Comparative Study of Kernel Based Classification and Feature Selection Methods With Gene Expression Data

by

Mingyue Tan

B.Sc., The University of British Columbia, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University of British Columbia

April 2006

Abstract

Gene expression profiles obtained by high-throughput techniques such as microarray provide a snapshot of expression values of up to ten thousands genes in a particular tissue sample. Analyzing such gene expression data can be quite cumbersome as the sample size is small, the dimensionality is high, and the data are occasionally noisy. Kernel methods such as Support Vector Machines (SVMs) [5, 45] have been extensively applied within the field of gene expression analysis, and particularly to the problems of gene classification and selection. In general, kernel methods outperform other approaches due to their ability to handle high dimensionality easily. In this thesis, we perform a comparative study of various state-of-the-art kernel based classification and feature selection methods with gene expression data. It is our aim to have all the results together in one place so that people can easily see their similarities and differences both theoretically and empirically.

In the literature, a feature selection method is evaluated by the classification accuracies using the features selected by the method. This evaluation criterion measures the classification capabilities of the data after the elimination of irrelevant features. We propose another criterion, called *stability*, to evaluate the feature selection methods in addition to classification accuracies. The feature set selected by a stable feature selection algorithm should not change significantly when some small changes are made to the training data. In this thesis, we use both of two evaluation criteria to compare feature selection methods.

It has been showed that cross validation technique can be used to improve feature selection methods in terms of classification accuracies [8]. In this thesis, we extend one existing feature selection method which utilizes Gaussian Processes (GP) [47] with Automatic Relevance Determination (ARD) [28, 34], and cross validation, and propose a new feature selection method. Experiments on real gene expression data sets show that our method outperforms all other feature selection methods in terms of classification accuracies, and achieves comparable stability as Sparse Multinomial Logistic Regression (SMLR) [23], the most stable feature selection method in the literature.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank all the people who gave me help and support throughout my degree.

First of all, I would like to thank my supervisors, Professor Raymond Ng and Professor Nando de Freitas, for their constant encouragement and rewarding guidance throughout this thesis work. Raymond introduced me to data mining research in his "Honours Hour" section of his undergraduate course. I would not come to graduate school without his inspiration in his "Honours Hour". During my Master studies, he gave me a wonderful introduction to the field of bioinformatics using data mining and machine learning approaches. I am grateful to Nando for giving me the Bayesian education and directing me to the research in kernel methods. I am amazed by Nando's broad knowledge in computer science and the way he made the learning process full of fun.

Secondly, I would like to thank Professor Kevin Murphy for dedicating his time and effort in reviewing my thesis.

Thirdly, I would like to extend my appreciation to my colleagues for their friendship and help, and especially to the following people: Timothy Chan, Jun Wang, Yuhan Cai, Lin Zhong, Suling Yang, and Yizheng Cai.

Last but certainly not least, I would like to thank my parents for their endless love and support.

MINGYUE TAN

The University of British Columbia
April 2006

Chapter 1

Introduction

Gene expression, also known as protein expression or simply expression, is the process by which a gene's coded information is converted into a final gene product (i.e. a protein or any of several types of RNA). Gene expression is a multi-step process that begins with transcription and translation and is followed by folding, post-translational modification of the protein product and targeting. Particularly, the process by which the genes encoded by the human genome are expressed as proteins involves two steps: DNA sequences are initially transcriped into mRNA sequences which in turn are translated into the amino acid sequences of the proteins that perform various cellular functions. Changes in cellular protein synthesis are often estimated by measuring mRNA levels. Measuring mRNA levels can provide a detailed molecular view of particular genes expressed in different cell types under different conditions. Expression of genes can be assessed with DNA microarray or SAGE (Serial analysis of gene expression) among several other techniques.

1.1 Gene Expression Data

Microarray techniques enable simultaneous measurements of the expression levels of tens of thousands of genes. These measurements are made by quantitating the

hybridization of celluar mRNA to an array of defined cDNA or oligonucleotide probes immobilized on a solid surface, such as glass, plastic or silicon chip. Each data point produced by a DNA microarray hybridization experiment represents the ratio of expression levels of a particular gene under two different experimental conditions. Typically, the numerator of each ratio is the expression level of the gene in the condition of interest, which the denominator is the expression level of the gene in the reference state of the cell.

The general goal of SAGE is similar to the DNA microarray. However, SAGE is a sequence-based sampling technique, where observations are not based on hybridization which result in more qualitative, analog values. Instead, SAGE obtains a quantitative profile of cellular gene expression. It measures the absolute count of the mRNA in each sample.

Gene expression data can be represented in matrix form, where columns correspond to genes and rows correspond to tissue samples. In many gene expression data, the samples can be divided into several subgroups or classes, such as tumor tissues and normal tissues. Each subgroup is called a *phenotype*. In this thesis, we focus on the binary case where samples come from two groups only.

The typically characteristics of gene expression data are high dimensionality and low sample size. The sample is small for the following reasons. First, it is very expensive to produce high-throughput gene expression data. It costs about $1,000 to produce a single microarray. Hence, a typical gene expression data with 50 samples (i.e. 50 microarrays) costs $50,000. This is only the technical price without human price adding on top of it. Second, it is not always easy to convince patients to donate their tissue samples for research analysis. Moreover, some diseases are difficult to detect, and therefore collecting a significant number of samples may take years.

1.2 Motivation and Challenges

Gene expression data can help in better understanding of cancer or other types of disease. One challenge in cancer treatment is to target therapies to pathogenetically distinct tumor types in order to maximize efficacy and minimize toxicity [19]. Thus cancer classification has been a central goal of gene expression analysis.

However, cancer classification based on gene expression data is not easy due to the characteristics of the data—high dimensionality and low sample size. The reduction in performance of an algorithm for data sets with many attributes is known as the curse of dimensionality [4]. To overcome the curse of dimensionality, we need to extract genes that are truly relevant to the disease status. This problem of identifying relevant features is known as feature selection.

Genes that are discriminative among different phenotypes are referred as informative genes. Figure 1.1 depicts informative genes with an example. Suppose there are 4 samples in a data set, where 2 samples are normal tissues and the other 2 samples are tumor samples. Each sample contains expression levels of 4 genes. Gene G1 and G2 are informative genes as they are consistent within each phenotype and discriminative between two phenotypes. In contrast, G3 and G4 are non-informative since they do not show difference between phenotypes. Informative genes are often referred as biomarkers. In the domain of our study, informative genes and biomarkers are interchangeable.

Improving the classification and prediction accuracy is not the only motivation for feature selection. Identification of informative genes can help researchers gain significant insights into the nature of a particular disease. Informative genes can be used in the development of efficient cancer diagnosis and classification platforms. In drug design, researchers record gene expression data for patient before, during and after treatment, aiming at identifying a subset of drug responsive genes

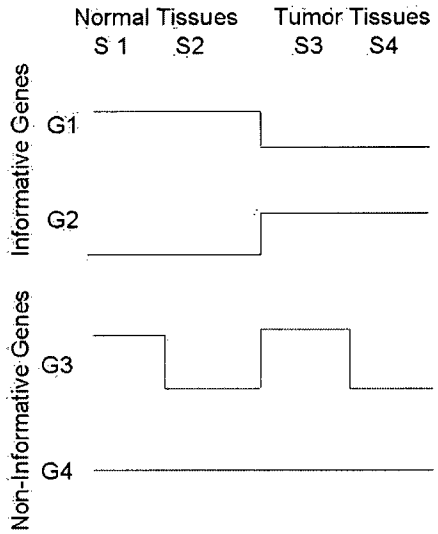


Figure 1.1: Illustration of informative genes. G1 and G2 are informative genes as they are consistent within each phenotype and discriminative between two phenotypes. G3 and G4 are non-informative since they do not show difference between phenotypes [13].

and identifying potential drug targets. Gene selection can also reduce the cost in diagnosis, because researchers only need to obtain profiles of relevant genes in order to predict class membership of a new patient. Smaller number of genes lead to cheaper chips, which in turn reduces the cost.

1.3 Problem Statement

In medicine, biomarkers are indicators of a particular disease state. In the context of gene expression analysis, biomarkers often stand for the minimum subset of informative genes. In this thesis, we focus on analyzing high-throughput gene expression data from two specific problem domains—cancer classification and biomarker identification.

Cancer Classification

Suppose we have a gene expression data set D of data points $x_i \in \{\mathbb{R}^d\}$ with binary

class labels $y_i \in \{-1, 1\}$, $D = \{(x_i, y_i)|i = 1, 2, ..., n\}$, $X = \{x_i|i = 1, 2, ...n\}$, $Y = \{y_i|i = 1, 2, ..., n\}$. Given this training data set, we wish to predict the class label y_* for a new data point $\mathbf{x}_*$.

Biomarker Identification

The quality of a subset of genes can be measured by several quality criteria including: test error and Leave-One-Out (LOO) validation error. Test error is defined as:

$$\text{Test error} = \sum_{i}^{N_{te}} \delta(\hat{y_i} \neq y_i) \tag{1.1}$$

where N_{te} is the number of test cases, and $\delta(s)$ is 1 if s is true, otherwise 0. In our case, $\delta(\hat{y_i} \neq y_i) = 1$ if the predicted class label $\hat{y_i}$ does not equal to the true label y_i of x_i. LOO validation error is evaluated as:

$$\text{LOO error} = \sum_{i}^{N_{LOO}} \delta(\hat{y_i} \neq y_i) \tag{1.2}$$

where N_{LOO} means the number of all validation cases. We discuss the LOO validation procedure in Chapter 2. Furthermore, we define test accuracy to be $(1 - \frac{\text{Test Error}}{N_{te}}) \times 100\%$, and similarly LOO accuracy to be $(1 - \frac{\text{LOO Error}}{N_{LOO}}) \times 100\%$.

Suppose we are given a training set D_{tr} and a test set D_{te}, and let $S = \{g_1, g_2, ..., g_d\}$ be the set containing all genes of the data sets D_{tr} and D_{te}. Ideally, we wish to identify a subset of features M such that $M \subseteq S$, and the test accuracy $Accu\_test(M)$ with features in M is the highest among the test accuracies using all possible subset of features. In the case where the highest accuracy is achieved by multiple subsets, we choose M to be the smallest subset. We refer M as the biomarker set, and the problem of identifying the biomarker set is called biomarker identification.

In real world problems, the test data is not available when selecting relevant features. Therefore, we cannot use test accuracies to identify biomarker sets. Instead, we can use the LOO accuracy on the training set to identify the biomarker

set. Specifically, we choose M to be the subset of features such that the LOO accuracy over the training set $Accu\_loo(M)$ with features in M is the highest among the LOO accuracies with all possible subset of features. Again, we use the size of the subsets to break the ties, and the smallest one wins.

Ideally, if the training set and the test set are drawn from the same underlying distribution, then we expect the biomarker set M_{tr} selected using LOO accuracy on the training data is identical to the true biomarker set M_{te} selected by the test accuracies.

1.4 Contribution

Our motivation in writing this thesis is to summarize the enormous amount of work that has been done in the field of kernel based methods with application to gene expression analysis. It is our aim to have all the results together in one place so that people can easily see their similarities and differences both theoretically and empirically.

Contributions of the thesis include: (1) We perform a comparative analysis of kernel based classification algorithms on gene expression data; (2) We perform a comparative analysis of kernel based feature selection methods with application to biomarker identification, and demonstrate experimentally the importance of gene selection; (3) We propose to use a stability measure, called *stability score*, to evaluate the sensitivity of feature selection methods to the changes in training data; (4) We propose an algorithm that utilizes a resampling method (e.g. cross validation), and demonstrate its effectiveness in improving the quality of biomarkers.

1.5 Thesis Outline

The remaining of the thesis is organized as follows. Chapter 2 defines related works and gives a number of definitions used in the context of our work. In Chapter 3, we review kernel based classifiers and provide experimental evaluation on their performance with gene expression data without feature selection. In Chapter 4, we review existing kernel based feature selection methods and show the comparison amongst these methods by experimental results. In Chapter 5, we show how to improve the stability and reliability of existing feature selection methods by cross validation, a special resampling method. We first present two existing algorithms that use this approach and then propose our new method. Finally, our conclusions are stated in Chapter 6, along with a number of suggestions for future work.

Chapter 2

Related Works

In this chapter, we give a general review of related works in the area of kernel based classification, feature selection, and cross validation. The concept of kernel and some of its basic properties are elaborated in Chapter 3 where we treat SVMs in more detail.

2.1 Kernel Based Classifiers

There are many classification models in the literature. Instead of reviewing them all, we focus on the classifiers used in this thesis: kernel based classifiers. Particularly, we place emphasis on SVM [5, 45], the Relevance Vector Machine (RVM) [44], and Gaussian Process Classifiers (GPC) [7, 8, 11, 25, 35, 38, 39, 47].

2.1.1 Kernel

Kernels provide a framework to represent data and must satisfy some conditions. Suppose we are given training samples $(x_1, y_1), ..., (x_n, y_n) \in \mathrm{X} \times \{\pm 1\}$, where X is a set from which the training cases x_i are taken. In classification and prediction, we need to generalize unseen data points. Given a new point $x \in \mathrm{X}$, we want to predict the corresponding $y \in \{\pm 1\}$. Intuitively, we want to choose y such that (x, y) is

in some sense similar to the training samples. Thus we need notions of similarity in X and in $\{\pm 1\}$ [42]. Measuring similarity of labels is easy as there are only two situations. Two labels are either identical or different. Similarity measure of inputs are not so obvious. Consider a similarity measures of the form:

$$k: \quad \mathrm{X} \times \mathrm{X} \rightarrow \mathbb{R} \tag{2.1}$$

$$(x_i, x_j) \rightarrow k(x_i, x_j) \tag{2.2}$$

that is, given two patterns, the function returns a real value characterizing their similarity. Such function k is called a *kernel* [42].

This general form is hard to study. A simple type of similarity measure is dot product. Obvious limitation of this kernel is that it is only defined when the data to be analyzed are vectors. For a more general object $\mathbf{x} \in \mathrm{X}$, we need to represent the object as a vector $\phi(\mathbf{x}) \in \mathbb{R}^p$, and then define a kernel for any $\mathbf{x}_i$, $\mathbf{x}_j \in \mathrm{X}$ by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \tag{2.3}$$

2.1.2 Relevant Kernel Methods

Support vector machine [45] is a generalization of the so-called optimal hyperplane algorithm. SVM algorithm creates a hyperplane that separates the data into two classes with the maximum margin. Unlike SVMs which use only support vectors to construct hyperplanes, Marron and Todd [31] use all vectors to construct hyperplanes aiming at solve the so-called "data piling" problem at the margin. Their algorithm is referred as Distance Weighted Discrimination (DWD).

One important limitation of SVM and DWD is that they make point prediction rather than generating predictive distributions. Tipping [44] formulated the Relevance Vector Machine (RVM), a probabilistic model whose functional form is equivalent to SVM. RVM achieves comparable prediction accuracy to SVM, yet also provides a full predictive distribution. RVM learns a classifier that is constructed

as a weighted linear combination of basis functions. Krishnapuram [22] argues the fact that RVM keeps too few basis functions may make it suffer from a systemic under-fitting. They propose Sparse Multinomial Logistic Regression (SMLR), which performs exact multinomial logistic regression with a sparsity-promoting prior.

Another important Bayesian kernel classifier is the Gaussian Process Classifier (GPC), which is derived from Gaussian process priors over functions. The main idea of the Gaussian process classifier is to assume that the class label y_i is obtained by transforming some real valued latent variable $f(x_i)$ associated with x_i. A Gaussian process prior is placed on $f(x_i)$, and is combined with the training data to obtain predictions for a new data point.

2.2 Feature Selection Methods

2.2.1 Feature Construction v.s. Feature Selection

Feature construction methods compute new features as a combination of the original ones and are often used in dimensionality reduction. Many popular feature construction techniques are based on a linear combination of the original features, such as Principal Component Analysis (PCA) which transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components that are ordered by decreasing variability. The first principal component is the combination of variables that explains the greatest amount of variation, and the second principal component accounts for the next largest amount of variation, and so on. Several nonlinear construction algorithms such as the one based on kernel methods [10] have been recently proposed. However, unfortunately, the common infeasibility of feature construction methods with application to gene expression analysis exist for both the linear and nonlinear case. Since the principal components are combinations of all input features, we lose the insights of which

subset of genes are most relevant [22]. Moreover, if all features have to be collected, the benefit of reducing diagnosis cost is lost.

Feature selection, on the other hand, chooses a subset of features from the original input space, which are supposed to be relevant to the task at hand. The subset of features can be combined in a proper way either in a subsequent stage or during feature selection [21]. The main point is that feature selection does not generate new features via a combination but selects features from the original input feature space. Therefore, only feature selection techniques address the problems we defined in Chapter 1—biomarker identification and diagnosis cost reduction. In subsequent sections, we review current approaches to feature selection in the literature. A good review article on feature selection is [18].

2.2.2 Methods for Feature Subset Selection

In the current literature, there are three basic approaches to the problem of feature subset selection in general, or biomarker identification in particular [18]. The three approaches include filter, wrapper, and embedded methods. *Filter* methods treat feature selection as an isolated step from the classifier design. Typically, a subset of features are selected as a preprocessing step, then a classifier is designed using only the features selected. *Wrapper* methods search an optimal subset of features with a classifier. They search through the space of possible feature subsets and score the quality of a particular subset S according to its predictive power (e.g. the accuracy of a classifier using only the subset S). Finally, *Embedded* methods perform feature selection in the process of training.

Filter Methods

Filter methods select features based on their ability to distinguish between the two classes without relying on a classifier. Statistical tests such as t-test and Wilcoxon

rank sum test are examples of the filter approach. In statistical tests, a ranking criterion is first introduced to quantify the discriminative capability of individual genes. Top ranked genes at a particular significant level are then returned as biomarkers. Fisher discriminant ratio (FDR) is an alternative to t-statistics [36]. In binary case, we may compute FDR for each feature i as:

$$FDR(i) = \frac{(\mu_+^{(i)} - \mu_-^{(i)})^2}{(\sigma_+^{(i)})^2 + (\sigma_-^{(i)})^2} \tag{2.4}$$

where $\mu_+^{(i)}$, $\mu_-^{(i)}$,$\sigma_+^{(i)}$,$\sigma_-^{(i)}$ are the mean and standard deviation of gene i on the positive and negative samples, respectively. Under the assumption that the class-conditional density functions are Gaussian, larger values of FDR(i) or t-statistic(i) suggest that feature i is better able to distinguish between two classes. Wilcoxon rank sum test works similarly, except that it does not assume any particular distribution of the training samples.

The major drawback of the filter approach is that it ignores the correlations between features. A set of low ranked features may perform well when combined together. On the other hand, a set of top ranked features may provide similar information for a prediction, and therefore their predictive power may not be better when combined together. Another pitfall of this approach is that data of low sample size may not reveal trustworthy information about their distribution.

Wrapper Methods

The limitation of filter methods which treat each feature isolated can be overcome by wrapper methods which consider feature selection as a search over the space of all possible feature subsets. Generally the wrapper methods consist in using prediction performance of a given classifier to assess the quality of subsets of features. An exhaustive search can be performed if the number of features is not too large. Brute force on high dimensions may become intractable. For example, there are

2^N possible feature subsets for a data set in N-dimensional space. In our case, N can easily be tens of thousands. Thus efficient search strategies, such as greedy search, can be used. Greedy search strategies come in two forms—forward selection and backward elimination. Moreover, one can use theoretical error bounds as an indicator of prediction performance. Guyon et al. [20] use this as a basis for assessing the importance of each feature at each iteration. Their algorithm, called Recursive Feature Elimination (RFE), is described in detail in Chapter 4 and 5.

Embedded Methods

Wrapper methods utilize learning machine (e.g. SVM classifier) as a black box to score subsets of variables according to their predictive power. This approach is universal and simple, but embedded methods that incorporate feature selection as part of the training process may be more efficient. In wrapper methods, to obtain the predictive performance, the data is split into training and test (or validation) samples. In embedded methods, all available data are used to train a classifier. An example of embedded methods is Automatic Relevance Determination (ARD) [27, 35] exploited by GPC. ARD parameters can be directly embedded into the covariance function of GPC, which can be considered a kernel that simplifies the problem in high dimensional space [35].

2.3 Cross Validation

Cross validation is a method of estimating the accuracy of a classification or regression model. There are mainly two types of cross validation: k-fold cross validation and Leave-One-Out (LOO) cross validation.

In k-fold cross validation, the data is partitioned into k folds (or subsets). One fold is in turn used for testing, and the remaining k-1 folds are used for training.

LOO cross validation is an extreme case of k-fold cross validation, with $k = n$, the number of total cases. Cross validation can be considered as a special form of resampling in the sense that all data are used for training—none has to be held back in a separate test set..

Chapter 3

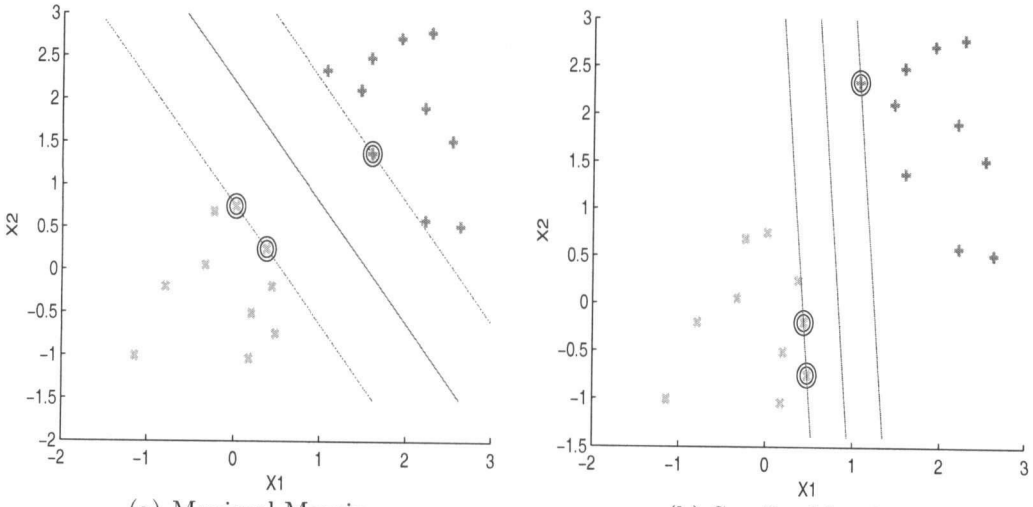
Evaluating Relevant Kernel Classification Algorithms

Although Chapter 2 introduced related work and explored various kernel classification algorithms, we use this chapter to extend the discussion and to evaluate these algorithms with application to tissue classification using gene expression data. In chapter , we shall discuss how to improve classification (or diagnosis) accuracy by incorporating feature selection techniques.

We focus on binary classification for simplicity. For an extensive study of multi-class tissue classification using gene expression, please refer to [26]. Let us assume a data set D of data points $x_i \in \mathbb{R}^d$ with binary class labels $y_i \in \{-1, 1\}$, $D = \{(x_i, y_i)|i = 1, 2, ..., n\}$, $X = \{x_i|i = 1, 2, ...n\}$, $Y = \{y_i|i = 1, 2, ..., n\}$. Given this training data set, we wish to predict the class label y_* for a new data point x_*. We discuss how various classifiers solve this problem in the following sections.

3.1 Support Vector Machines

In this section, we present some basic natures of SVM algorithm. More theoretical background information can be found in [12, 42, 45].



(a) Maximal Margin (b) Smaller Margin

Figure 3.1: Two dimensional linearly separable data. Both classifiers correctly classify the data, but the margin in (a) is bigger than the one in (b). The classifier in (a) is the maximal margin classifier of this data set.

3.1.1 Maximal margin classifier

Sometimes the data are linearly separable as illustrated in Figure 3.1. A data set is linearly separable if there exists a line (2-D), a plane (3-D), or a hyperplane (4-D or higher) which correctly separates the two groups of data such that data from same group all fall on the same side, and no two data points from different groups fall on the same side. We make this assumption for now. We shall discuss the non-separable case in section 3.1.3.

For a specific data set, there may exist multiple classifiers that can correctly separate the two classes. For example, in Figure 3.1, both classifiers correctly separate all the data points. In maximal margin classification, an optimal linear classifier is the one that maximizes the margin, which is defined as the minimum distance of the data objects to the decision boundary. In this case, the classifier on the left is the maximal margin classifier. The definition of *margin* is given in Section 3.1.2 (see Figure 3.2).

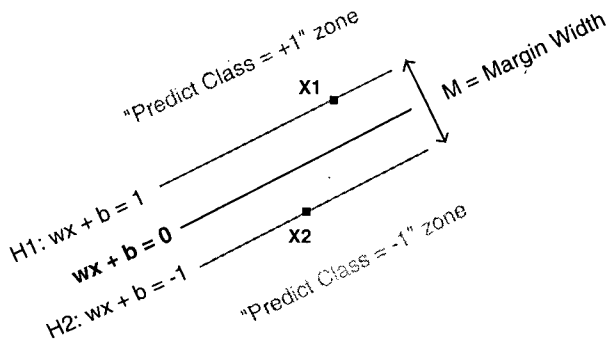


Figure 3.2: Linear SVM—the simplest case. H1 and H2 are hyperplanes, and X1 and X2 are support vectors.

3.1.2 Linear SVM

A 2-D example of linear SVM is illustrated in Figure 3.2. Mathematically we have:

$$x_i \cdot w + b \geq +1 \text{ for } y_i = +1$$
$$x_i \cdot w + b \leq -1 \text{ for } y_i = -1,$$

or in the combined form:

$$y_i(x_i \cdot w + b) \geq 1 \tag{3.1}$$

Furthermore, we suppose x_1 and x_2 are two vectors that lie on the hyperplanes H_1 and H_2 respectively, then we have:

$$H_1 : x_1 \cdot w + b = +1$$
$$H_2 : x_2 \cdot w + b = -1$$

Hence, the margin can be computed as:

$$M = \left| \frac{(x_1 - x_2) \cdot w}{|w|} \right| = \frac{2}{|w|} \tag{3.2}$$

Now we can form the constrained optimization problem as:

$$\begin{aligned} \min \quad & M = w^T w \\ \text{s.t.} \quad & (x_i \cdot w + b) y_i \geq 1, i = 1, ..., n \end{aligned} \tag{3.3}$$

Therefore, we need to optimize a quadratic function subject to linear constraints. The solution involves constructing a dual problem where a Lagarange multiplier α_i is associated with every constraint in the primary problem.

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i [y_i (x_i w + b) - 1] \tag{3.4}$$

The corresponding Karush-Kuhn-Tucker (KKT) conditions are:

1. $\bigtriangledown L = 0$
2. $y_i(x_i \cdot w + b) \geq 1$
3. $\alpha_i \geq 0$
4. $\alpha_i [y_i (x_i \cdot w + b) - 1] = 0$

The dual problem (3.4) can now be written as:

$$\begin{array}{ll} \max_{\alpha \geq 0} & [\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y_i (x_i^T x_j) y_j \alpha_j] \\ \text{s.t.} & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{array} \tag{3.5}$$

The last KKT condition implies that only the α_i's corresponding to points at the boundary can have non-zero values. These N_s points at the boundary are called *support vectors* since they support the hyperplane on both sides of the margin. Furthermore, only these N_s support vectors are needed to specify the optimal linear classification boundary.

For a new data object x^*, we predict its label as follows:

$$\hat{y}^* = sign(wx^* + b) = sign \left[\sum_{i=1}^{N_s} \alpha_i y_i (x_i^T x^*) + b \right] \tag{3.6}$$

A nice property of SVM is that during both training (Equation 3.5) and testing (Equation 3.6), data points always appear in the form of dot products. We shall see how this property make it possible to reduce the computational cost by kernelization in section 3.1.3.

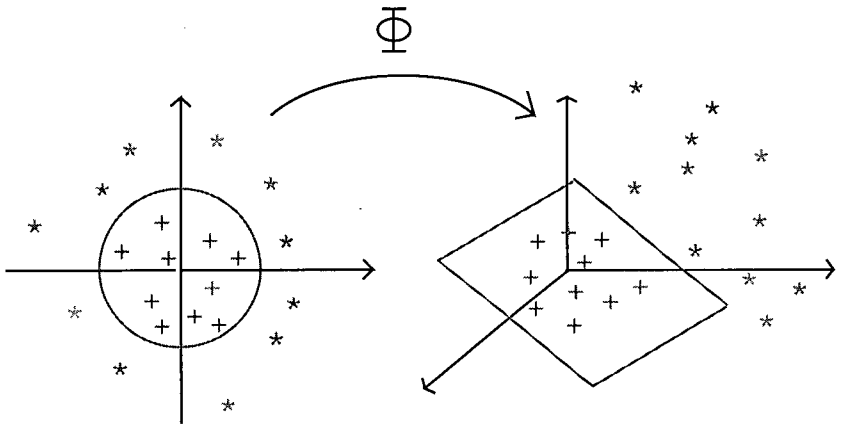


Figure 3.3: Non-linear SVM. Data on the left are not linearly separable in the two dimensional space. Via some transformation, say Φ, data points in 2D can be transformed to 3D feature space where they become linearly separable.

3.1.3 Nonlinear SVM

In many real world applications, data are not linearly separable in the original input space as illustrated in Figure 3.3. To solve this problem, we can map the data to a higher-dimensional space, where it becomes linearly separable. Intuitively, the higher the dimensionality, the sparser the distribution of the data, and the more likely to separate the data linearly.

Kernel trick

Recall that the linear classifier only relies on dot product between vectors in both training (Equation 3.5) and prediction (Equation 3.6). If every data point is mapped into some high-dimensional space via some transformation $\Phi : x \rightarrow \phi(x)$, then the dot product becomes: $\phi(x_i)^T\phi(x_j)$. Similarly, the $\phi(x)$ appears only in the form of dot products. Mercer's theorem states that, under some mild conditions, dot product in some expanded higher dimensional feature space correspond to kernel functions defined in the lower dimensional input space, that is: $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$.

Hence, to train and test SVMs, we do not need to know Φ. We only need to choose an appropriate kernel function. Here are some commonly used kernel functions:

- Linear kernel

  $K(x_i, x_j) = {x_i}^T x_j$

- Polynomial kernel

  $K(x_i, x_i) = ({x_i}^T x_j + 1)^p$

- Gaussian kernel (a.k.a. rbf kernel)

  $K(x_i, x_j) = \exp(-\frac{1}{2\sigma^2}(x_i - x_j)^T(x_i - x_j))$

- Sigmoid

  $K(x_i, x_i) = tanh(\alpha {x_i}^T x_j - \beta)$

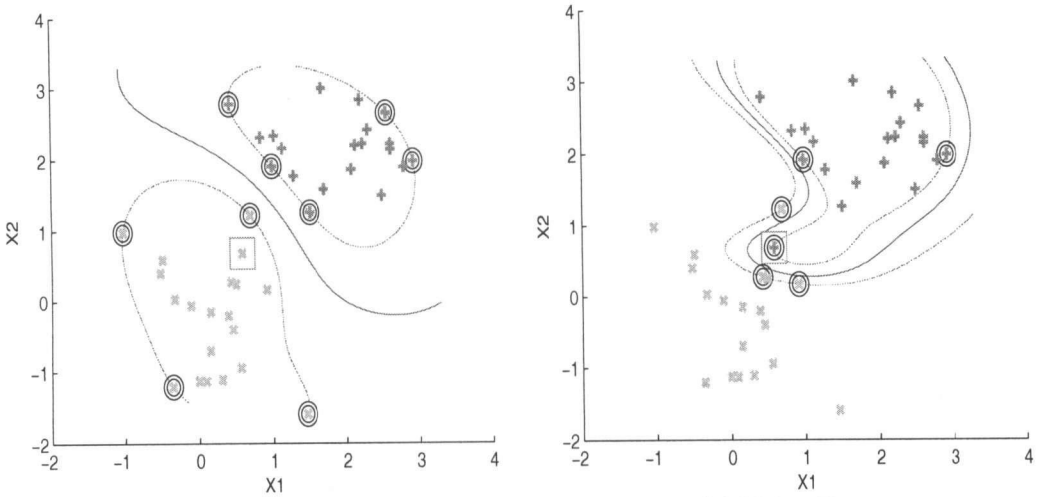
Hard margin v.s. soft margin

In many real world applications, especially in gene expression analysis, the data contains some noise which makes the data non-separable. If we use the nonlinear svm techniques discussed above to deal with this type of non-separability with the constraints that all data must be correctly classified, overfitting problem may incur as illustrated in Figure 3.4.

Slack variables ξ can be added to allow misclassification of diffcult or noisy examples to improve the generalization of the classifier. This is called C-SVC [45]. The quadratic optimization incorporating slack variables is formulated as:

$$\begin{aligned} \min_{w,b,\xi} \quad & \tfrac{1}{2} w^T w + C \textstyle\sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & y(w^T x_i + b) \geq 1 - \xi_i, i = 1, ..., n \\ \text{and} \quad & \xi_i \geq 0 \end{aligned} \tag{3.7}$$

where $\xi_i \geq 0$ for all i. The parameter C in (3.7) can be viewed as a penalty factor which controls overfitting.



(a) True Data (b) Noisy Data

Figure 3.4: Overfitting problem of hard margin with noisy data. Suppose the data in (a) are correct. If we change the class label of one data point (e.g. the one marked by the red square), we get the separating hyperplane as shown in (b) with hard margin. The hyperplane in (b) overfits the noisy data, and has a poor generalization ability.

In summary, SVM classifier has the following nice properties among many others:

- Its flexibility in choosing a similarity function: By Mercer's Theorem, every semi-positive definite symmetric function is a kernel, and any valid kernel function can be used as the similarity measure.
- Its sparseness of solution when dealing with large data sets—only support vectors are used to specify the separating hyperplane.
- Its ability to handle large feature spaces—kernelizaton reduces the computation cost, and its complexity does not depend on the dimensionality of the feature space.
- Overfitting can be controlled by soft margin approach.
- Its nice math property: a simple convex optimization problem which is guar-

anteed to converge to a single global solution.

3.2 Distance Weighted Discrimination

As discussed in Chapter 1, gene expression data produced by high-throughput techniques are characterized by High Dimensionality and Low Sample Size (HDLSS). Marron and Todd [31] presents a novel view of the performance of SVM in HDLSS settings via projecting the data onto the normal vector of the separating hyperplane. This view reveals substantial so-called "data piling" problem at the margin. They argue that data piling may adversely affect the generalization performance of SVM classifiers. To avoid the data piling problem, they propose a new classification algorithm called "Distance Weighted Discrimination" (DWD).

The major difference between DWD and SVM is that the former uses all training vectors to define the classifier (or separating hyperplane), while the latter only uses support vectors. Recall that SVMs try to find the classifier that maximizes the minimum distance of the data objects to the decision boundary as illustrated in Equation 3.3. In DWD, an optimal linear classifier is the one that minimizes the sum of the reciprocals of the residues, perturbed by a penalized vector ξ. Mathematically, DWD forms the following optimization problem:

$$\begin{aligned} \min_{r,w,\beta,\xi} \quad & \sum_{i=1}^{n} \frac{1}{r_i} + C \sum_{i=1}^{N} \xi_i \\ \text{s.t} \quad & r_i = y_i(x_i^T w + b) + \xi_i \\ & w^T w = 1, r \geq 0, \xi \geq 0 \end{aligned}$$

Again, C is a penalty parameter to control misclassification and overfitting. The equation above can be reformulated into the form of of a so-called second-order cone programming (SOCP) problem. We skip the SOCP formulation for DWD since optimization is not the major topic of the thesis. Interested readers can refere to

[31] for the formulation and derivation of SOCP for DWD.

3.3 Bayesian Kernel Classifiers

In this section, we present the probabilistic, or Bayesian approach to learning kernel classifiers. The Bayesian approach exhibits some differences with respect to the framework of risk minimization. Two key distinctions include: (1) In the Bayesian approach, one can incorporate prior knowledge into the process of estimation; (2) In contrast to SVM, which simply returns a binary decision, yes or no, a probabilistic classifier returns the probability, $P(y = 1|\mathbf{x}$, that an object x belongs to the class of interest indicated by the binary variable y. The probability answer is more desirable than a simple binary decision as it provides additional information about the certainty of the prediction. A binary classifier returns two probabilities with one for each class, and the final class label is decided by choosing the one with higher probability.

The following two subsections present two Bayesian learning algorithms—Gaussian processes and Relevance Vector Machines (RVM).

3.3.1 Gaussian Processes Classifiers

Formally, we have the following definition for Gaussian process (GP):

Definition 3.1 (Gaussian Process) *Given an index set $\mathcal{X}$, a collection of random variables $f(x)$ with $x \in \mathcal{X}$ is a Gaussian process if, for every finite set $\{x_1, ..., x_n\} \subset \mathcal{X}$, the random variable $\{f(x_1), ..., f(x_n)\}$ has a multivariate Gaussian distribution, with mean $\mu \in \mathcal{R}^n$ and covariance $K \in \mathcal{R}^{n \times n}$.*

Gaussian processes were originally developed for the problem of regression. In regression, the output (or dependent) variable is a real value rather than a binary number representing the class membership. In order to adapt this method to the

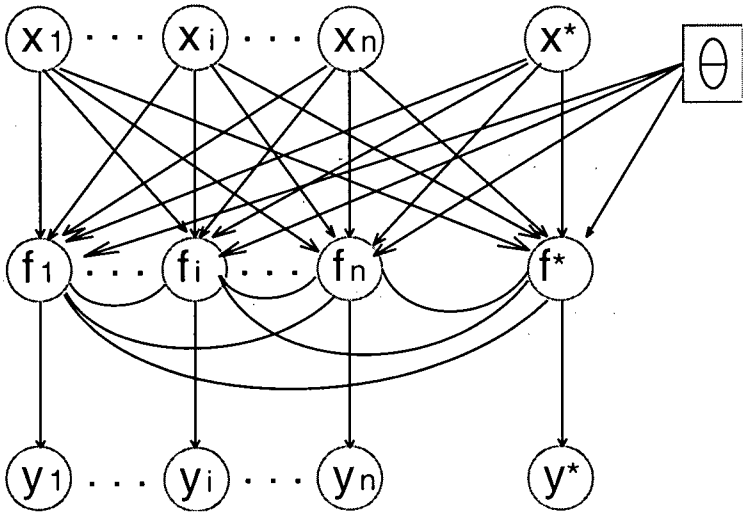


Figure 3.5: Graphical model for GPC with n training data points and one test data point. $X_{1:n}$ and $Y_{1:n}$ are observed. Given x^*, we wish to predict y^*. f_i is a latent variable associated with x_i, and y_i is obtained by transforming f_i. Given f, x and y are independent. f_i and f^* are joint Gaussian, and the Gaussian process prior is parameterized by Θ [25].

problem of classification, the concept of latent variables are introduced. The main idea of Gaussian process classifier (GPC) is to assume that the class label y_i is obtained by transforming some real valued latent variable $f(x_i)$ associated with x_i. The graphical model for GPC is shown in Figure 3.5. This graphical model encodes the assumption that x and y are independent given f. A bayesian framework of GPC is described with more details in the following.

Gaussian process prior

We place a Gaussian process prior on the function $f(\cdot)$, meaning that for any finite set $X = \{x_1, ..., x_m\}$, the random vector $\boldsymbol{f} = [f(x_1), ..., f(x_m)]^T$ is a Gaussian. Without loss of generality, we can assume such a process has a zero mean. The covariance between $f(x_i)$ and $f(x_j)$ can be defined as

$$\Sigma_{ij} = cov(x_i, x_j) = v_0 \exp\left\{-\frac{1}{2}\sum_{m=1}^{d} l_m(x_i^m - x_j^m)^2\right\} + v_1 + v_2\delta(i,j), \qquad (3.8)$$

We say this GP prior is parameterized by $\boldsymbol{\theta}$ which is also known as hyper-parameters. More explicitly, if we use the covariance function defined in 3.8, then $\boldsymbol{\theta} = [v_0, l_m, v_1, v_2]$ is a hyperparameter vector. In particular, the hyperparameter v_0 specifies the overall vertical scale of variation of the latent values, v_1 is the overall bias of the latent values from zero mean, v_2 is the latent noise variance, and l_m is the Automatic Relevance Determination (ARD) variable for the m-th feature that controls the contribution of this feature in the modelling. We shall describe ARD in detail in Chapter 4 when we discuss the problem of feature selection. We assume $l_m = 1$ for all m in this chapter. In fact, $\Sigma_{ij} = K(x_i, x_j)$ can be any kernel function such as the four listed in section 3.1.3.

By the definition of Gaussian Process, the prior probability of these latent function values $\{f(x_i)\}$ given $\boldsymbol{\theta}$ is a multivariate Gaussian, that is $p(\boldsymbol{f}|\mathbf{x}, \boldsymbol{\theta}) \sim N(0, \Sigma)$:

$$p(\boldsymbol{f}|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f}) \tag{3.9}$$

Likelihood for class label

The likelihood $p(D|\boldsymbol{f}, \boldsymbol{\theta})$ is the joint probability of observing the sample labels given the latent function values. The likelihood can be evaluated as a product of the likelihood function:

$$p(D|\boldsymbol{f}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|f_i, \boldsymbol{\theta}) \tag{3.10}$$

We give two commonly used likelihood functions below.

1. **Probit function**([30], [7], [35]):

$$p(y_i|f_i, \boldsymbol{\theta}) = \Phi(y_i f_i) \tag{3.11}$$

where Φ is the cumulative distribution function (c.d.f.) of standard Gaussian distribution $N(0, 1)$. In the presence of noise from inputs or targets, we may assume that

the latent function values are contaminated by a Gaussian noise which is independent of inputs. If we use δ to denote the noise, then δ has zero mean and an unknown variance σ^2, i.e. $N(\delta; 0, \sigma^2)$. The likelihood function becomes $p(y_i|f_i) = \Phi(\frac{y_i f_i}{\sigma})$;

2. **Logistic function**: This is given by

$$p(y = 1|f_i, \boldsymbol{\theta}) = \frac{\exp(f_i)}{1 + \exp(f_i)} \tag{3.12}$$

Posterior probability

The posterior probability can be written as

$$p(\boldsymbol{f}|D, \boldsymbol{\theta}) = \frac{1}{p(D|\boldsymbol{\theta})} \prod_{i=1}^{n} p(y_i|f_i, \boldsymbol{\theta}) p(\boldsymbol{f}|\mathbf{x}, \boldsymbol{\theta}) \tag{3.13}$$

where the prior probability $p(\boldsymbol{f}|\mathbf{x}, \boldsymbol{\theta})$ is defined as in (3.9), and the likelihood function $p(y_i|f_i, \boldsymbol{\theta})$ is defined as in (3.11) and (3.12). The normalization factor $p(D|\boldsymbol{\theta})$ is known as the evidence for $\boldsymbol{\theta}$, and is defined as $p(D|\boldsymbol{\theta}) = \int p(D|\boldsymbol{f}, \boldsymbol{\theta}) p(\boldsymbol{f}|\mathbf{x}, \boldsymbol{\theta}) d\boldsymbol{f}$. A popular idea to compute the evidence is to approximate the posterior distribution $p(\boldsymbol{f}|D, \boldsymbol{\theta})$ as a Gaussian using Laplace approximation or Expectation Propagation (EP). Then the evidence can be calculated analytically ([27], [30]).

Model adaptation

In the full Bayesian treatment, the class probability is obtained by integrating over the hyperparameters $p(y_*|x_*, D) = \int p(y_*|x_*, D, \boldsymbol{\theta}) p(\boldsymbol{\theta}|D) d\boldsymbol{\theta}$. Monte Carlo methods [35] can be used to approximate the integral. However, this might be too costly to use in practice. Therefore, in this thesis, rather than integrating over the hyperparameters we fix them by maximizing the posterior probability $p(\boldsymbol{\theta}|D)$, where $p(\boldsymbol{\theta}|D) \propto p(D|\boldsymbol{\theta}) p(\boldsymbol{\theta})$. The prior distribution $p(\boldsymbol{\theta})$ can be specified by domain knowledge, or some vague uninformative distribution. In the latter case, the optimal values inferred by Maximizing A Posterior (MAP) will converge to the ones inferred by Maximizing the marginal Likelihood (ML).

Prediction

Suppose we have found the optimal settings of hyperparameters θ^*, then let us take a test sample x_*, for which the class label y_* is unknown. By the definition of Gaussian process, the latent variable $f(x_*)$ and the $\boldsymbol{f} = [f(x_1), ..., f(x_n)]^T$ have a joint multivariate Gaussian distribution, i.e.

$$\begin{bmatrix} \boldsymbol{f} \\ f_* \end{bmatrix} \sim \mathcal{N}\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma & \boldsymbol{k} \\ \boldsymbol{k}^T & \mathcal{K}(x_*, x_*) \end{pmatrix} \right]$$

where $\boldsymbol{k} = [\mathcal{K}(x_*, x_1), \mathcal{K}(x_*, x_2), ..., \mathcal{K}(x_*, x_n)]^T$. The conditional distribution of $f(x_*)$ given $\boldsymbol{f}$ is also a Gaussian:

$$p(f_*|\boldsymbol{f}, D, \boldsymbol{\theta}^*) = \frac{p(f_*, \boldsymbol{f}|x_*, D, \boldsymbol{\theta}^*)}{p(\boldsymbol{f}|\mathbf{x}, \boldsymbol{\theta}^*)} \propto \exp\left(-\frac{1}{2}\frac{(f(x_*) - \boldsymbol{f}^T\Sigma^{-1}\boldsymbol{k})^2}{\mathcal{K}(x_*, x_*) - \boldsymbol{k}^T\Sigma^{-1}\boldsymbol{k}}\right) \tag{3.14}$$

The predictive distribution of $P(f(x_*)|D, \theta^*)$ can be computed as

$$p(f_*|D, \boldsymbol{\theta}^*) = \int p(f_*|\boldsymbol{f}, D, \boldsymbol{\theta}^*)p(\boldsymbol{f}|D, \boldsymbol{\theta}^*)d\boldsymbol{f} \tag{3.15}$$

As both terms of the integrand are (or can be approximated as) Gaussians, the predictive distribution (3.15) can be simplified as a Gaussian $N(f(x_*); \mu_{x_*}, {\sigma_{x_*}}^2)$ with mean μ_{x_*} and variance ${\sigma_{x_*}}^2$. Refer to [7] for explicit forms of μ_{x_*} and ${\sigma_{x_*}}^2$. The class label y_{x_*} can then be decided as $\arg\max_i p(y_* = i|x_*, D, \boldsymbol{\theta}^*)$.

3.3.2 Relevance Vector Machines and Other Sparse Classifiers

Sparse classification algorithms obtain sparse solutions for regression and classification while maintaining their Bayesian interpretation. The family of sparse classification algorithms includes Relevance Vector Machines (RVM) [44], sparse online Gaussian processes [11], Sparse Multinomial Logistic Regression (SMLR) [23], and some others. We include RVM and SMLR in our comparative study.

By exploiting a probabilistic Bayesian framework, RVM learns a classifier that is constructed as a weighted linear combination of basis functions. The weights

Name	Number of Samples	Dimensionality	Type of source
Colon cancer	62(22+/40-)	2000	microarrays
Leukemia	72(25+/47-)	7129	microarrays
Lung cancer	26(11+/15-)	19624	SAGE

Table 3.1: Gene expression data used for classification

are estimated by ML or MAP in the presence of training data. There is no restriction on basis functions. For example, if we use a kernel function centered on the training samples, the learned classifier will be similar to SVM. If we use original features or their transformed form as the basis functions, then the learned classifier can be considered as a feature weighting algorithm. We shall discuss the performance of RVM for feature selection in Chapter 4.

We omit the details of the Bayesian framework of RVM as it is not the core issue concerned by this thesis. For a full derivation of RVM, please refer to [44].

3.4 Experimental Results

We conduct an experimental evaluation on three real gene expression data sets. Table 3.1 provides a summary of those reported here.

The Colon cancer data set, originally studied by Alon et al. [1], contains the expression levels of the 2000 genes from 22 normal and 40 tumor tissues. The task is to discriminate tumor from normal tissues.

The Leukemia data set, originally analyzed by Golub et al. [19], consists of expression values of 7129 genes from 47 samples of Acute Lymphoblastic Leukemia (ALL) and 25 samples of Acute Myeloid Leukemia (AML). The task is to discriminate the two types of leukemia.

The Lung cancer data set, generated by BC Cancer Regency, consists of 26 SAGE libraries (samples) with 15 normal libraries, 5 CIS (carcinoma institu)

libraries, and 6 invasive libraries. The normal libraries are from bronchial brushings, the CIS libraries are from bronchial biopsies, and the invasive libraries are from frozen resected samples. The values are the sum of all the tag counts that map to a particular gene. There are a total of 19,624 genes including a small number of hypothetical genes. The task is to discriminate normal samples from CIS/INV ones.

In keeping with standard practice on kernel classifiers, we normalize the original expression data sets so that the mean is zero and the standard deviation is one. We use several off-the-shelf implementations including OSU-SVM 3.00 [1] for SVM, RVM code from B. Krishnapuram [23], and W. Chu's C code for GPC [7].

The specific form of kernel was selected by cross-validation: the one giving the best result is used for all classifiers to allow comparison. Kernel parameters were selected by cross-validation using SVM, and the best parameters were used for all methods. Regularization parameters are method-specific and therefore was chosen by cross-validation for each method. In our experiment, a linear kernel was used for all three data sets.

3.4.1 Diagnosis Accuracy Comparison

Diagnostic accuracies are commonly assessed by cross validation. We use k-fold cross validation with $k = 10$. More specifically, we randomly divide the data set into 10 partitions (or folds), and ensure that each partition contains an equal number (if possible) of samples from a class. For example, we divide the Colon cancer data set into 10 folds such that each fold contains 4 tumor samples and 2 or 3 normal samples. With the 10 partitions ready, we pass the data to classifiers 10 times. Each time, one fold is used for testing, and the remaining $k - 1 = 9$ folds are used for training. The average accuracy from the 10 experiments are reported in Figure 3.6. Neither classifier is incorporated with feature selection in these experiments. We

[1] Available from http://www.ece.osu.edu/~maj/osu_svm

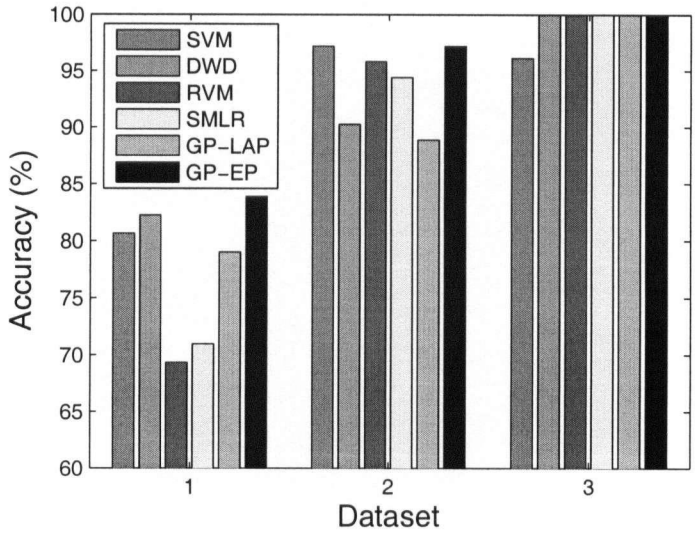


Figure 3.6: 10-fold cross-validation accuracies of various classifiers without feature selection

---

make the following observations from the results in Figure 3.6:

- Among the three data sets, Colon data set is the hardest to classify for all classifiers. This is likely caused by the low quality of the data: the expression values may contain lots of noise, and some class labels might be mislabelled. Lung cancer data has the lowest sample size and highest dimensionality, but it is predicted most correctly for all classifiers. This is mainly due to its high quality.

- Among the classification algorithms, Gaussian processes with EP outperforms all other classifiers as it achieves the highest accuracies for all data sets. However, the difference is not significant.

- Notice SVM is the only classifier that failed to achieve 100% accuracy for the Lung cancer data. The reason behind this might be the *data-piling* problem discussed in section 3.2.

		True Class	
		positive	negative
Classified As	positive	**T**rue **P**ositives	**F**alse **P**ositives
	negative	**F**alse **N**egatives	**T**rue **N**egatives

Table 3.2: ROC confusion matrix

- Sparse classification algorithms (e.g. RVM and SMLR) obtained the lowest accuracies for the Colon data. This may imply that sparse classification algorithms are heavily affected by irrelevant features.

- Besides the effect of the quality of the data sets, the presence of a significant number of irrelevant features may make the classification somewhat prone to the *curse of dimensionality*. We shall see in Chapter 4 that diagnostic accuracies can be improved dramatically after eliminating irrelevant features.

3.4.2 ROC Curve Analysis

In this subsection, we use Receiver Operating Characteristics (ROC) curves to evaluate and compare classifiers. An ROC curve is a useful technique for visualizing, organizing and selecting classifiers based on their performance [15].

In binary classification, given a classifier and an object to be classified, there are four possible outcomes. If the object is positive, and it is correctly classified as positive, it is counted as a *true positive*; if it is misclassified as negative, it is counted as *false negative*. Similarly, if the object is negative, and it is correctly classified as negative, it is counted as *true negative*; if it is misclassified as positive, it is counted as *false negative*. Table 3.2 shows a two by two confusion matrix that summarizes the four outcomes.

Let TP, FP, TN and FN be the total number of true positives, false positives, true negatives and false negatives, respectively. Furthermore, let P and N be the total number of positive and negative objects respectively. Then we have the

following terms associated with ROC curves:

- *True Positive Rate* (also called *hit rate* and *recall*): TP rate $= \frac{TP}{P}$
- *False Positive Rate* (also called *false alarm rate*): FP rate $= \frac{FP}{N}$

ROC graphs are two-dimensional graphs with TP rate as Y-axis and FP rate as X-axis. A *discrete classifier* (e.g. decision tree) is a classifier that only outputs class labels [15]. Given a test set and a discrete classifier, we can compute the pair of (FP rate, TP rate), which corresponds to a single point in ROC space.

Rather than a discrete class label, some classifiers like the six we consider here, give each object a numerical value that represents the degree to which the object belongs to a class. The value can be a strict probability, or an uncalibrated score. Specifically, SVM and DWD give uncalibrated scores, in which case the only property that holds is that higher scores indicates higher probability that an object is of a class. As opposite to SVM and DWD, the remaining four Bayesian classifiers give a strict probability that an object belongs to a class. Any of the six classifiers can be turned into a discrete classifier by thresholding: if the probability or score is above the threshold, the classifier produces a *yes*; otherwise, a *no*. By varying the threshold, we can produce different points in the ROC space.

In this thesis, we adopt the method described in Algorithm 2 of the paper [15] and generate the ROC curves as follows: (1) merge test sets $T_1, T_2, ..., T_{10}$, generated from 10-fold cross-validation, into one large test set T; (2) sort T by their scores (or probabilities); (3) let threshold to be $+\infty$ and produces the point $(0, 0)$; (4) reduce the threshold from highest score to lowest score, and compute (FP rate, TP rate) pair for each threshold; (5) connect the points to generate a ROC curve.

ROC curves show the tradeoff between benefits (TP) and costs (FP). The closer the curve follows the left-hand border and then the top border, the more accurate the test; the closer the curve follows the 45-degree diagonal, the less accurate

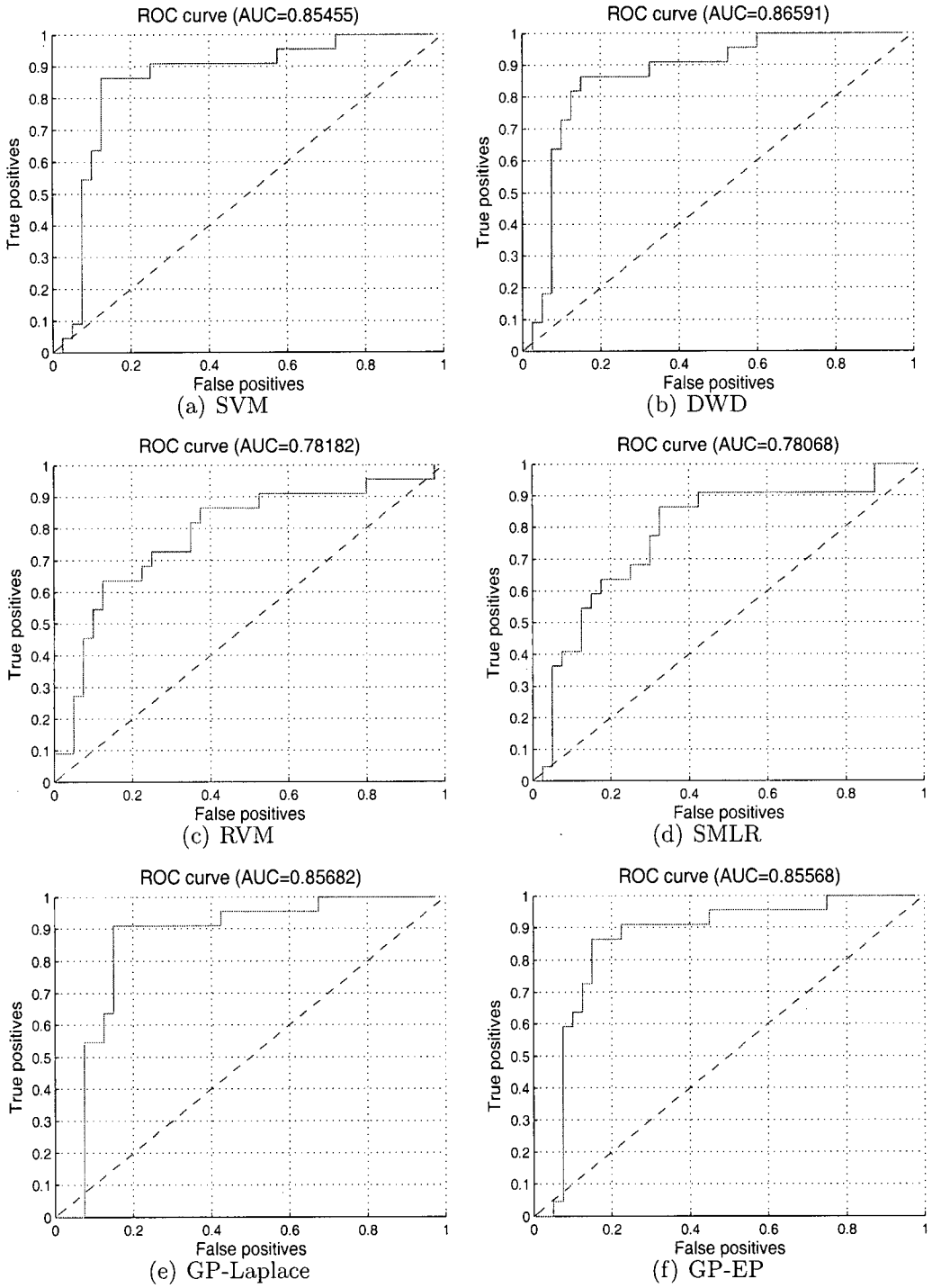


Figure 3.7: ROC curves of the six classifiers for the Colon cancer data

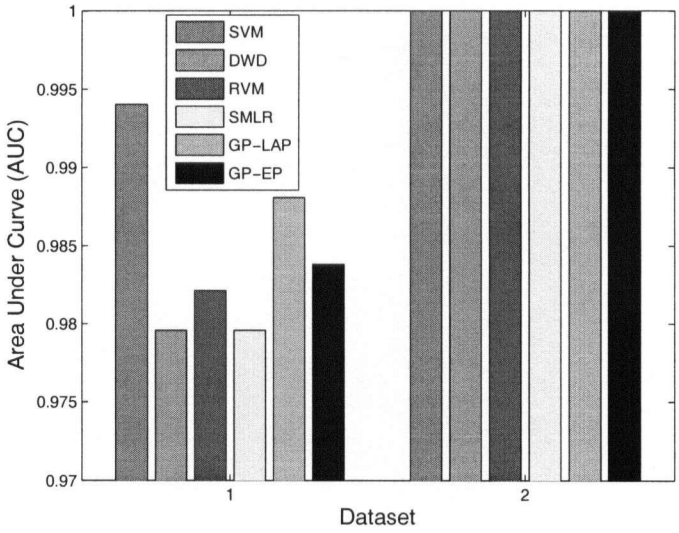


Figure 3.8: AUC values of various classifiers for the Leukemia (1) and Lung cancer data (2)

---

the test.

In addition to ROC curves, we can use a single scalar value—the Area Under the ROC Curve (AUC)—to represent the classifier performance [33]. The AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Therefore, higher AUC indicates better average performance.

Figure 3.7 shows the ROC curves of various classifiers with Colon data set. As our ROC curves are generated from a low-sample-sized data set, the curves in Figure 3.7 are step functions with sharp corners. This makes the AUC values only an estimate of the true probability. Therefore, when using these AUC values to compare two classifiers, one must be aware that the difference may not be statistically significant. Having said that, an AUC value gives a good general measure of predictiveness. As shown in Figure 3.7, AUC values given by SVM, DWD, GP-Laplace and GP-EP are comparable (≈ 0.85), and dominate the AUC values of

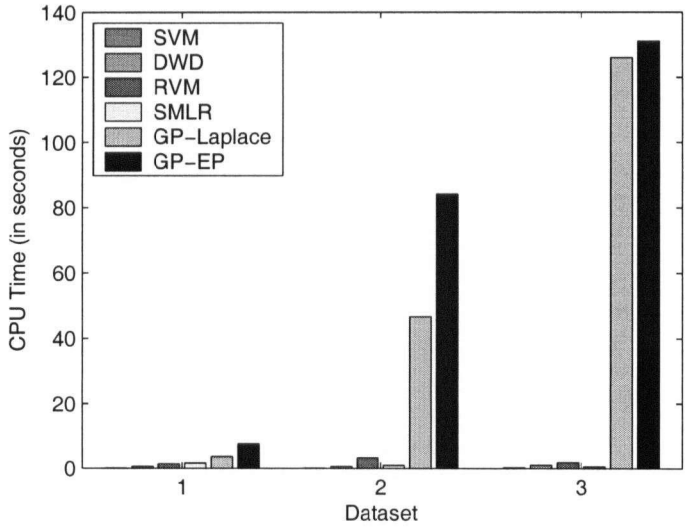


Figure 3.9: Graphical summary of CPU costs of various classifiers

RVM and SMLR, both of which are approximately 0.78. ROC curves of various classifiers with either Leukemia or Lung cancer data exhibit similar shapes, so we omit them here. Figure 3.8 presents a summary of AUC values of various classifiers with Leukemia and Lung cancer data. As Figure 3.8 shows, all classifiers achieve comparably excellent performance with these two data sets.

3.4.3 CPU Cost Comparison

The CPU time taken by the six classifiers considered in this chapter, over the three data sets, is summarized in the graphical display of Figure 3.9. The bars are grouped by data set with 1, 2 and 3 representing Colon cancer, Leukemia and Lung cancer data set respectively. The color of the bars indicate the classifier, and the height shows the CPU time (in seconds) consumed by each classifier.

The machine used was an Intel PC with a single 2.66GHz processor and a 1GB RAM. The CPU time reported here is the average time of the 10-fold cross validation experiments. In other words, the CPU time reported is the average

Classifier	Programming Languages Used	Source of Code
SVM	C++ (Matlab mex interface)	[32, 6]
DWD	Matlab	[31]
RVM	Matlab	[23]
SMLR	Matlab	[23]
GPC with Laplace	C	[7]
GPC with EP	C	[7]

Table 3.3: 10-fold cross-validation accuracy without feature selection

time needed to train with 9-fold data and to test with the remaining 1-fold data. Additionally, for all methods, this CPU time includes the time of loading the data.

From some point of view, it is not fair to compare the CPU costs as these classification methods are not implemented in the same programming languages. We report the CPU costs here not aiming to compare the absolute values but to provide some information about the running times of various classifiers. What is important is whether the running time is acceptable, and whether one CPU cost is significantly different from another while the two have comparable accuracies. A summary of the implementations are provided in Table 3.3. All six classification methods have reasonably low CPU cost with the maximum cost ($\approx$ 131 seconds) occurs when classifying the Lung cancer data (dimensionality = 19624) using Gaussian processes with EP. For all three data sets, SVM, DWD, RVM and SMLR give very similar low cost regardless dimensionality of the data. GP classifiers have higher costs than the other four classifiers over all three data sets. The difference becomes significant when dealing with the two data sets with higher dimensions. It is safe to say that the former four classifiers have much better scalability than Gaussian process classifiers.

Chapter 4

Comparing Feature Selection Methods for Kernel Machines

As mentioned in Chapter 1, there are often tens of thousands of genes in a gene expression data set generated by high-throughput technologies such as microarray and SAGE, but probably only a small subset of informative genes are relevant to the phenotypes under investigation. Identifying informative genes is a very important research problem in bioinformatic research. For example, informative genes can be used in the development of efficient cancer diagnosis and in the design of effective drugs. In this chapter, we evaluate and compare existing feature selection methods in the context of gene expression analysis.

4.1 Data Sets and Preprocessing

We use the same gene expression data sets as in previous chapter (see Table 3.1). We split the training and test data in the same way as in previous studies for fair comparison. According to the source of Leukemia data set, their training set consists 38 samples (27 ALL and 11 AML), and their test set consists 34 samples (20 ALL and 14 AML). Since there is no defined training and test set from the

Dataset	Number of Training Samples	Number of Test Samples	Number of Genes
Colon	32(11+/21-)	30(11+/19-)	2000
Leukemia	38(11+/27-)	34(14+/20-)	7129
Lung	14(6+/8-)	12(5+/7-)	19624

Table 4.1: Gene expression data used for feature selection

source of Colon data, we randomly split the data into 32 samples for training and 30 samples for testing. Similarly, we randomly split the Lung cancer data into 14 samples for training and 12 samples for testing. Table 4.1 provides a summary of the characteristics of the data sets.

There are many different ways to preprocess the data. For example, Golub et al. [19] first normalizes the feature vectors of training samples, and then normalizes the test samples by subtracting the mean and dividing the standard deviation of the training samples. It makes sense to normalize the training and test samples separately as in [19], because usually the test data is not available when we build classifiers from the training data. However, the sample size of our data is very small, so it would be better if we merge the training and test samples, and normalize them together. One drawback about this type of normalization is that we need to rebuild our classifier before predicting a new test sample, but we gain some improvement in the quality of the normalized data, which we think more important. Therefore, we merge the training and test samples, and normalize the data so that expression values for each gene i have a zero mean and unit variance.

4.2 Recursive Feature Elimination

Recursive Feature Elimination is a feature selection algorithm proposed by Guyon et al. in [20]. RFE performs feature selection by iteratively training a classifier with current set of features and remove the feature with the smallest ranking criterion

Algorithm SVM-RFE(X_0, $\mathbf{y}$) {
/* input: training samples $X_0 = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k, ..., \mathbf{x}_l]^T$ where $\mathbf{x}_k \in R^n$ */
/* input: class labels $\mathbf{y} = [y_1, y_2, ..., y_k, ..., y_l]^T$ */
/* output: feature ranked list $\mathbf{r}$ */
(1) Initialize:
Subset of surviving features $\mathbf{s} = [1, 2, ..., n]$
Feature ranked list $\mathbf{r} = []$
(2) Repeat: until s=[]
Restrict training examples to good feature indices
$X = X_0(:, \mathbf{s})$
Train the classifier
α =SVM-train$(X, \mathbf{y})$
Compute the weight vector of dimension length(s)
$\mathbf{w} = \sum_k \alpha_k y_k \mathbf{x}_k$
Compute the ranking criteria
$c_i = (w_i)^2$, for all i
Find the feature with smallest ranking criterion
$f = \arg\min(\mathbf{c})$
Update feature ranked list
$\mathbf{r} = [\mathbf{s}(f), \mathbf{r}]$
Eliminate the feature with smallest ranking criterion
$\mathbf{s} = \mathbf{s}$(1:f-1, f+1:length(s))
} /* end algorithm */

Figure 4.1: SVM-RFE Algorithm [20]

at each iteration. According to the order of elimination, a ranked feature list r is generated at the end: the earlier a feature is eliminated, the lower its rank is. When SVM, usually with linear kernel, is used as the training classifier, the algorithm is referred to as SVM-RFE. Figure 4.1 presents an outline of SVM-RFE algorithm in the linear case.

SVM-RFE can also be generalized to remove more than one feature per iteration for speed reasons. In this chapter, unless otherwise stated, we use SVM-RFE by eliminating one gene at a time. Given a ranked list r, we can obtain subsets of genes by selecting top K genes in the list. By varying the value of K, we can obtain multiple subsets. In the case of Colon data, we get 2000 subsets with size ranging from 1 to 2000. As mentioned in Section 1.3, ideally we want to find the

subset of genes with which the test accuracy is the highest. We define *optimal subset* to be the subset of features with which a classifier achieves the highest test accuracy, and the smallest optimal subset is called the *biomarker set.*

According to [20] and [16], the features selected matter more than the classifier used. Therefore, we evaluate the quality of subsets of genes using SVM only. Figure 4.2 shows the accuracies of SVM on test data using various subsets of genes. Table 4.2 presents a summary of optimal subsets selected using test accuracy. For the Colon data set, SVM achieves best performance of 5 test errors (i.e. 83.3% accuracy) with 53 different optimal subsets, and the smallest optimal subset contains 416 genes. For the Leukemia data set, there is only one single optimal subset with size of 5. For the Lung cancer data set, any subset with first 176 genes included are optimal subsets.

If the ranks returned by SVM-RFE reveal the true ranks of genes in terms of relevance, and furthermore if we adopt the assumption described in [7] that including a relevant gene lowers the test error and including an irrelevant gene increases the test error, then the ideal curve of accuracies of SVM with respect to various subsets should have continuous global optima in an interval starting at the biomarker set. The curve with multiple spaced optima (e.g. zigzag curve) is not desirable, because it implies that relevant features and irrelevant genes are not separated but somehow mixed together.

As discussed earlier, in real world problem, the test accuracy is not available beforehand. Therefore, we have to use the LOO accuracy on the training data as the criterion to select optimal subsets and/or the biomarker set. Table 4.3 gives a summary of optimal subsets selected by LOO accuracy on the training set. The LOO accuracies on the training set are all 100%. If the optimal subsets are reliable, we expect the test accuracies with these optimal subsets to be very high. In Figure 4.2, we show both the LOO accuracies on the training set and the test accuracies

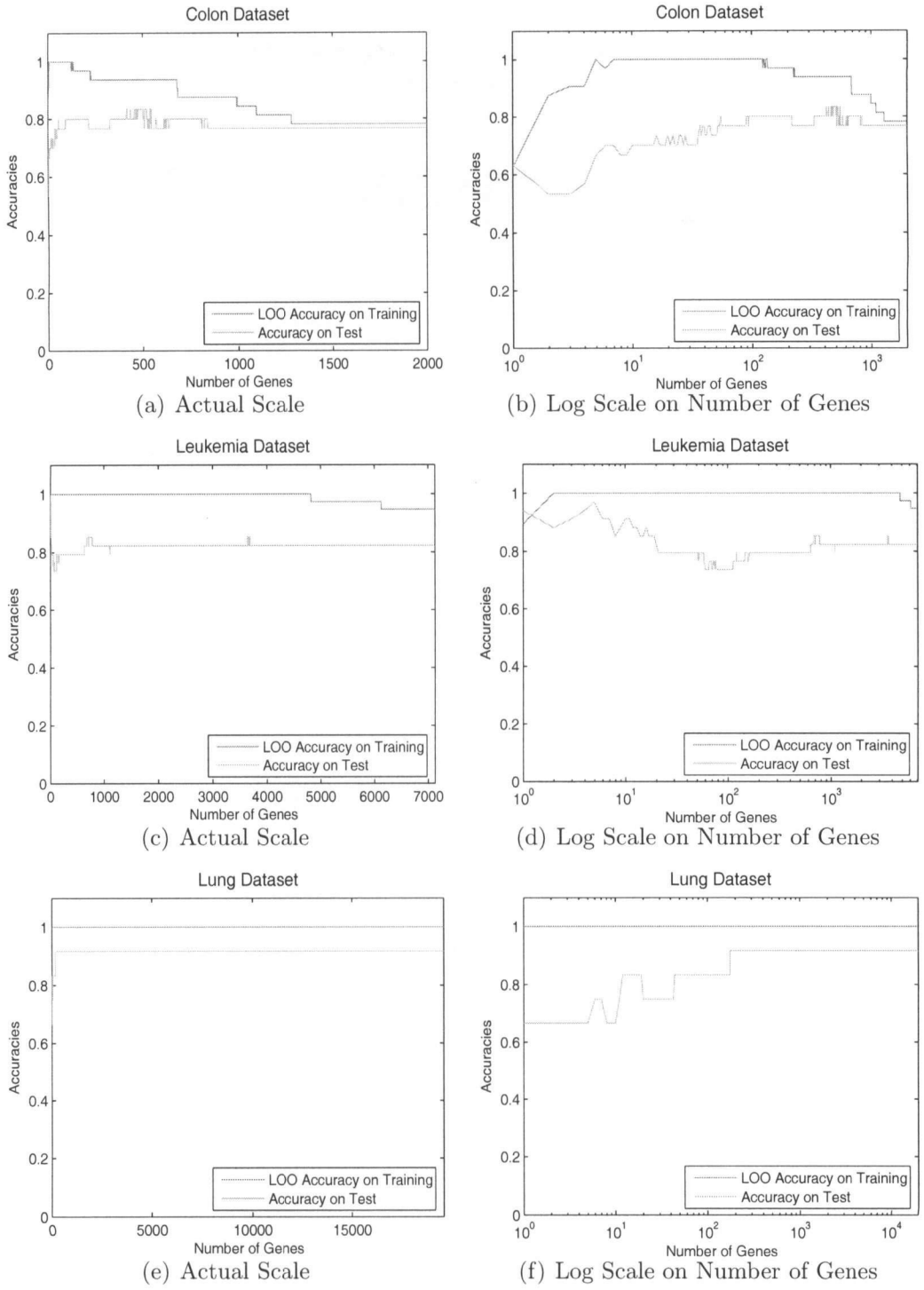
Dataset	Optimal Subsets			All Genes	
	# opt. sets	test accu.	# genes in biomarker set	test accu.	# genes
Colon	53	83.3	416	76.7	2000
Leukemia	1	97.1	5	82.3	7129
Lung	19449	91.7	176	91.7	19624

Table 4.2: Evaluation of the feature ranks given by SVM-RFE—optimal subsets selected using test accuracies. An *optimal set* is a subset of genes with which the classifier SVM achieves the highest test accuracy. The smallest optimal set is called the *biomarker set*. Test accuracy for the optimal subsets is the classification accuracy of SVM on test data using genes in the optimal subsets only.

Dataset	LOO Accu. on Training	# Genes	# Optimal Subsets
Colon	100	5	126
Leukemia	100	2	4821
Lung	100	1	19624

Table 4.3: Evaluation of the feature ranks given by SVM-RFE—optimal subsets selected using LOO accuracies on the training data. An *optimal set* is a subset of genes with which the classifier SVM achieves the highest LOO accuracy over the training samples. The smallest optimal set is called the *biomarker set*. LOO accuracy for the optimal subsets is the LOO accuracy on the training set using genes in the optimal subsets only.

on the test set with various subsets of genes. The curve of LOO accuracy on the training data and the curve of test accuracy do not achieve the optima at the same gene subsets. This means the biomarker set selected using the training set is not identical to the true biomarker set selected using the test accuracy. Additionally, as shown in Figure 4.2, there is a big gap between the two curves, which implies the problem of overfitting. Additionally, as discussed above, a large number of discontinuous optimal subsets are not desirable as that implies the unreliability of ranks of the features.



(a) Actual Scale (b) Log Scale on Number of Genes

(c) Actual Scale (d) Log Scale on Number of Genes

(e) Actual Scale (f) Log Scale on Number of Genes

Figure 4.2: Results of forward selection: accuracies of SVM with various number of genes ranked by SVM-RFE. The X-axis represents the number of genes, and the Y-axis represents accuracies (test accuracies and LOO accuracies on training data). The X-axes in (a), (c), and (e) are in actual scales, while the X-axes in (b), (d), and (f) are in logarithmic scales.

4.3 GPC with ARD

Automatic Relevance Determination (ARD)([27], [35]) is one of the most successful Bayesian methods for feature selection and sparse learning. We focus on exploiting ARD to perform feature selection on high dimensional gene expression data.

When used in feature selection, ARD is a hierarchical approach where the relevance of each input feature is represented by a hyperparameter. In the context of Gaussian process classifier, ARD hyperparameters can be directly embedded into the covariance function as in Equation 3.8. Let w_d be the weight of the d-th feature and α^{-1} be the variance of w_d, then we have $p(w_d|\alpha_d) = \mathcal{N}(0, \alpha^{-1})$. Obviously, if $\alpha_d \to \infty$, then variance $\alpha^{-1} \to 0$, and weight $w_d \to 0$, which implies the feature is irrelevant. If $\alpha_d \ll \infty$, then the variance is finite, and the weight can vary, which indicates the d-th feature is relevant. ARD optimizes $\hat{\alpha} = \arg\max_{\alpha} p(y|X, \alpha)$. During optimization some α_d approaches to ∞, and the corresponding irrelevant features will be eliminated.

Figures 4.3—4.5 show the distribution of ARD values (i.e. weights) returned by GPC-Laplace and GPC-EP for three data sets. From the figures, we can see that many ARD values returned by GPC are very close to zero but not zero. Therefore, to select a subset of genes, one needs to specify a threshold for ARD values. For all of the three datasets, ARD values returned by GPC with Laplace are more spread out than that returned by GPC with EP. Sparser ARD values make it easier to specify a threshold and give more information about the ranks of features in terms of relevance. In the next chapter, we will show how to effectively select features utilizing the ARD values returned by GP-Laplace.

For now, we know ARD values indicate the level of relevance (or rank) of a feature: feature with higher ARD value has higher rank in terms of relevance. Therefore, similar to SVM-RFE, GPC with ARD returns a ranked list of genes.

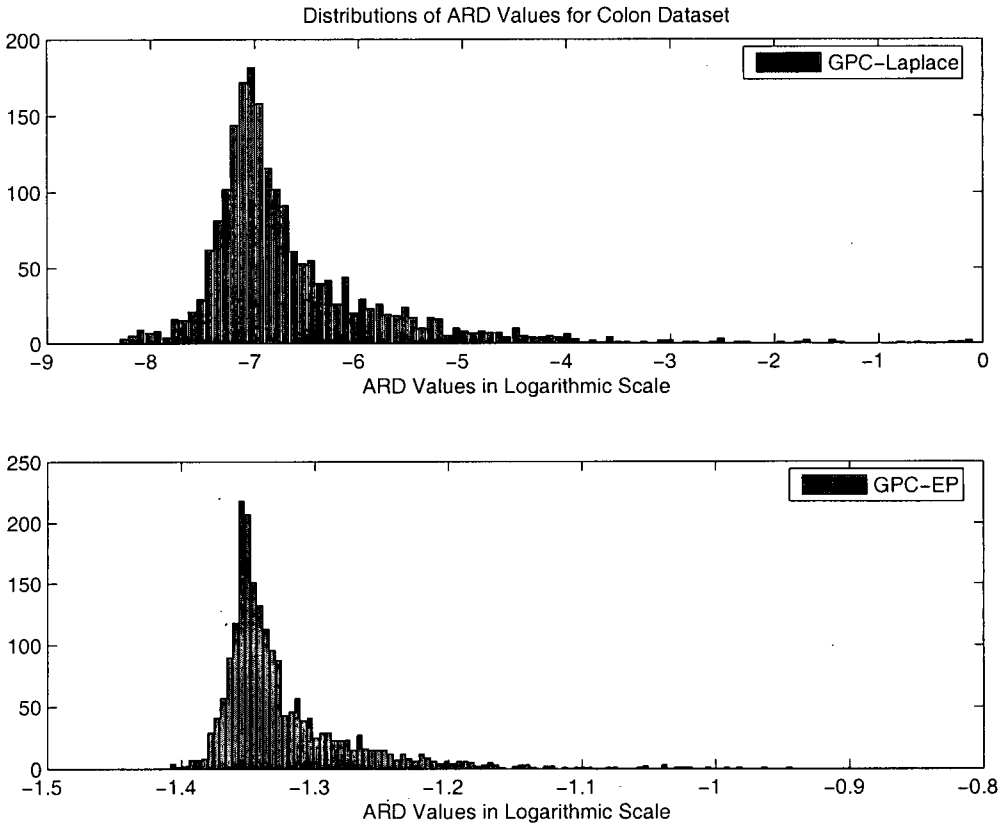


Figure 4.3: Distributions of ARD values for the Colon data set

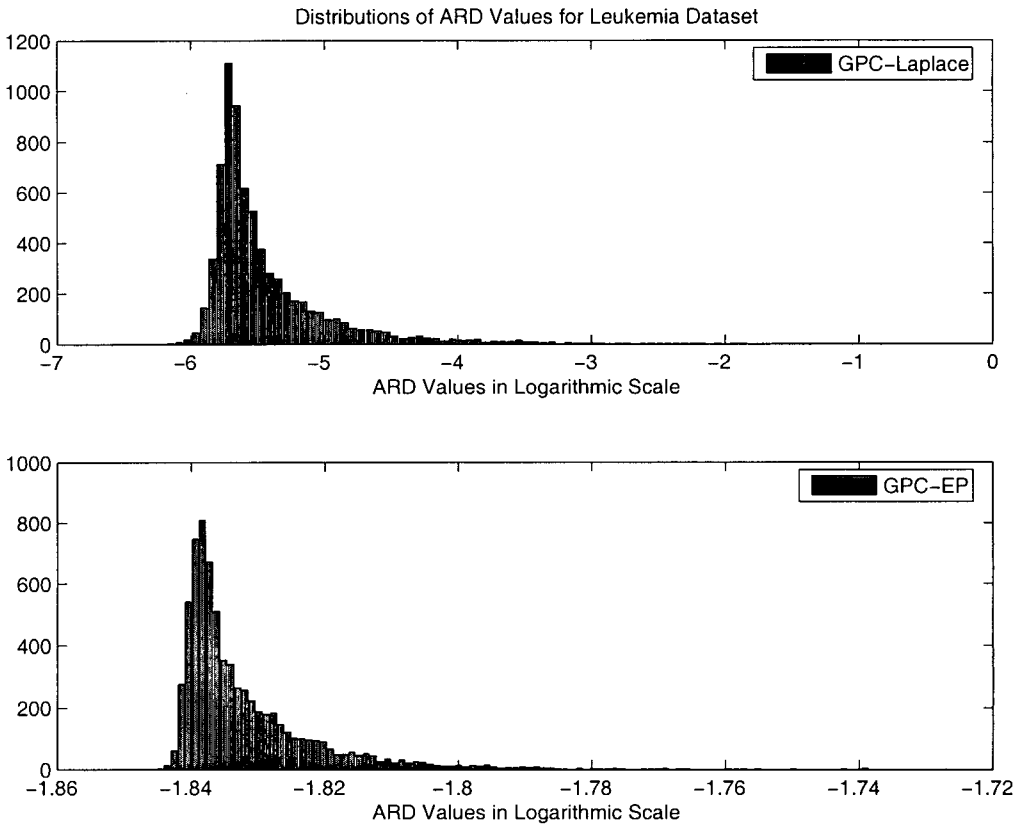


Figure 4.4: Distributions of ARD values for the Leukemia data set

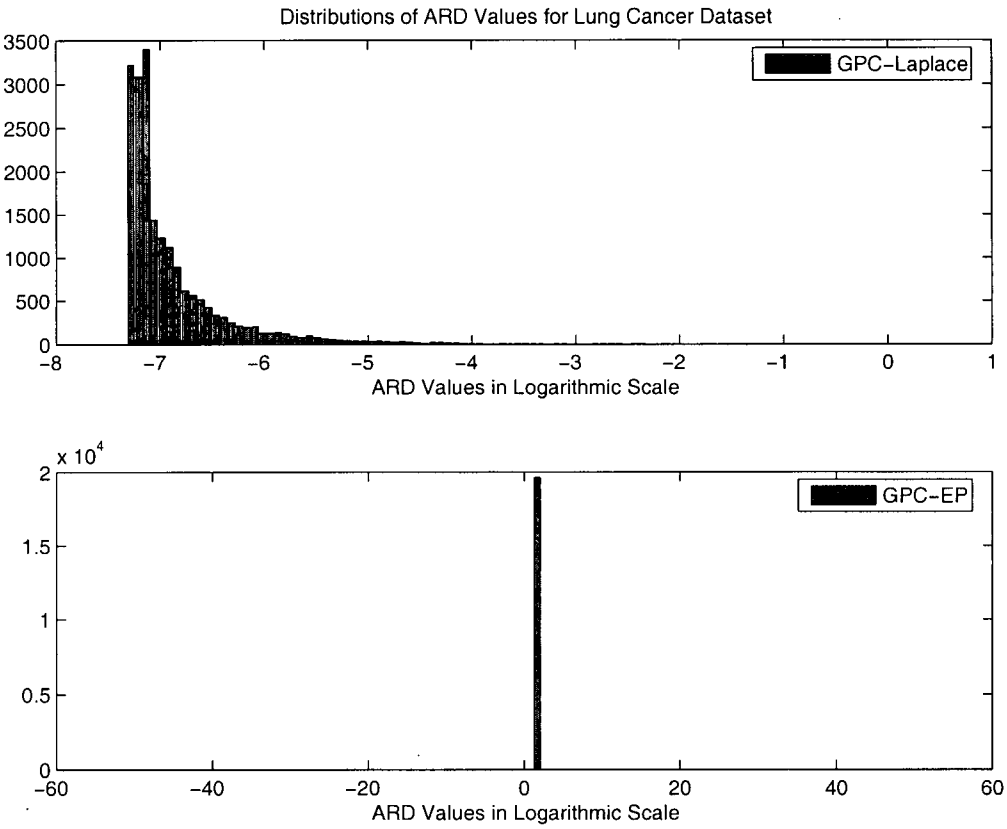


Figure 4.5: Distributions of ARD values for the Lung cancer data set

Dataset	Algorithm	Optimal Subsets		
		# opt. sets	test accu.	# genes in biomarker set
Colon	GP-Laplace	191	80	8
	GP-EP	1	83.3	4
Leukemia	GP-Laplace	1	94.1	27
	GP-EP	11	94.1	1
Lung	GP-Laplace	1	100	3

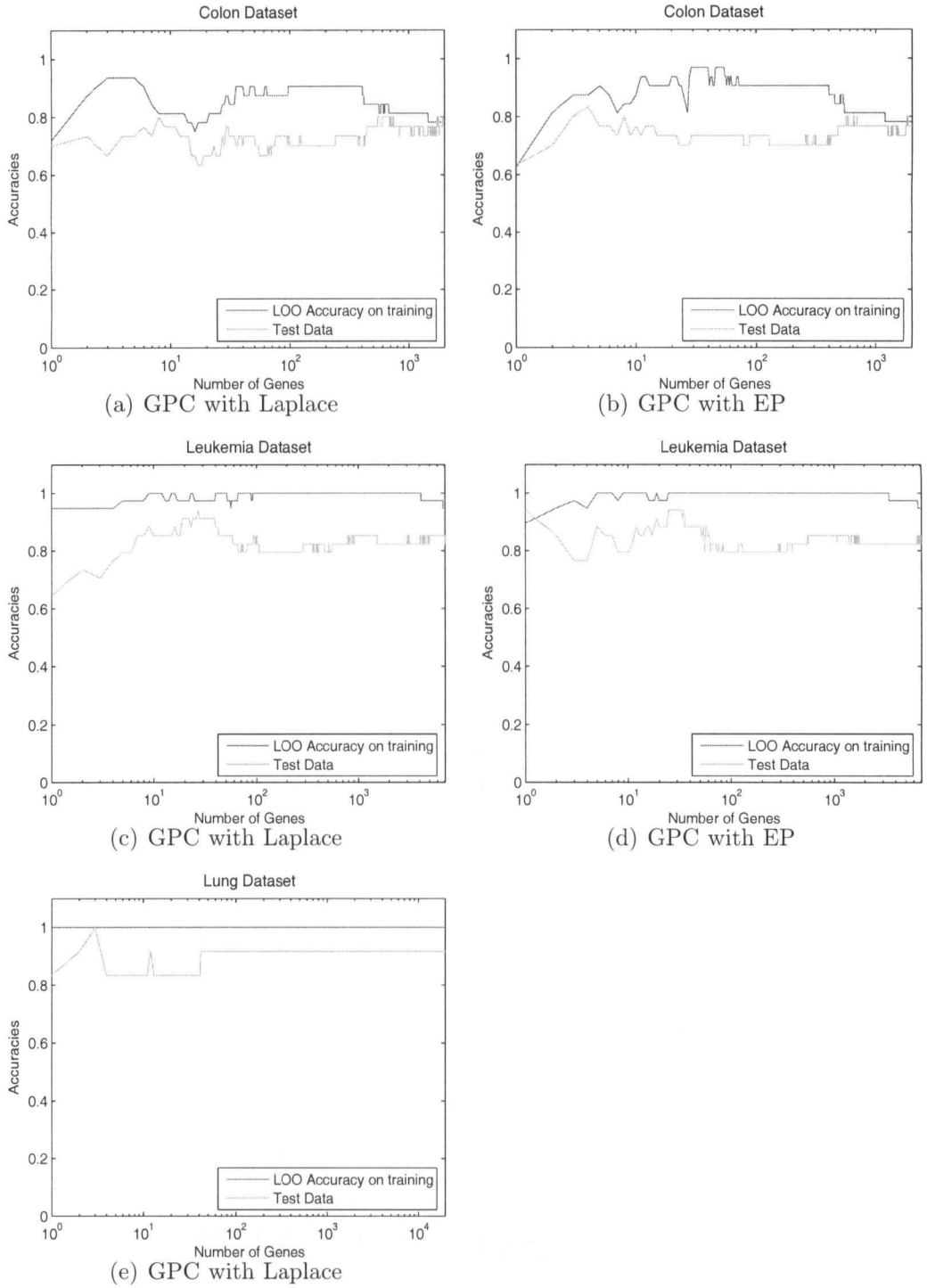
Table 4.4: Evaluation of the feature ranks given by GPC-ARD—optimal subsets selected using test accuracies. Compare to Table 4.2.

Dataset	Algorithm	LOO Accu. on Training	# Genes	# Opt. Sets
Colon	GP-Laplace	93.8	3	3
	GP-EP	96.9	29	21
Leukemia	GP-Laplace	100	9	4102
	GP-EP	100	5	3430
Lung	GP-Laplace	100	1	19624

Table 4.5: Evaluation of the feature ranks given by GPC-ARD—optimal subsets selected using LOO accuracies on the training data. Compare to Table 4.3.

Forward selection is needed to select the biomarkers. We evaluate the reliability of ARD values as a ranking criteria in a similar way as we evaluate SVM-RFE. The performance of SVM with various subsets of genes are reported in Tables 4.4—4.5 and Figure 4.6, where the columns and figures have similar meanings as in Table 4.2-4.3 and Figure 4.2 .

Similar to SVM-RFE, the curve with multiple spaced optima (e.g. zigzag curve) in Figure 4.6 is not desirable, as it indicates the unreliability of the ranking of features. Qi et al. [37] have shown that evidence optimization can lead to over-fitting by picking one from many classifiers that correctly classify the data. This is consistent with our findings in Figure 4.6, where the LOO accuracies on the training data are significantly better than the test accuracies. In next chapter, we will see how a resampling technique, such as cross validation can be used to address the problem of overfitting.



(a) GPC with Laplace

(b) GPC with EP

(c) GPC with Laplace

(d) GPC with EP

(e) GPC with Laplace

Figure 4.6: Results of forward selection: accuracies of SVM with various number of genes ranked by ARD values. The X-axis represents the number of genes, and the Y-axis represents accuracies (test accuracies and LOO accuracies on training data). GPC-Laplace is used to get the ARD values for (a), (c), and (e). ARD values from GPC-EP are used to get (b) and (d).

From Figure 4.2 and 4.6, we can see that if appropriate subset of genes are selected, we can improve the classification capability of the gene expression data. To evaluate SVM-RFE and GPC-ARD in identifying biomarkers, we use LOO accuracy on training data to select a biomarker set and compute the test accuracy using genes in biomarker sets only. The results are included in Table 4.6.

4.4 RVM and SMLR on Feature Space

Sparse classification algorithms, such as RVM and SMLR, obtain sparse solutions for regression and classification while maintaining their Bayesian interpretation. Sparsity-promoting priors are used to encourage the weight estimates to be either significantly large or exactly zero. As mentioned in Chapter 3, if we use original features or their transformed forms as the basis functions, then the learned classifier can be considered as a feature weighting algorithm which returns sparse weights. Therefore, in the context of gene expression analysis, unlike SVM-RFE which simply ranks genes, RVM and SMLR automatically select one optimal subset of genes, which is the biomarker set. In this section, we assess the quality of biomarkers selected by these two sparse classifiers.

Table 4.6 presents the test accuracies using biomarker sets selected by the five feature selection algorithms considered in this chapter. As mentioned earlier this Chapter, the training and test data for the Leukemia data are specified by the source of the data, but there are no predefined training and test data for the other two data sets. We randomly partitioned the data set into the training set and the test set. For the Leukemia data, test accuracies for all feature selection methods are higher than the accuracy with all genes. However, all feature selection methods fail on the other data sets. We suspect that the failure might be caused by

Test Accuracies of SVM with biomarker sets selected by various feature se-

lection methods. The numbers (e.g. 76.7%) below the data sets (e.g. Colon) are the test accuracies using all genes without any feature selection or feature weighting. The training and test data for the Leukemia data are predefined. For this data set, test accuracy with any biomarker set is higher than the accuracy with all genes. The failure of feature selection methods on the other data sets might be due to the possible unfortunate partition of training and test data.

The results from one partition may not reveal the true performances of feature selection methods. Therefore, we do the following: we first split the data into 10 folds to get f1, f2, ... , f10, and then leave the fold f_i $(i = 1, 2, ..., 10)$ out, select biomarker set b_i using the remaining 9 folds. We then train the 9 folds with b_i only using several classifiers and compute the number of errors when classifying the test data f_i. The total number of errors in 10 folds are reported in Table 4.7. Now for the Colon data, the classification errors are reduced by using feature selection methods. We exclude the Lung cancer data set from now on, because several methods can not handle it.

According to the results in Table 4.6 and Table 4.7, there is no feature selection method that consistently outperforms other methods in terms of classification accuracies. In the next chapter, we shall see how to further improve SVM-RFE and GPC with ARD by utilizing cross validation techniques. We shall also evaluate the sensitivity of the feature selection methods in selecting features with respect to changes in the training data.

Dataset	Algorithm	Test Accu. (%)	# Genes in Biomarker Set
Colon (76.7%)	SVM-RFE	66.7	5
	GPC-ARD-LAP	66.7	3
	GPC-ARD-EP	73.3	29
	RVM	63.3	3
	SMLR	63.3	16
Leukemia (82.3%)	SVM-RFE	88.2	2
	GPC-ARD-LAP	88.2	9
	GPC-ARD-EP	88.2	5
	RVM	91.2	2
	SMLR	94.1	20
Lung (91.7%)	SVM-RFE	66.7	1
	GPC-ARD-LAP	83.3	1
	SMLR	83.3	9

Table 4.6: Test Accuracies of SVM with biomarker sets selected by various feature selection methods. The numbers (e.g. 76.7%) below the data sets (e.g. Colon) are the test accuracies using all genes without any feature selection or feature weighting. The training and test data for the Leukemia data are predefined. For this data set, test accuracy with any biomarker set is higher than the accuracy without feature selection. The failure of feature selection methods on the other data sets might be due to the unfortunate partition of the training and test data.

Dataset	Algorithm	Total # of Errors	Avg. Size of of Biomarker Sets
Colon ($N = 62$)	All genes	17	2000
	SVM-RFE	14	9
	GPC-ARD-LAP	15	21
	RVM	10	6
	SMLR	17	30
Leukemia ($N = 72$)	All genes	2	7129
	SVM-RFE	9	4
	GPC-ARD-LAP	8	5
	RVM	7	5
	SMLR	2	27

Table 4.7: Total number of errors in 10 experiments and the average size of biomarker sets. The number (e.g. $N = 62$) below the data sets (e.g. Colon) is the total number of validation cases in the 10-fold cross validation. No feature selection method outperforms all other methods in terms of classification accuracies on both data sets.

Chapter 5

Cross Validation Optimization

An intuitive solution to solve the problem of low sample size is resampling. Commonly used resampling methods that generate new data from the distribution of existing data, such as interpolation and extrapolation, do not help in the case of gene expression analysis for the following reasons. First, it is not reasonable to model the distribution from a small number of samples. Second, the conclusion drawn from synthetic data may not be applicable to real-world problems. Therefore, the only option left is to use cross validation. In this way, we can make the best use of the available real data.

In the literature, the qualities of feature selection methods are only evaluated by classification accuracies. We suggest to use another criterion, called *stability*, to evaluate how sensitive a feature selection method is to the changes of training data. In this chapter, we first evaluate the stability of RVM and SMLR in selecting features, and show why they are not suitable to be optimized using cross validation. We then present two existing feature selection methods that utilize cross validation, and then propose our own method. Experimental results show effectiveness and efficiency of our algorithm.

Experiment	Features (indices)
1	[356 377 765 1969 1924 1976]
2	[356 377 765 1769 1859 1976]
3	[356 377 765 1769 1859 1976]
4	[353 377 765 1013 1024 1759]
5	[356 377 765 1769 1859 1976]
6	[353 377 493 765 1050 1757 1769 1823]
7	[350 377 493 765 792 1769 1823 1976]
8	[377 700 765 1482 1769 1859]
9	[356 377 765 1769 1859 1976]
10	[377 717 1353 1419 1555]

Table 5.1: Features selected by RVM for the Colon data set

5.1 Stability of Feature Selection Methods

Ideally, if only a small portion of the training data set is changed, we expect that the set of features selected by a feature selection algorithm does not change much. *Stability* measures how stable a feature selection algorithm is in selecting features.

We evaluate the stability of RVM and SMLR as follows. We first split the data into 10 folds, and each time one fold is left out and the remaining 9 folds are used for training. We end up with 10 different optimal subsets of features returned by the 10 different training sets. Table 5.1 lists the 10 optimal subsets of genes selected by RVM with the Colon data set. It is hard to interpret the stability of a feature selection algorithm using a table like Table 5.1. Therefore, we propose to use a single scalar value—stability score S—to represent the stability of a feature selection algorithm with a particular data set.

We first define the following terms and notations before giving the definition of stability score S.

- k—the number of folds we divide the data set into (e.g. $k = 10$).
- $D^{\backslash i}$—the training set containing all folds but the i-th one.
- F_i—the feature set selected using $D^{\backslash i}$.

- $|A|$—the cardinality of set A.
- N_i—the number of features that appear in i subsets. For example, N_1 represents the number of features that only appears in one subset, and N_k represents the number of features that appear in all subsets.

We now define the stability score S as follows:

$$S = \frac{\sum_{i=2}^{k}(\frac{i}{k} \times N_i)}{|\bigcup_{i=1}^{k} F_i|} \tag{5.1}$$

Since RVM only selects a small number of genes, we ignore the relative ranking of selected genes and only care about which subset of genes are selected. The denominator is the cardinality of the union set of all feature sets, and the numerator is the weighted sum of the number of common genes. The score S is defined such that it has a range of $[0, 1]$. If all feature sets are identical, then $S = 1$ because: (1) $F_i = F_j$, $\forall\ i, j = 1, 2, ..., k$; (2) $\bigcup_{i=1}^{k} F_i = F_j$, for any $j = 1, 2, ..., k$. Therefore, $|\bigcup_{i=1}^{k} F_i| = |F_1|$; (3) $N_k = |F_1|$, and $N_j = 0$ for all $j \neq k$; and finally $S = \frac{\sum_{i=2}^{k}(\frac{i}{k} \times N_i)}{|\bigcup_{i=1}^{k} F_i|} = \frac{N_k}{|F_1|} = 1$. In the contrast, if all feature sets are disjoint, $S = 0$ as every selected feature is selected once, i.e. $N_1 = |\bigcup_{i=1}^{k} F_i|$, and $N_i = 0$ for $i = 2, 3, ..., k$. This is the reason why we omit $i = 1$ in the numerator of Equation 5.1.

Table 5.2 shows the number of occurrences of all features selected by RVM on Colon data set. We can compute the stability score of RVM on this data set as:

$$S = \frac{\frac{2}{10} \times 3 + \frac{5}{10} \times 2 + \frac{6}{10} \times 1 + \frac{8}{10} \times 1 + \frac{9}{10} \times 1 + \frac{10}{10} \times 1}{23} = 0.213$$

Table 5.3 reports the stability scores of RVM and SMLR on Colon and Leukemia data sets. RVM usually selects about 6 features, while SMLR selects 20-30 features in each training. According to our stability evaluation criterion, SMLR is more stable than RVM, as the former has higher stability scores with both data sets. If

# Occurrences	Features (indices)	# Features (N_i)
1	{350, 700, 717, 792, 1013, 1024, 1050, 1353, 1419, 1482, 1555, 1757, 1759, 1924}	$N_1 = 14$
2	353, 493, 1823	$N_2 = 3$
3	–	–
4	–	–
5	356,1859	$N_5 = 2$
6	1976	$N_6 = 1$
7	–	–
8	1769	$N_8 = 1$
9	765	$N_9 = 1$
10	377	$N_{10} = 1$

Table 5.2: Number of occurrences of features selected by RVM for the Colon data set

	RVM	SMLR
Colon	0.213	0.243
Leukemia	0.150	0.321

Table 5.3: Stability scores of RVM and SMLR on the Colon and Leukemia data sets

every feature is shared by half of the feature sets, then we expect the stability score to be 0.3. We propose that the stability of a feature selection algorithm is *acceptable* if its stability score is at least 0.3. In our case, each training set is different by one fold (or partition) of data samples rather than one single data sample. Therefore, we lower our acceptable score to be 0.2. According to our criterion, the stability of RVM is not acceptable as its stability score for the Leukemia data is only 0.15. In summary, both RVM and SMLR select a small number of features, and SMLR is more stable than RVM. The idea of using cross-validation to improve the stability of feature selection algorithms is to average the weights of features selected using different subsets of training samples.

Initialize	Ranked feature set $R = []$;
	Selected subset $S = [1, ..., d]$
Repeat	until all features are ranked
	a) Train t linear SVMs on subsets of the original training data, with features in set S as input variables
	b) Compute and normalize the weight vectors using Equation 5.2
	c) Compute the ranking scores c_i for features in S using Equation 5.3
	d) Find the feature with the smallest ranking score: $e = \arg\min_i c_i$
	e) Update: $R = [e, R], S = S - [e]$
Output	Ranked feature list R

Figure 5.1: Algorithm of MSVM-RFE [14]

5.2 Multiple SVM-RFE

In [14], Duan et al. proposes a new feature selection algorithm called MSVM-RFE (Multiple SVM-RFE) based on SVM-RFE (Figure 4.1) by adding cross validation into each feature elimination step. Unlike SVM-RFE, which trains SVM once using all training samples, MSVM-RFE trains multiple SVMs using different subsets of training samples at each step, and computes the ranking score from a statistical analysis of weighted vectors of multiple SVMs. The outline of MSVM-RFE is presented in Figure 5.1.

Let $\mathbf{w}_j$ be the weight vector returned by the jth SVM and w_{ji} be the weight value associated with the ith feature. The weight vectors are normalized as:

$$w_j = \frac{w_j}{\|w_j\|} \tag{5.2}$$

Let $v_{ji} = w_{ji}^2$, and the ranking score is computed as:

$$c_i = \frac{\bar{v}_i}{\sigma_{v_i}} \tag{5.3}$$

where $\bar{v}_i = \frac{1}{t}\sum_{j=1}^{t} v_{ji}$ and $\sigma_{v_i} = \sqrt{\frac{\sum_{j=1}^{t}(v_{ji}-\bar{v}_i)^2}{t-1}}$

5.3 Forward Feature Selection with Gaussian Processes

In [8], Chu et al. proposed a feature selection algorithm based on Gaussian processes to identify biomarkers for tasks with ordinal labels. We refer their algorithm as GP-ARD-FS. Figure 5.3 presents the outline of their algorithm.

In the phase of preprocessing, the goal is to discard a significant amount of genes that have low discriminative capabilities. They adopt the idea in paper [13] and use Wilcoxon rank sum statistic as the ranking criterion. Let X_i be the vector that consists of expression values of all samples for gene i. Split X_i into X_i^+ and X_i^-, where X_i^+ and X_i^- are vectors of expression values of samples from positive class and negative class, respectively. For each gene i, Wilcoxon rank sum test is performed to test the null hypothesis that X_i^+ and X_i^- come from distributions with equal medians. The major steps of informative gene selection using Wilcoxon rank sum test are as follows [13]:

- Set significance level $\alpha = 0.01$.
- Compute Wilcoxon-statistics for every gene i using X_i^+ and X_i^-.
- Use the statistics to compute the corresponding p-values.
- Select the genes whose p-values are smaller than the significance level α.

Smaller p-values cast doubt on the hypothesis that the distributions of samples from two different classes are identical. In other words, genes with smaller p-values have high discriminative capabilities. The original Colon data set consists of 2000 genes. There are only 373 genes significantly differentially expressed at the significance level of $\alpha = 0.01$. Therefore, after preprocessing the dimensionality of the data set is reduced to 373.

We illustrate the remaining steps of the algorithm using a concrete example. Suppose the data set after preprocessing is $D = \{(x_i, y_i)|i = 1, 2, ..., 9\}$ and $x_i \in \mathbb{R}^6$

and $Y_i \in \{\pm 1\}$. Notice that the aim of this concrete example is to elaborate the algorithm. In real gene expression data, the dimensionality is much higher than the sample size. Figure 5.2 illustrates the algorithm step by step using a running example.

5.4 GP-ARD-RFE

In comparison with other feature selection techniques, GP-ARD-FS is effective in selecting relevant genes. However, it has several drawbacks which we will discuss later. We propose a new algorithm called GP-ARD-RFE aiming to overcome the drawbacks of previous algorithms. Figure 5.7 presents an outline of our algorithm. Figure 5.4 - 5.6 explain our algorithm with two concrete examples.

In the phase of preprocessing, GP-ARD-FS uses the Wilcoxon rank sum test to eliminate genes that are not significantly differentially expressed at a significance level. A statistical test—either t-test or non-parametric Wilcoxon rank sum test—is used to determine the statistical significance of an observation. In the context of gene expression analysis, statistical test assesses the discriminative capability of a gene by testing if the distribution of normal samples is identical to that of tumor issues for this particular gene. One major shortcoming of using the rank sum test is that data of low sample size does not reveal trustworthy information about the distribution.

Alternatively, we propose to use the average ARD values to eliminate a portion of irrelevant genes at the early stage. This procedure is presented under *"RFE"* in Figure 5.7. During the phase of feature elimination, we adopt the settings of [20] and recursively eliminate chunks of features at a time. At the first iteration, we keep the number of genes, which is the closest power of 2. For example, the original Colon cancer data set contains 2000 genes. At the first iteration, we will

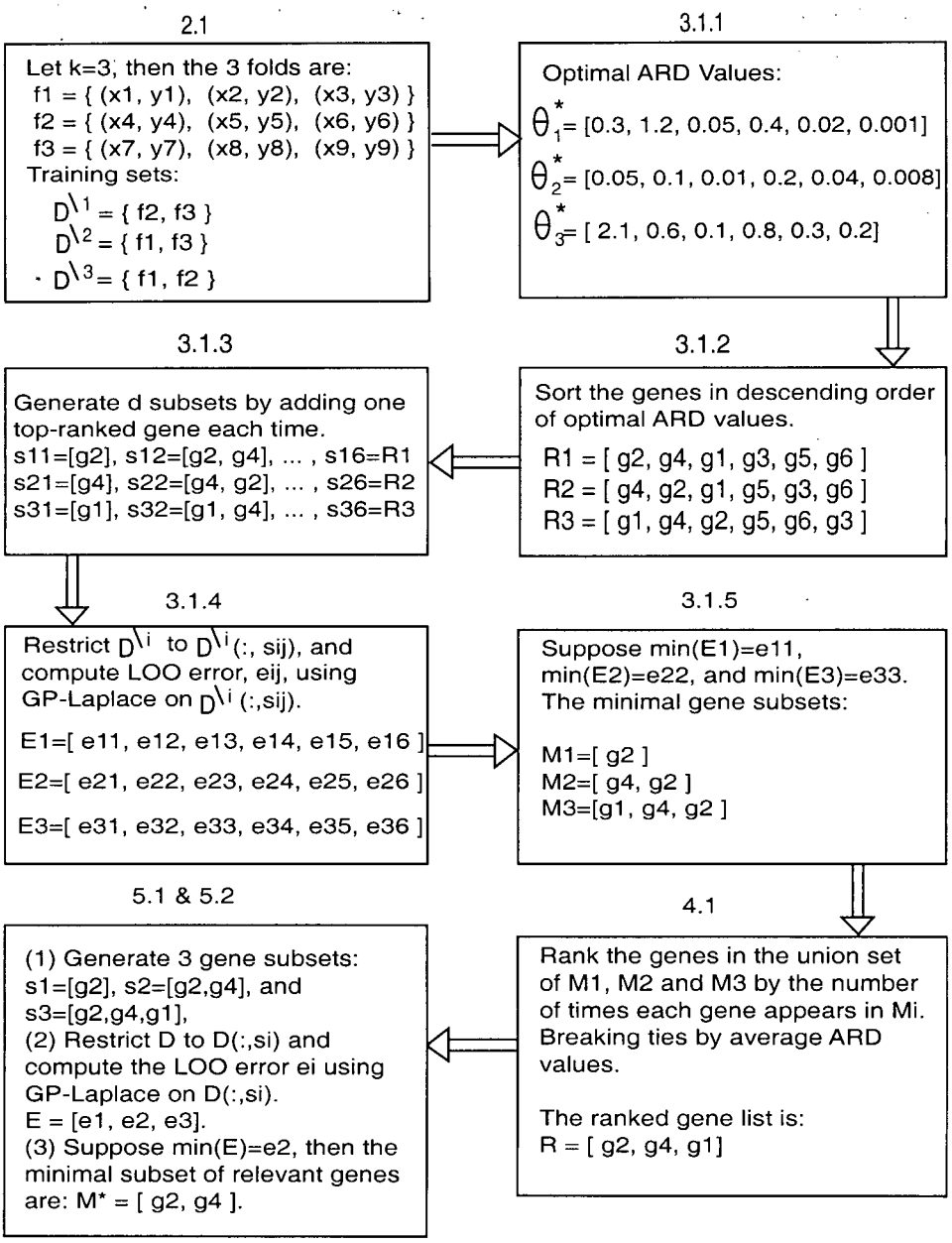


Figure 5.2: Illustration of GP-ARD-FS algorithm: a running example

keep $1024(= 2^{10})$ genes. At subsequent iterations, we eliminate half of current subset of genes until the number of remaining genes is 2^p, where p is specified by the user. Empirically, $p = 8$ is a good choice. Thus, the number of genes of the Colon data set after each iteration are: 1024, 512 and 256.

We randomly split the data into k folds as usual. One fold is left out in turn and the remaining $k-1$ folds are passed to the classifier of GP-ARD with Laplace for training. The classifier carries evidence maximization and returns the optimal θ^*, a vector of optimal values of ARD parameters. Let θ_i^* be the optimal ARD values returned by GP-ARD trained with all folds except the ith fold. Based on the optimal ARD values θ_i^*, the genes are then sorted in descending order of relevance, and a rank list R_i is generated.

The ARD values returned by GP-Laplace are not calibrated. Before we compute the average of ARD values, it is important for us to normalize the ARD values so that the norm of θ_i^* is 1 for all i. In this way, each θ_i^* contributes to the average ARD values fairly. For example, in Figure 5.4, θ_3^* has larger values than θ_1^* in general. Without normalization, θ_3^* would have greater impact on the average ARD values than θ_1^*. Specifically, we normalize θ_i^* as follows:

$$\theta_i^* = \frac{\theta_i^*}{\|\theta_i^*\|} \tag{5.4}$$

Average ARD value is not the only criterion we use to decide which genes to eliminate. Suppose the number of genes to keep at this iteration is m. We identify the following two groups of genes:

- L—the set of features whose rank is lower than m in all R_i

  $L = \{g | g \in C, g \notin R_i(1:m), \forall i\}$

- T—the set of features that are within top m in all R_i.

  $T = \{g | g \in C, g \in R_i(1:m), \forall i\}$

Genes in T survive this iteration, and genes in L are eliminated regardless of their ARD values. The idea is that if everyone thinks that gene g should stay (or leave), then we should not eliminate (or keep) it, respectively. For example, in Figure 5.4, $g5$ is ranked within top 4 in all R_i's (see 2.1.1.3), but its rank in S according to average ARD values is the fifth (see 2.1.2). We keep $g5$ as it is a common top gene.

Suppose the number of genes in T is $|T|$, and the number of genes that should survive at this iteration is m, then we have $m - |T|$ spaces left for surviving genes. We choose these genes according to their average ARD values. We iteratively eliminated chunks of genes until 2^p (e.g. 256) genes left.

We illustrate feature elimination using a concrete example as shown in Figure 5.4 - 5.5. Suppose the data set consists of 9 samples: $D = \{(x_i, y_i)|i = 1, 2, ..., 9\}$. Each data sample contains 6 genes at the beginning. That is, $x_i \in \mathbb{R}^6$. We set $p = 1$, and thus the number of features left after RFE is $2^1 = 2$.

After the number of features are reduced to 2^p, we perform k-fold validation again on data with these 2^p genes: we get optimal ARD values, normalize them, sort genes in descending order of average ARD values, and finally generate a rank list S. No genes are eliminated at this iteration, but the ranked list S is produced according to average ARD values. This ranked list will be used in subsequent forward selection.

In the phase of forward selection, we add one top-ranked gene in R each time into a gene subset r, and carry out LOO cross validation using GP-Laplace without ARD with genes in r as input variables. Alternatively, SVM can also be used in forward selection. In either case, the gene subset with the lowest LOO error is identified as the minimal subset of biomarkers.

In summary, the advantages of using GP-ARD-RFE over GP-ARD-FS include:

(1) Feature elimination utilizing average ARD values is more reliable than Wilcoxon rank sum test due to the low sample size setting of gene expression data.

	All genes	373 genes (GP-ARD-FS)	1024 genes (GP-ARD-RFE)	512 genes (GP-ARD-RFE)
LOO Accuracy	80.65	82.26	88.71	95.16

Table 5.4: Ineffectiveness of Wilcoxon rank sum (a preprocessing step used by GP-ARD-FS) in eliminating irrelevant features. LOO Accuracies using different subsets of genes: (1) "All genes"; (2) top 373 genes selected by Wilcoxon rank sum test with $p = 0.01$; (3) top 1024 genes selected by GP-ARD-RFE; (4) top 512 genes selected by GP-ARD-RFE. LOO accuracy with the top 373 genes is not significantly higher than the accuracy without feature selection, and is even lower than the accuracy with the top 1024 genes selected by GP-ARD-RFE.

Table 5.4 displays the LOO error using different subsets of genes on Colon data. In the preprocessing step of GP-ARD-FS, there are 373 genes significantly differentially expressed in the rank sum test at the significance level of $p = 0.01$. The LOO accuracy using these 373 genes is 82.26%, showed in the second column, is not significantly greater than the LOO accuracy (80.65%)using all genes. However, if we use GP-ARD-RFE to eliminate irrelevant genes, the LOO accuracies are significantly improved: the LOO accuracy with top 1024 genes is 88.71%, and the accuracy with top 512 genes is 95.16%. We report these two accuracies in the third and fourth column. To sum up, rank sum is not as reliable as GP-ARD-RFE in eliminating irrelevant features.

(2) ARD values are normalized so that ARD values from each training contribute fairly to the average.

(3) GP-ARD-RFE is more efficient than both MSVM-RFE and GP-ARD-FS in terms of CPU cost. Take GP-ARD-FS as an example, the most costly step is step (3.1.4)—carrying out LOO validation on every training set. Suppose there are N training samples in D-dimensional space, then there will be a total of $K \cdot N \cdot D$ validations to be performed for K-fold experiments. Whereas, in GP-ARD-RFE, costly forward selection is only performed once at the end.

(4) In GP-ARD-FS, many important features can be discarded in step (3.1.5),

as usually the minimal gene set M_i is very small and only features in M_i are candidates of the final biomarker set M^*. Instead of finding the optimal minimal gene set at an early stage, we use a more conservative approach—we discard unimportant features to avoid possible exclusion of relevant features.

There are some variations of our algorithm. For example, GP-EP rather than GP-Laplace can be used as the classifier in forward selection. Moreover, if multiple subsets achieve the same LOO accuracy, we can use the posterior probabilities to break the tie.

5.5 Experimental Results

We conduct an experimental evaluation on various feature selection algorithms using Colon cancer data set and Leukemia data set. We exclude Lung data set in the comparisons as its dimensionality is too high for some algorithms as we discussed in Chapter 4. We evaluate the following feature selection methods: SVM-RFE, RVM, SMLR, GP-Laplace, MSVM-RFE, GP-ARD-FS, and GP-ARD-RFE, where the last three methods utilize cross validation. We use two criteria to do the evaluations: classification accuracy using biomarkers and stability in selecting features.

5.5.1 Comparison Using Stability

In Section 5.1, we compute the stability scores of RVM and SMLR using Colon and Leukemia data sets. We first split the data into 10 folds, and each time one fold is left out and the remaining 9 folds are used for training. We end up with 10 different optimal biomarker sets returned by the 10 different training sets. We then compute the stability score of the 10 biomarker sets using the formula 5.1.

As a side effect, we may find some potential outlying samples at the Step 3.1 presented in Table 5.6, where we run forward selection to identify biomarkers.

	Colon	Leukemia
SVM-RFE	0.12	0.14
GP-LAP	0.18	0.11
RVM	0.21	0.15
SMLR	0.24	**0.32**
MSVM-RFE	0.14	0.08
GP-ARD-FS	**0.27**	0.12
GP-ARD-RFE	**0.27**	0.28

Table 5.5: Stability scores of various feature selection methods on the Colon and Leukemia data sets. SMLR and GP-ARD-RFE are consistently more stable than other methods.

For example, we found that when we compute the LOO error using biomarker sets, one sample in the Colon data set was constantly classified incorrectly using any biomarker set.

Table 5.5 presents the stability scores of the five feature selection algorithms including RVM, SMLR, MSVM-RFE, GP-ARD-FS, and GP-ARD-RFE on Colon and Leukemia data sets. We found GP-ARD-RFE and SMLR outperform other feature selection algorithms in terms of stability. Therefore, the genes selected by GP-ARD-RFE and SMLR are more trustworthy.

5.5.2 Comparison Using Classification Accuracy

There are two ways to evaluate the feature selection methods using classification accuracies:

(1) We can use all data as training data, select the biomarker set, and then compute the LOO accuracy with biomarker sets only using a classifier. This evaluation technique is used in [8]. The flaw of using this type of evaluation is that it does not reveal the generalization ability of the algorithms because all data was used to select biomarkers.

(2) We first split the data D into 10 folds to get f1, f2, ... , f10, and then

	Accuracy (Colon)	Accuracy (Leukemia)
All genes	72.6% (17/62)	**97.2% (2/72)**
SVM-RFE	77.4% (14)	87.5% (9)
GP-Lap	75.8% (15)	88.9% (8)
RVM	83.9% (10)	90.2% (7)
SMLR	72.6% (17)	**97.2% (2)**
MSVM-RFE	79.0% (13)	94.4% (4)
GP-ARD-FS	79.0% (13)	93.1% (5)
GP-ARD-RFE	**87.1% (8)**	**97.2% (2)**

Table 5.6: Leave-One-Fold-Out accuracies using biomarkers selected by various feature selection methods. The number in brackets are the total number of errors in 10 folds. Part of the results are included in Table 4.7. GP-ARD-RFE achieves the highest accuracies on both data sets.

leave the fold $f_i (i = 1, 2, ..., 10)$ out, select biomarker set b_i using the remaining 9 folds. We then train the 9 folds with b_i only using several classifiers and compute the number of errors when classifying the test data f_i. The accuracy is computed as: $\frac{N-N_error}{N} \times 100\%$, where N is the total number of samples in the whole data set D, and N_error is the total number of errors in 10 folds. This evaluation techniques are used in [23]. Unlike the first evaluation strategy, the second strategy reveals generalization ability of the feature selection algorithms as the test data is not used to select biomarkers.

Table 5.6 presents the results using the second evaluation strategy. MSVM-RFE extends SVM-RFE by utilizing cross validation, similarly, GP-ARD-FS and GP-ARD-RFE extend previous GP-ARD. According to Table 5.6, all of the three feature selection methods that incorporate cross validation techniques outperform their ascendants without cross validation. We find that GP-ARD-RFE outperforms other feature selection methods in terms of both accuracies and stabilities.

1. **Preprocessing** (1.1) Use the Wilcoxon rank sum test as a criterion to measure the variance of the expression values in different classes for each gene.
   (1.2) Eliminate genes that are not significantly differentially expressed in the rank sum test at the significance level of $p = 0.01$. Suppose the new data set after elimination is $D = \{(x_i, y_i)|i = 1, ..., n\}$ and $x_i \in \mathbb{R}^d$.
2. **Initialization** (2.1) Split the data set D into k mutually disjoint folds. Let $D^{\setminus i}$ be the training set containing all folds but the i-th one.
   (2.2) Initialize $i = 1$.
3. **Loop** (3.1) While $i \leq k$, let $D^{\setminus i}$ be the training data
   (3.1.1) Train GP-ARD-Laplace using $D^{\setminus i}$: maximizing the evidence $P(D^{\setminus i}|\theta)$ several times and choose the one with the highest evidence as the optimal ARD values θ_i^*.
   (3.1.2) Sort the genes in descending order of ARD values θ_i^*, and get the ranked gene list $R_i \in \mathbb{R}^d$.
   (3.1.3) Generate d gene subsets by adding one top-ranked gene in R_i each time such that subset s_{i1} contains the first gene in R_i, and s_{i2} contains the top-2 genes in R_i, and so on.
   (3.1.4) For each gene subset s_{ij}, $j = 1, ..., d$, carry out the LOO cross validation using GP-Laplace (without ARD) on $D^{\setminus i}$ with only genes in s_{ij}.
   (3.1.5) Let set M_i be the minimal gene subset that yielded the minimal LOO error in (3.1.4).
   (3.1.6) $i = i + 1$
4. **Ranking** (4.1) Rank the genes in the union set of $\{M_i\}_{i=1}^k$ by the number of times each gene was selected in the k subsets $\{M_i\}_{i=1}^k$. Genes with the same number of occurrences are further ranked by their average ARD values. Refer the ranked list as R.
5. **Selection** (5.1) Run forward selection on R, which means repeating steps (3.1.3)-(3.1.4) with R_i replaced by R and $D^{\setminus i}$ by D.
   (5.2) Let set M^* be the minimal gene subset that yielded the minimal LOO error in (5.1).
6. **Exit** Return the minimal subset of relevant genes M^*.

Figure 5.3: Algorithm of GP-ARD-FS [20]

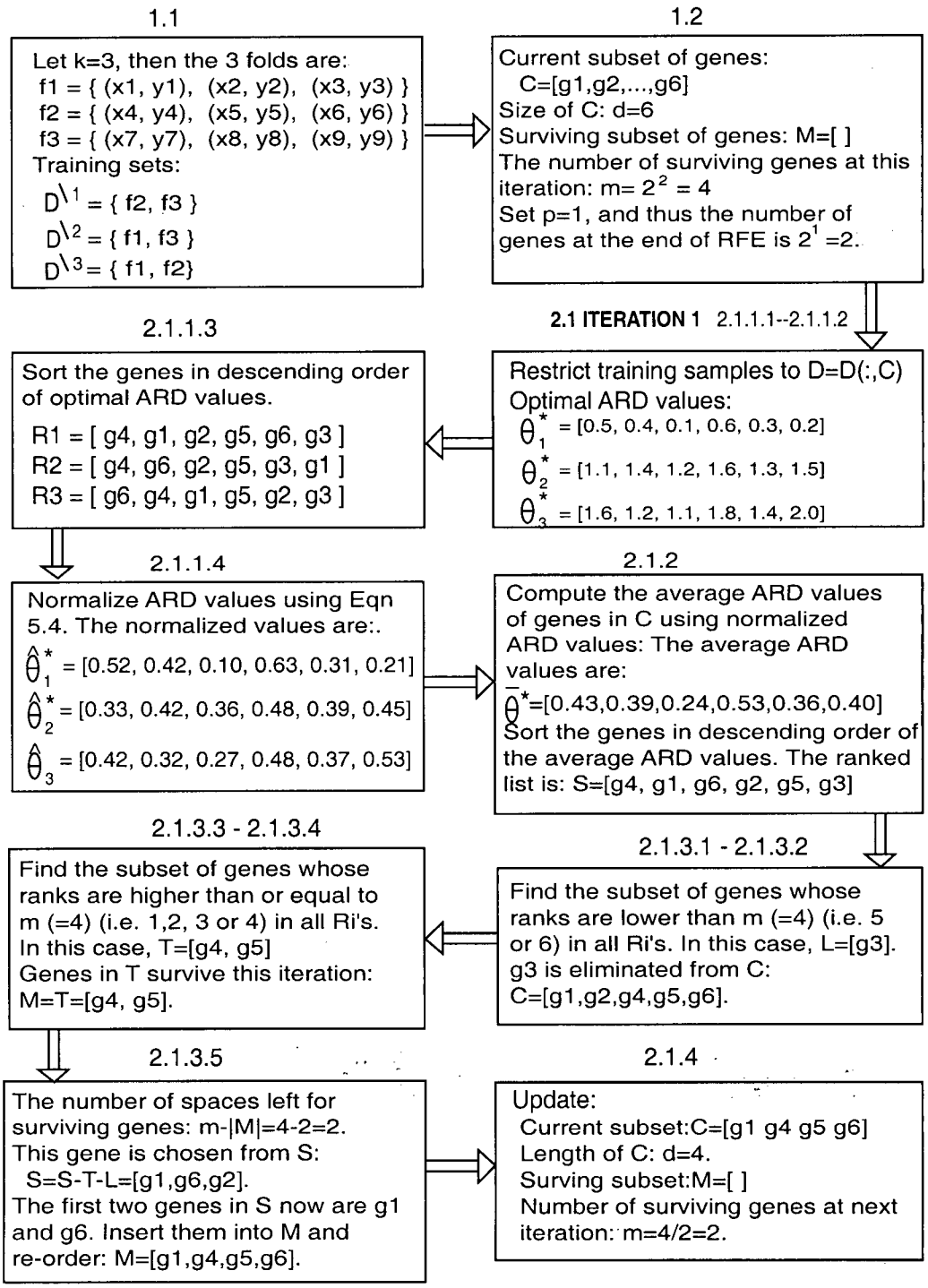


Figure 5.4: Illustration of feature elimination of GP-ARD-RFE algorithm: a running example (Part 1)

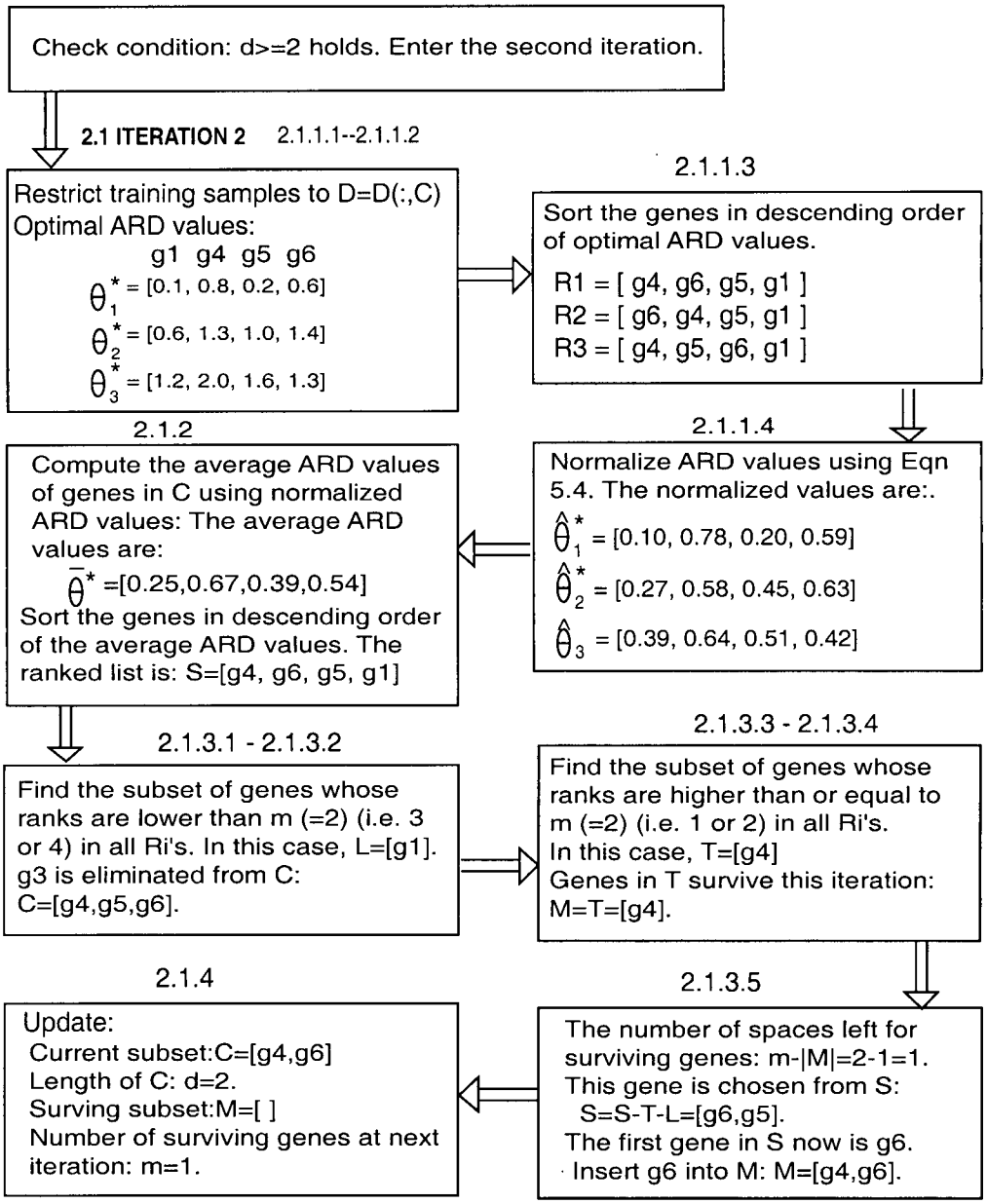


Figure 5.5: Illustration of feature elimination of GP-ARD-RFE algorithm: a running example (Part 2)

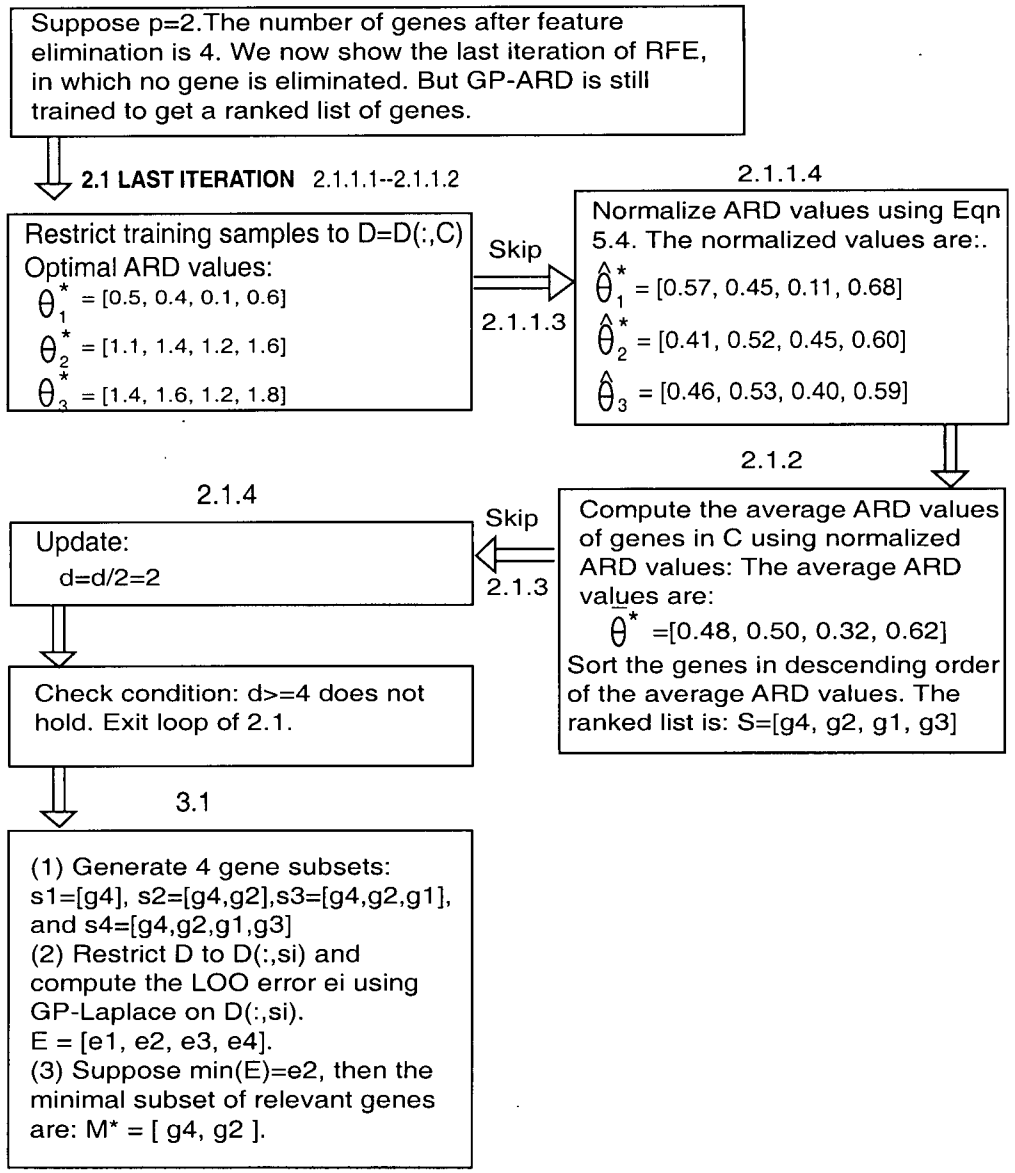


Figure 5.6: Illustration of GP-ARD-RFE algorithm: the steps after feature elimination

1. Initialization (1.1) Split the data set D into k mutually disjoint folds. Let $D^{\setminus i}$ be the training set containing all folds but the i-th one.
(1.2) Set current subset of genes $C = [1, ..., D]$, and $d = |C|$
Let the subset of genes which will survive this iteration be $M = []$
Set the number of surviving genes at this iteration to $m = 2^{\lfloor \log_2(D) \rfloor}$
Set user defined argument $p = 8$

2. RFE (2.1) While $d \geq 2^p$
(2.1.1) Initialize $i = 1$, and Loop: while $i < k$
(2.1.1.1) Restrict training samples $D^{\setminus i}_{new}$ to be $D^{\setminus i}(:, C)$.
(2.1.1.2) Train GP-ARD-Laplace using $D^{\setminus i}_{new}$: maximizing the evidence $P(D^{\setminus i}_{new}|\theta)$ several times and choose the one with the highest evidence as the optimal ARD values θ^*_i.
(2.1.1.3) If $d > 2^p$, do the following:
Sort the genes in descending order of ARD values θ^*_i, and get the ranked gene list $R_i \in \mathbb{R}^d$.
(2.1.1.4) Normalize θ^*_i using Equation 5.4, and get the normalized optimal ARD values $\hat{\theta}^*_i$.
(2.1.2) Compute the average ARD values $\bar{\theta}$ using $\hat{\theta}^*_i$, where $i = 1, ..., k$. Rank the genes in C by the average ARD values, Let S be the ranked list of genes.
(2.1.3) If $d > 2^p$, do elimination as follows:
(2.1.3.1) Find the subset of genes whose ranks are lower than m in all $R_i, i = 1 : k$. Let L be the set containing these common low ranked genes. Genes in L will be eliminated at this iteration regardless of their average ARD values.
(2.1.3.2) Remove genes in L from C: $C = C - L$.
(2.1.3.3) Find the subset of genes whose ranks are higher than or equal to m in all $R_i, i = 1 : k$. Let T be the set containing these genes. Genes in T will not be eliminated at this iteration regardless of their average ARD values.
(2.1.3.4) Update surviving set $M = T$;
(2.1.3.5) The number of genes in set M now is $|T|$. $S = S - T - L$.
Insert the top $m - |T|$ genes in S into M so that $|M|$ is m.
(2.1.4) If $d > 2^p$ update:
Current subset of features $C = M$;
The length of current subset $d = |C|$
Subset of surviving features $M = []$
The number of surviving features $m = d/2$.
Otherwise, update $d = d/2$ (to exit the loop of 2.1).

3. **Selection** (3.1) Run forward selection on R as in (5.1) of Figure 5.3:
(3.1.1) Generate $d = 2^p$ subset of genes by adding one top ranked gene each time.
(3.1.2) For each gene subset, compute LOO error using D.
(3.1.3) Let set M^* be the minimal gene subset that yielded the minimal LOO error in (3.1.2).

4. **Exit** Return the minimal subset of relevant genes M^*.

Figure 5.7: Algorithm of GP-ARD-RFE

Chapter 6

Conclusion and Future Work

In this thesis, we did a comparative study on kernel methods with application to gene classification and selection. Gene expression data obtained by high-throughput techniques poses three challenges: high dimensionality, low sample size, and poor quality. Kernel methods outperform other approaches in gene expression data analysis due to their ability to handle high dimensionality easily. Experiments show that all kernel methods (considered in this thesis) are efficient in classifying high dimensional data without feature selection or feature weighting. For a data set in 19624 dimensions, all kernel methods finish the classification in about 2 minutes. In terms of classification accuracies, our experiments with three real gene expression data show that SVM and Gaussian process classifiers with Expectation Propagation outperform their counterparts.

Compared with gene classification, biomarker identification is a more important problem in gene expression analysis. We investigate four kernel-based methods in feature selection or feature ranking. Sparse kernel methods, such as RVM and SMLR, obtain sparse solutions for regression and classification by utilizing sparsity-promoting priors. If original features or their transformed forms are used as basis functions, RVM and SMLR automatically select an optimal subset of genes, which

is also known as biomarker set. Unlike RVM or SMLR, SVM-RFE and GPC with ARD ranks all features without selecting an optimal subset. Forward selection has to be used to identify biomarkers.

To evaluate the quality of selected features, classification accuracy is commonly used in the literature. However, this criterion only measures one desired characteristic of good feature selection methods. That is, the elimination of irrelevant features does not adversely affect the discriminative capability of the two groups of data. Another important property of a good feature selection method is stability, which measures how sensitive a method is to small changes in the training data. If one sample in the training set is changed, the biomarker set selected by a feature selection method is changed dramatically, then the method is not stable. In turn, the features selected by this method are not trustworthy. Therefore, we evaluate the feature selection algorithms using both of the two criteria.

The costs of low sample size can be reduced by utilizing cross validation techniques. By using cross validation, we make full use of the available data and aggregate the performance from multiple trainings. MSVM-RFE is the cross validation improvement for SVM-RFE, and GP-ARD-FS and GP-ARD-RFE are two cross validation improvements for GPC with ARD. Experimental results show that by utilizing cross validation techniques, feature selection methods are improved in terms of both accuracies and stabilities. The proposed algorithm GP-ARD-RFE extends from GP-ARD-FS by replacing the useless preprocessing step using Wilcoxon rank sum tests by a step of recursive feature elimination utilizing ARD values. As a side effect of GP-ARD-RFE, we may identify some potential outliers. However, the obstacle of the poor quality of gene expression data is not systematically conquered. We would like to explore this problem in future work.

Bibliography

[1] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Science*, 96:6745–6750, 1999.

[2] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundation of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[3] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

[4] R. E. Bellman. *Adaptive Control Processes.* Princeton University Press, Princeton, NJ, 1961.

[5] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, ACM Press, 1992.

[6] C. Chang and C. Lin. Libsvm: a library for support vector machines, 2001. Software available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[7] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. Technical report, Gatsby Unit, University College London, 2004.

[8] W. Chu, Z. Ghahramani, F. Falciani, and D. L. Wild. Biomarker discovery in microarray gene expression data with Gaussian processes. *Bioinformatics*, 21(16):3385–3393, 2005.

[9] K. K. Chin. *Support Vector Machines Applied to Speech Pattern Classification.* Master's thesis, Darwin College, Cambridge University, 1999.

[10] N. Cristianini, H. Lodhi, and J. Shawe-Taylor. Latent semantic kernels. *Journal of Intelligent Information Systems (JJIS)*, 18(2-3):127–152, 2002.

[11] L. Csato and M. Opper. Sparse online Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[12] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, Cambridge, UK, 2000.

[13] L. Deng, J. Pei, J. Ma, and D. Lee. A Rank Sum Test Method for Informative Gene Discovery. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 410–419, 2004.

[14] K. Duan, J. C. Rajapakse, H. Wang, and F. Azuaje. Multiple SVM-RFE for Gene Selection in Cancer Classification with Expression Data. *IEEE Transactions on Nanobioscience*, 4(3):228–34, September 2005.

[15] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. Technical Report, HP Labs, 2003.

[16] T. Furey, N. Cristianini, N. Duffy, D. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16:906-914, 2000.

[17] S. H. Friend and R. B. Stoughton. The Magic of Microarrays. *Scientific American*, pages 44–53, February 2002.

[18] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157-1182, 2003. Special issue on variable and feature selection.

[19] T. R. Golub, D. K. Slomin, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caliguiri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531-537, 1999.

[20] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[21] S. Hochreiter and K. Obermayer. Gene selection for Microarray data. In B. Schölkopf, K. Tsuda, and J. Vert, editors, *Kernel Methods in Computational Biology*, pages 319-355. MIT Press, 2004.

[22] B. Krishnapuram. *Adaptive classifier design using labeled and unlabeled data.* PhD Thesis, Department of Electrical Engineering, Duke University, 2004.

[23] B. Krishnapuram, L. Carin, M. A. T. Figueiredo, and A. J. hartemink. Sparse multinomial logistic regression: fast algorithms and generalization bounds. *IEEE Transaction on pattern analysis and machine intelligence*, 27(6):957–968, 2005.

[24] B. Krishnapuram, L. Carin, and A. Hartemink. Gene expression analysis: joint feature selection and classifier design. In B. Schölkopf, K. Tsuda, and J. Vert, editors, *Kernel Methods in Computational Biology*, pages 299–318. MIT Press, 2004.

[25] H. Kim and Z. Ghahramani. The EM-EP algorithm for Gaussian process classification. In *Proceedings of the Workshop on Probabilistic Graphical Models for Classification at ECML*, 2003.

[26] T. Li, C. Zhang, and M. Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15): 2429–2437, 2004.

[27] D. J. C. MacKay. A practical Bayesian framework for back propagation networks. *Neural Computation*, 4(3):448–472, 1992.

[28] D. J. C. MacKay, Bayesian methods for backpropagation networks. *Models of Neural Networks* III, pages 211–254, 1994.

[29] D. J. C. Mackay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–165. Berlin, Springer-Verlag, 1998.

[30] T. P. Minka. *A family of algorithm for approximate Bayesian inference.* Ph.D. thesis, Massachusetts Institute of Technology, January 2001.

[31] J. S. Marron and M. Todd. Distance Weighted Discrimination. http://www.unc.edu/depts/statistics/postscript/papers/marron/HDD/DWD/DWD1.pdf, 2002.

[32] J. Ma, Y. Zhao, and S. Ahalt. OSU SVM Classifier Matlab Toolbox, version 3.00, http://www.ece.osu.edu/~maj/osu_svm/, 2002.

[33] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

[34] R. M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer, 1996.

[35] R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report No. 9702, Department of Statistics, University of Toronto, 1997.

[36] P. Pavlidis, J. Weston, J. Cai, and W. Grundy. Gene functional classification from heterogeneous data. In *Proceeding of the fifth annual international conference on computational biology*, pages 249–255. ACM Press, 2001.

[37] Y. Qi, T. P. Minka, R. W. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 671–678, 2004.

[38] C. R. Rasmussen. Gaussian process in machine learning. Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003. In O. Bousquet, U. von Luxburg, and G. Rätsch, editors, *Revised Lectures* 3176:63–71. Springer-Verlag, Heidelberg, 2004.

[39] M. Seeger. Notes on Minka's expectation propagation for Gaussian process classification. Technical report, University of Edinburgh, 2002.

[40] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.

[41] B. Schölkopf. *Support vector learning.* Munich, Oldenbourg Verlag, 1997.

[42] B. Schölkopf and A. J. Smola. *Learning with Kernels.* Cambridge, MA, MIT Press, 2002.

[43] C. J. Stone. Optimal rates of convergence for nonparametric estimators. *Annals of Statistics*, 8(6):1348–1360, 1980.

[44] M. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:221–244, 2001.

[45] V. N. Vapnik. *The Nature of Statistical Learning Theory.* New York, Springer Verlag, 1995. ISBN 0-387-94559-8.

[46] V. E. Velculescu, L. Zhang, B. Vogelstein, and K. W. Kinzler. Serial analysis of gene expression. *Science*, 270:484-487, 1995.

[47] C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342-1351, 1998.