Yearwood. Classification for accuracy and insight: A weighted sum approach. In Peter Christen, Paul J. Kennedy, Jiuyong Li, Inna Kolyshkina, and Graham J. Williams, editors, *AusDM 2007*, volume 70 of *CRPIT*, pages 203–208, Gold Coast, Australia, 2007. ACS.

[Qui86] J.R. Quinlan. Induction of decision trees. In *Machine Learning*, pages 81–106, 1986.

[RPF04] P. Reutemann, B. Pfahringer, and E. Frank. A toolbox for learning from relational data with propositional and multi-instance learners. In G. I. Webb and X. Yu, editors, *Proc 17th Australian Joint Conference on Artificial Intelligence*, volume 3339 of *LNAI*, pages 1017–1023, Cairns, Australia, 2004. Springer.

[SNP] DNA microarrays. Retrieved Dec. 25 2009, from `http://www.ncbi.nlm.nih.gov/About/primer/microarrays.html`.

[Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[UCI] UC Irvine machine learning repository. Retrieved Dec. 25 2009, from `http://archive.ics.uci.edu/ml/`.

[WZ07] S. Wang and J. Zhu. Improved centroids estimation for the nearest shrunken centroid classifier. *Bioinformatics*, 23(8):972–979, 2007.

[zyg] Gene combinations picture. Retrieved Dec. 25 2009, from http://www.fao.org/docrep/006/x3840e/X3840E04.gif.

Appendix A

Details of Experimental Results

In this section, we explain the details of our experiments for each classifier on PCRT dataset. It should be noted that, the optimal number of features is a parameter that should be set in each fold using another layer of cross-validation. The way we are doing it here, that is having a set of choices for the parameter and announcing the parameter leading to best result, leads to optimistic evaluation of performance of classifier. We designed our experiment in this way on purpose, so that we can more quickly find the promising classification methods and then set the parameters in a second stage only for that classifiers. However, the optimistic results are all below or around baseline. That is they are so bad that we do not see any need to find the actual accuracy with correct way of parameter setting.

A.1 Decision Tree and Decision Table Experiments

We used C4.5 decision tree in the first experiment. In this and next four sections, independent of the classifier used we considered using k top features based on information gain for $k \in \{5, 10, 100, 500, 1000\}$. Table A.1 shows the training and test error in 10-fold cross-validation experiments related to each number of features. The best test accuracy is 58.54%, which happens when we select 100 features.

Next, we used decision tables and achieved a 63.41% accuracy corresponding to 500 features. The training accuracy is 91.46% here (Table A.2). We used Weka implementation of the algorithms of this section [RPF04].

# Features Selected	Training Accuracy (%)	Test Accuracy (%)
5	84.15	47.56
10	97.56	46.34
100	96.34	**58.54**
500	97.56	48.34
1000	96.34	41.46

Table A.1: C4.5 Decision Tree Results

# Features Selected	Training Accuracy (%)	Test Accuracy (%)
5	80.49	50
10	85.37	46.34
100	87.80	58.54
500	91.46	**63.41**
1000	91.46	60.77

Table A.2: Decision Table Results

A.2 Naive Bayes Experiments

In this experiment, we used a naive Bayes classifier. The train and test classification accuracy using 10-fold cross-validation can be seen in Table A.3. The highest test accuracy is 64.63% and happens when we select 500 features. The corresponding training accuracy is 100%. We used Weka implementation of this algorithm [RPF04].

# Features Selected	Training Accuracy (%)	Test Accuracy (%)
5	95.12	54.88
10	98.78	48.78
100	100	54.76
500	100	**64.63**
1000	100	62.20

Table A.3: Naive Bayes Classifier Results

A.3 Simple Logistic Regressor

In this experiment, we used simple logistic regressors [LHF05]. Again the best classification accuracy which is 67.07% is related to 500 features with a corresponding

100% train accuracy (Table A.4). We used Weka implementation of this algorithm [RPF04].

# Features Selected	Training Accuracy (%)	Test Accuracy (%)
5	95.12	48.78
10	100	53.66
100	100	62.20
500	100	**67.07**
1000	100	65.85

Table A.4: Simple Logistic Regressor Used as a Classifier Results

A.4 SVM Experiments

The best accuracy happens when we select the top 100 features which is 65.85%. The corresponding training accuracy is 100% (Table A.5). We used Weka implementation of this algorithm [RPF04].

# Features Selected	Training Accuracy (%)	Test Accuracy (%)
5	95.12	47.56
10	98.78	52.44
100	100	**65.85**
500	100	63.41
1000	100	62.20

Table A.5: SVM Results

A.5 Lasso Experiments

We tested Lasso as another strong classifier. The parameters we needed to set for Lasso was the number of iterations and λ. The number of iterations is an upper bound on the number of features. We considered 5, 10, 30, 50, 100, and 500 as the number of iterations, and $0, 0.001, 0.01, 0.1, 1, 10, 100,$ and 1000. as the values for parameter λ. Table A.6 shows the results achieved using 10-fold cross-validation. The best accuracy here is 62.5% using 1000 features and $\lambda = 1000$.

λ	5	10	30	50	100	500
0	41.50	46.50	44	46	48.25	48.50
0.001	53.50	52.25	51	55	40	47.75
0.01	44.75	39.75	40	42.50	43.75	48.5
0.1	53	57.50	43	46.75	56.75	44.25
1	51.25	51.50	39	34	53	58.25
10	50.25	41.50	39.25	38	39.75	45.25
100	51.25	47.75	47.75	38	45	47.25
1000	**62.5**	50	42.25	43.75	42.75	54.25

Table A.6: Lasso Test Accuracy Results

The best Lasso result is 48.5% when number of iterations is 500 (Table A.6). The corresponding train accuracy is 96.34%. The best Lars result is 62.5% with $\lambda = 1000$ and 5 iterations.

A.6 Principal Component Analysis

The goal of this set of experiments is to use principal component analysis (PCA) as a feature extraction method [Jol02]. We first extracted 81 principal components. We have 82 samples so these are all of the principal components. Then we selected the first k components where $k \in \{5, 10, 20, 81\}$. Then we use a classification algorithm on the reduced data. To make sure that our classification algorithm is not weak, we tested four different methods. Since PCA is only a transformation and does not use the class value, it is not needed to infold application of it. Table A.7 shows the results.

classifier	5	10	20	81
Naive Bayes	53.66	47.56	51.22	58.54
SVM	62.20	62.20	54.88	57.32
5-nearest neighbour	54.88	52.44	52.44	58.54
C4.5	62.20	62.20	57.32	56.10

Table A.7: Accuracy Results of PCA, followed by Different Classification Methods

A.7 Bagging

We used bagging [Bre96], as an ensemble method to compare with SMUV and FMUV ensemble methods. In this method each classifier in the pool is trained on a subset of samples selected by bootstrap sampling. Table A.8 shows the results.

# Features Selected	Test Accuracy (%)
5	54.88
10	53.66
100	56.10
500	52.44
1000	**58.54**

Table A.8: Bagging Results

A.8 Boosting

We used boosting [FS96], as another ensemble method to compare with SMUV and FMUV ensemble methods. In this set of experiments, we used Adaboost as the classifier. Table A.9 shows the results.

# Features Selected	Test Accuracy (%)
5	50
10	53.66
100	**63.07**
500	57.32
1000	54.88

Table A.9: Boosting Results

A.9 SMUV Experiments

In these set of experiments again for each fold we selected 5, 10, 100, 500 or 1000 features from training data and learned a pool of naive Bayes classifiers on the reduced data. Each classifier in the pool uses a training set generated by bootstrap sampling. The results are then combined using original MUV combination rule.

Table A.10 shows the training and testing error for different number of features. These results show that the best test accuracy is related to 500 features. However, it is 62.2% that is no better than our baseline. The training accuracy for 500 features is 100%.

# Features Selected	Training Accuracy (%)	Test Accuracy (%)
5	93.90	45.12
10	98.78	52.44
100	100	54.88
500	100	**62.20**
1000	100	50

Table A.10: SMUV Results

A.10 FMUV Experiments

In this section we report our experiments on FMUV classifiers where each classifier uses a subset of features. We assume our classifiers are unbiassed. When features used by different classifiers do not have any overlap, the FMUV and SMUV combination rules will be the same. In Section A.10.1 we report our experiments on such a classifier. In section A.10.2 we report our experiments where classifiers do have overlap in features they use.

A.10.1 Classifiers without Feature Overlap

In the experiments of this section, we use 10-fold cross-validation for evaluation. We use in-fold feature selection to pick the 2000 most informative features with respect to class variable. Then we cluster them to k clusters for $k \in \{5, 50\}$. We form 5 groups of features using three methods of selecting features from each cluster. Features are selected and removed from cluster without replacement using one of these methods:

- Method 1: Select the next random feature.
- Method 2: Select the next best feature.

- Method 3: Select the next least correlated feature with the features selected so far for the classifier.

Then we learn 5 naive classifiers on these 5 feature sets and combine their results using FMUV combination rule.

Number of clusters	Method 1	Method 2	Method 3
5	57.32	52.44	57.32
50	52.44	60.98	62.20

Table A.11: FMUV Test Accuracy Results without Replacement

A.10.2 Classifiers with Feature Overlap

Table A.12 shows the results when we impose overlap between features used by classifiers. Again, we learn 5 naive Bayes classifiers and combine their results.

Number of clusters	Method 1	Method 2	Method 3
5	53.66	62.20	62.20
50	62.20	62.20	62.20

Table A.12: FMUV Test Accuracy Results with Replacement

A.11 Best Results on Benchmark Datasets

We report the best results ever on benchmark datasets used in Section 5.2. To the best of our knowledge, the best classification accuracy for Spect Heart dataset is 90.4% [LK02], for Chess dataset is 99.2% [Hol93], for Vote dataset is 97.01% [QSY07], for Corral dataset is 100% [GZ02] and mofn-3-7-10 dataset is 100% [GZ02].