ABSTRACT

HUANG, LINGKANG. Variable Selection in Multi-class Support Vector Machine and Applications in Genomic Data Analysis. (Under the direction of Professors Zhao-Bang Zeng and Hao Helen Zhang).

Microarray techniques provide new insights into cancer diagnosis using gene expression profiles. Molecular diagnosis based on high-throughput genomic data sets presents a major challenge due to the overwhelming number of variables and complex multi-class nature of tumor samples. In this dissertation, the author first tackled a multi-class problem related to liver toxicity severity prediction using the Random Forest and GEMS-SVM (Gene Expression Model Selector using Support Vector Machine). However, the original SVM regularization formulation does not accommodate the variable selection. Most existing approaches, including GEMS-SVM, handle this issue by selecting genes prior to classification, which does not consider the correlation among genes since they are selected by univariate ranking. In this dissertation, the author developed new multi-class SVM (MSVM) approaches which can perform multi-class classification and variable selection simultaneously and learn optimal classifiers by considering all classes and all genes at the same time. The original multi-class SVM proposed by Crammer and Singer (2001) does not perform the variable selection. By using the MSVM loss function proposed by Crammer and Singer (2001), the author developed new variable selection approaches for both linear and non-linear classification problems. For linear classification problems, four different sparse regularization terms were included in the objective function respectively. For nonlinear classification problems, two different approaches have been developed to tackle them. The first approach was used in non-linear MSVMs via basis function transformation. The second approach was used in non-linear MSVMs via kernel functions. The newly developed methods were applied to both simulation and real microarray data sets. The results demonstrated that the methods presented in this dissertation could select a much smaller number of genes, compared with other existing methods, with high classification accuracy to predict the tumor subtypes. The combination of high accuracy and small number of genes makes the new methods as powerful tools for disease diagnostics based on expression data and target identifications of the therapeutic intervention.

Variable Selection in Multi-class Support Vector Machine and Applications in Genomic Data Analysis

by
Lingkang Huang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Doctor of Philosophy

Bioinformatics and Statistics

Raleigh, North Carolina

2008

APPROVED BY:

Dr. Zhao-Bang Zeng Chair of Advisory Committee	Dr. Hao Helen Zhang Co-Chair of Advisory Committee
Dr. Gregory C. Gibson	Dr. Russell D. Wolfinger
Dr. Pierre R. Bushel	

DEDICATION

To My Family

BIOGRAPHY

Lingkang Huang was born in Baoji, China. She graduated from Fudan University with a bachelor's degree in Microbiology. She obtained a master's degree in Entomology (in the field of Virology and Molecular Genetics) from University of Kentucky and a master's degree in Computer Science from Kansas State University. Then she entered the Statistical Genetics and Bioinformatics program at North Carolina State University in 2003 and pursued a Ph.D. co-major in Statistics and Bioinformatics. In the meantime, she had four years intern experience at the National Institute of Environmental Health Sciences (NIEHS) as a predoctoral student from 2003 to 2007. Currently, she is full-time working at GlaxoSmithKline as a principle scientist in the pharmacogenetics division.

ACKNOWLEDGMENTS

I would like to express sincere gratitude to my advisors Dr. Zhao-Bang Zeng and Dr. Hao Helen Zhang for their constant guidance, encouragement and support in my study. I also thank the other committee members Dr. Gregory C. Gibson, Dr. Russell Wolfinger and Dr. Pierre R. Bushel for their insights and comments on my dissertation. Special thanks go to Dr. Pierre R. Bushel, for his kind support and guidance in my four-year intern at NIEHS. It has been a pleasant experience to study in this program/university and work at NIEHS as an intern. I would like to convey my gratitude to all the people who have made the resources available to me, without which I would not be able to complete my study. Finally, I would like to thank my husband and my parents in China for their unconditional love, encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Statistical learning or machine learning is the task of learning knowledge or patterns from large data repositories. In a typical setting, the data has a set of features with an outcome measurement. The outcome measurement can be either quantitative or categorical. The goal of statistical or machine learning is to learn a model which can predict the outcome based on the given features. Usually, the training data with the known outcome measurement is first given to train the model, then the learned model is used to predict the outcomes of future data with unknown outcomes. A good model should be able to predict the outcome of future data with high precision.

Learning from data can be categorized into supervised and unsupervised learning based on whether the outcome measurement is utilized or not. Unsupervised learning methods, such as clustering, self-organizing maps and associations rule, do not utilize outcome measurements while the supervised learning methods do. Supervised learning problems can be further categorized into regression problems (with quantitative outcome measurements) and classification problems (with categorical outcome measurements). Popular methods for regression problems include parametric methodologies such as linear regression, logistic regression, generalized linear models, etc. and non-parametric methodologies such as smoothing spline, kernel methods, local polymorphism, etc. Popular methodologies for classification include support vector machines (SVM), K-nearest neighbor (KNN) and tree based methods such as CART, Random Forest, and etc.

The general statistical learning problem is defined as follows: given a training data set containing n pairs of observations $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, which are $i.i.d.$ from $P(\mathbf{x}, y)$, $\mathbf{x} \in R^p$ and $y \in R$ or $y = \{1, 2, \ldots, K\}$, we want to choose the best function from

$\{\hat{f}(\mathbf{x}, \alpha), \alpha \in \Lambda\} : R^p \to R$ *or* $R^p \to \{1, 2, \ldots, K\}$. A loss function is used to measure the goodness-of-fit of the learned model. The "loss" usually means the error, which is defined as the discrepancy between Y and $f(\mathbf{X}, \alpha)$. There are many ways to measure the discrepancy using loss functions L. For examples, the loss functions for regression problems are

- Square loss: $L(Y, f(\mathbf{X})) = (Y - f(\mathbf{X}))^2$
- Absolute error loss: $L(Y, f(\mathbf{X})) = |Y - f(\mathbf{X})|$
- L_p loss: $L(Y, f(\mathbf{X})) = |Y - f(\mathbf{X})|^p$.

The loss functions for classification problems are

- Hinge loss: $L(Y, f(\mathbf{X})) = [1 - Yf(\mathbf{X})]_+$
- Exponential loss: $L(Y, f(\mathbf{X})) = e^{-Y \cdot f(\mathbf{X})}$
- 0-1 loss:

$$L(Y, f(\mathbf{X})) = \begin{cases} 0 & \text{if} \quad Y = f(\mathbf{X}) \\ 1 & \text{if} \quad Y \neq f(\mathbf{X}). \end{cases}$$

The goal of learning is to minimize the risk function or average loss. The risk function $R[f]$ is defined as the expected loss: $R[f] = E_{\mathbf{X},Y} L(Y, f(\mathbf{X})) = \int L(y, f(\mathbf{x})) \mathrm{d}P(\mathbf{x}, y)$ (Vapnik, 1999). The $R[f]$ is not computable since $P(X, Y)$ is generally unknown. However, since the training data set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ are *i.i.d.* samples from the population $P(X, Y)$, in practice we minimize the average loss over the training data set, called empirical risk function $R_{emp}[f]$, $R_{emp}[f] = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$. From the law of large number (LLN), when $n \to \infty$, $R_{emp}[f] \to R[f]$ (Vapnik, 1999).

1.1 Classification

As stated before, supervised learning with the categorical outcome measurement is characterized as classification problems. A classification problem can be further categorized into binary classification and multi-class classification.

For binary classifications, Y can take only two values. Usually, $Y \in \{0, 1\}$ or $Y \in \{-1, +1\}$. For multi-class classifications, Y can take more than two values, $Y \in \{1, 2, \ldots, K\}$, $K > 2$. The goal of learning in multi-class classifications is to choose the

best function from $\{\hat{f}(\mathbf{x}, \alpha), \alpha \in \Lambda\} : R^p \to \{1, 2, \ldots, K\}$. The loss function L for multi-class classifications is a $K \times K$ matrix. The element $L(r, l)$ is the cost paid for misclassifying the subject in class r to class l. The $L(r, l)$ may not be equal to $L(l, r)$ since different costs may be paid for misclassifying class r to class l and misclassifying class l to class r. For example, suppose we are studying breast cancer with severity level I, II and III. Thus, all subjects will be categorized into 4 classes: normal without breast cancer, breast cancer level I, level II, or level III. Naturally speaking, misclassification of patient with breast cancer level I to normal healthy person will pay more cost than misclassification of normal healthy person to breast cancer level I. In a special case, if we use the 0-1 loss, the L results in a matrix with all 1s except 0s on the diagonal of the matrix.

1.1.1 Bayes Classification Rule

The theoretical optimum classification rule is called the Bayes rule. It serves as the golden standard to evaluate the performance of the trained classifiers. In this section, we discuss bayes rules for both binary and multi-category classification. We first state the bayes rule for equal cost cases and then extend it to unequal cost cases. The derivation can be found in Devroye et al. (1996).

Binary Classification

Equal Cost Case

For the equal cost case, misclassifying the sample in class 1 to class 0 or misclassifying the sample in class 0 to class 1 is given the same cost/penalty. For binary classification problems, if we use the 0-1 loss, the loss function is an indicator function, $L(Y, f(\mathbf{X})) = I(Y \neq f(\mathbf{X}))$. Therefore, the risk is

$$\begin{aligned} R[f] &= E_{\mathbf{X},Y} I(Y \neq f(\mathbf{X})) \\ &= E_{\mathbf{X}} E_{Y|\mathbf{X}} I(Y \neq f(\mathbf{X})) \\ &= E_{\mathbf{X}}[P(Y = 1|\mathbf{X}) I(f(\mathbf{X}) = 0) + P(Y = 0|\mathbf{X}) I(f(\mathbf{X}) = 1)]. \end{aligned}$$

To minimize the risk function $R[f]$, the optimal classification rule is

$$\phi_B(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad P(Y=1|X=\mathbf{x}) > P(Y=0|X=\mathbf{x}), \\ 0 & \text{if} \quad P(Y=0|X=\mathbf{x}) > P(Y=1|X=\mathbf{x}). \end{cases}$$

The above bayes rule can also be written as $\phi_B(\mathbf{x}) = I[P(Y=1|X=\mathbf{x}) > 0.5]$. The error rate of ϕ_B is called Bayes error.

If the prior probability is defined as $\pi_1 = P(Y=1)$, $\pi_0 = P(Y=0)$ and the class densities for Class 1 and Class 0 are defined as $f_1(\mathbf{x}) = P(X=\mathbf{x}|Y=1)$ and $f_0(\mathbf{x}) = P(X=\mathbf{x}|Y=0)$ respectively, the posterior probability can be calculated as the following:

$$P(Y=1|X=\mathbf{x}) = \frac{P(X=\mathbf{x}|Y=1)P(Y=1)}{P(X=\mathbf{x})} \propto \pi_1 f_1(\mathbf{x}),$$
$$P(Y=0|X=\mathbf{x}) = \frac{P(X=\mathbf{x}|Y=0)P(Y=0)}{P(X=\mathbf{x})} \propto \pi_0 f_0(\mathbf{x})$$

where the marginal density $P(X=\mathbf{x}) = \pi_1 f_1(\mathbf{x}) + \pi_0 f_0(\mathbf{x})$.

Therefore, the bayes rule is

$$\begin{aligned} \phi_B(\mathbf{x}) &= \begin{cases} 1 & \text{if} \quad \pi_1 f_1(\mathbf{x}) > \pi_0 f_0(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if} \quad \dfrac{f_1(\mathbf{x})}{f_0(\mathbf{x})} > \dfrac{\pi_0}{\pi_1} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Unequal Cost Case

In the real world, misclassifying a person with tumor to "normal" is more severe than misclassifying a normal person into "cancer". Similarly, misclassifying true emails into "spam" is much worse than misclassifying spam into "email". In such situations, we need to assign different cost/penalty for different types of misclassification. For unequal cost situations, we define $C(0,1)$ as the cost for misclassifying a sample from Class 0 to Class 1 and $C(1,0)$ as the cost for misclassifying a sample from Class 1 to Class 0. Then the risk

function is

$$\begin{aligned} R[f] &= E_{\mathbf{X},Y} C(Y, f(\mathbf{X})) \\ &= E_{\mathbf{X}} E_{Y|\mathbf{X}} C(Y, f(\mathbf{X})) \\ &= E_{\mathbf{X}}[C(1,0)P(Y=1|\mathbf{X})I(f(\mathbf{X})=0) + C(0,1)P(Y=0|\mathbf{X})I(f(\mathbf{X})=1)]. \end{aligned}$$

And, the bayes rule is

$$\phi_B(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad C(1,0)P(Y=1|X=\mathbf{x}) > C(0,1)P(Y=0|X=\mathbf{x}), \\ 0 & \text{if} \quad C(1,0)P(Y=0|X=\mathbf{x}) > C(0,1)P(Y=1|X=\mathbf{x}). \end{cases}$$

Alternatively speaking,

$$\begin{aligned} \phi_B(\mathbf{x}) &= \begin{cases} 1 & \text{if} \quad \dfrac{P(Y=1|X=\mathbf{x})}{P(Y=0|X=\mathbf{x})} > \dfrac{C(0,1)}{C(1,0)}, \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if} \quad P(Y=1|X=\mathbf{x}) > \dfrac{C(0,1)}{C(0,1)+C(1,0)}, \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

When $C(1,0) \gg C(0,1)$, the classifier tends to categorize all subjects to Class 1; vice versa, when $C(1,0) \ll C(0,1)$, the classifier tends to categorize all subjects to Class 0.

Multi-class Classification Problems

The loss function in multi-class classifications is a $K \times K$ matrix, with $C(r,l)$ representing the cost of misclassifying a subject from Class r to Class l. The Bayes rule is

$$\begin{aligned} \phi_B(\mathbf{x}) &= \arg\min_f E_{X,Y} C(Y, f(\mathbf{X})) \\ &= \arg\min_f E_X E_{Y|\mathbf{X}} C(Y, f(\mathbf{X})), \end{aligned}$$

where $E_{Y|\mathbf{X}} C(Y, f(\mathbf{X})) = \sum_{k=1}^{K} I(f(\mathbf{X})=k) \sum_{l=1}^{K} C(l,k) P(Y=l|\mathbf{X})$.

Equal Cost Case

A special case is the 0-1 loss, where $C(k,l) = I(k \neq l)$. As we stated before, the

risk function is

$$\begin{aligned} E_{Y|\mathbf{X}}C(Y, f(\mathbf{X})) &= \sum_{k=1}^{K} I(f(\mathbf{X}) = k) \sum_{l=1}^{K} C(l,k) P(Y = l|\mathbf{X}) \\ &= \sum_{k=1}^{K} I(f(\mathbf{X}) = k) \sum_{l=1}^{K} P(Y = l|\mathbf{X}) \\ &= \sum_{k=1}^{K} I(f(\mathbf{X}) = k)[1 - P(Y = k|\mathbf{X})]. \end{aligned}$$

The Bayes rule is $\phi_B(\mathbf{x}) = \arg\min_f E_X E_{Y|X} C(Y, f(\mathbf{X}))$. Hence, it can be expressed as the following (Lee et al., 2004)

$$\phi_B(\mathbf{x}) = k^\star \text{ if } k^\star = \arg\min_{k=1,\ldots,K} [1 - P(Y = k|X = \mathbf{x})].$$

This is equivalent to, $P(Y = k^\star|\mathbf{x}) = \max_{k=1,\ldots,K} P(Y = k|X = \mathbf{x})$.

Unequal Cost Case

If C(k,l) is an unequal cost matrix, then

$$E_{Y|\mathbf{X}} L(Y, f(\mathbf{X})) = \sum_{k=1}^{K} I(f(\mathbf{X}) = k) \sum_{l=1}^{K} C(l,k) P(Y = l|\mathbf{X}).$$

The Bayes rule is $\phi_B(\mathbf{x}) = \arg\min_f E_X E_{Y|\mathbf{X}} C(Y, f(\mathbf{X}))$. Hence, it can be expressed as the following (Lee et al., 2004)

$$\phi_B(\mathbf{x}) = \arg\min_{k=1,\ldots,K} \sum_{l=1}^{K} C(l,k) P(Y = l|X = \mathbf{x})$$

Decision Boundary

For equal costs, in a binary classification problem the decision boundary of the Bayes rule is $\{\mathbf{x} : \frac{P(Y=1|\mathbf{x})}{P(Y=0|\mathbf{x})} = 1\} \equiv \{\mathbf{x} : P(Y = 1|\mathbf{x}) = \frac{1}{2}\}$. In a multi-class classification problem, the decision boundary of the Bayes rule between Class k and Class l is $\{\mathbf{x} : P(Y = k|\mathbf{x}) = P(Y = l|\mathbf{x})\}$. The boundary for unequal costs can be similarly defined.

1.1.2 Classification of High Dimensional Data

With the boost of high technique development in biology fields, such as microarray techniques, a large amount of high-throughput data with "large p small n" property have

been generated. The "large p small n" data often has variables in the magnitude of tens or hundreds of thousands (p), which greatly exceeds the size of training sample size n, i.e. $p \gg n$. Most classical multivariate analysis methods cannot be applied to such high dimensional data due to the fact that there are not enough samples to properly estimate the underlying covariance structure. The key reason for this failure is that the covariance matrix is singular and cannot be inverted. These classical classification methods, including linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logistic regression and etc., can only be applied to data with the sample size greater than the variable dimensionality.

Several solutions have been developed to confront the challenge posed by such high dimensional data. Some dimension reduction techniques, such as principle component analysis (PCA), are often used before applying multivariate analysis methods. PCA can identify a small number of orthogonal principle components, which encapsulate the maximum amount of variation in the original high dimensional data. The number of principle components can be as large as the $n - 1$, but usually the first several components catch the most variation and summarize the most important information in the original data set. Instead of using the original high dimensional variables, one can use a few principle components in the classification methods. But the coefficients estimated from the model are for the principle components, each of which is a linear combination of all original variables. Therefore, the interpretation becomes harder. An alternative way is to select a subset of most relevant variables before fitting the classification model. We call this approach as the "filter" method since variables are filtered using some criteria before the model fitting. Variable selection and model fitting are two separated procedures. The limitation for this approach is that the variable filtering criteria are usually based on the univariate measurement without considering the interaction effects among variables.

In addition to the dimension reduction and variable filtering techniques, a lot of other machine learning approaches have also been proposed for the classification/discriminant analysis. Popular approaches used in classification include KNN, neural network, tree-based methods, and support vector machine etc.

The simplest method in machine learning approaches is KNN. In KNN, an object is classified by a majority vote of its neighbors and the object is assigned to the class most common among its k nearest neighbors. The KNN approach has been widely used in sample classification of microarray and other -omics data (Li et al., 2001, 2004; Pan et al., 2004; Yao and Ruzzo, 2006). In spite of their simplicity, KNN methods show some other

great advantages compared with many other classification methods. KNN methods don't make any assumption about the underlying distribution of the samples. Further, KNN is especially successful when the decision boundaries are irregular, or a class has multiple prototypes, or many classes can not be categorized by a simple model (Yao and Ruzzo, 2006). The reason that KNN works best for class boundaries that are irregular is that KNN is a local method and the classification relationship is only based on the neighborhood data points. This local approximation method doesn't work well if the data points are far away and are not close like a neighbor. Therefore, if the dimensionality is very high, none of objects are actually close to each other in such a high dimensional space. The KNN for very sparse neighborhoods in a high dimensional space will result in high variances.

Artificial neural network (ANN) is a mathematical or computational model based on biological neural networks. The detailed discussion about this algorithm can be found in Rumelhart et al. (1986). Neural network are powerful because of their massively parallel, distributed structure and ability to learn and generalize (Greer and Khan, 2004). The nonlinearity property of a neural network allows it to learn and adapt to nonlinear signals (Greer and Khan, 2004). ANN or ANN-based methods have been widely used in pattern recognition of microarray data or other biological data analyses (Tarca et al., 2005; Linder et al., 2004; Liu et al., 2007; Pal et al., 2007; O'Neill and Song, 2003). However, there are some drawbacks about this algorithm. The training time for neural time is usually longer than other methods. ANN is more like a "black-box" for non-linear statistical data modeling and solutions are encoded in the weights. Therefore, they are not as immediately interpretable as results generated by rules or trees methods (Narayanan et al., 2002).

Tree-based methods can be categorized as single-tree methods or ensemble-tree methods. CART (Classification and Regression Trees)(Breiman et al., 1984) is a popular single-tree method. CART works by recursively splitting the feature space into a set of non-overlapping regions and then labeling each region with the most abundant class in this region. The advantages of CART include (1) it can handle both categorical or numerical variables; (2) it automates stepwise variable selection and is able to handle high dimensional data; (3) it is robust to outliers or misclassification points in the training set; (4) The result are easy to be interpreted. But the major disadvantage of CART is its high variance due to its hierarchical nature of the process. The effect of a small error in the top split can propagate down to all the splits below it (Hastie et al., 2001). Therefore, a small change in the data or in the top split can lead to a very different tree, which makes the interpretation

somewhat cautious. Another limitation is the lack of smoothness of the prediction surface, which may degrade the performance.

To overcome the high variance of a single tree, ensemble-tree methods were proposed. The Random Forest is an ensemble of trees using bagging and bootstrap techniques. The Random Forest method first generates a large number of bootstrap samples using random sampling with replacement. Each bootstrap sample contains randomly selected features as well as samples. A forest is formed from a large number of trees, each of which is built from a bootstrap sample. Trees in the forest may be very different since they are built from different samples and features, so they may have different splitting features or splitting cutting points for the same feature. The bagging technique proposed by Breiman (1996) averages the prediction results from all trees using the majority voting. Bagging reduces the variance of a single tree process, and leads to an improved prediction accuracy. However, bagging technique using the averaging scheme makes the interpretation difficult. Many great advantages make Random Forest a popular method for microarray or other -omics data analysis (Diaz-Uriarte and Alvarez, 2006). First, this tree-based method works well for large p small n data set. Second, it can handle a mixture of categorical and continuous predictors. Third, it usually achieves a good prediction accuracy and is robust to noise features or outlier samples. Fourth, it naturally incorporates the interaction effects due to the tree structure. Fifth, it can handle multi-class classifications as well as the binary-class classifications. Sixth, variable selection is automatically performed using importance measures. Given these promising advantages, Random Forest has been widely used for classification and variable selection in analysis of mass spectrometry data (Izmirlian, 2004; Hettick et al., 2006; Wu et al., 2003), SNP data (Schwender et al., 2004; Lunetta et al., 2004; Bureau et al., 2005), and microarray data (Gunther et al., 2003; Hoffmann et al., 2006). Strobl et al. (2007) further discussed the bias of variable importance measures in situations where potential prediction variables vary in their scales of measurement or their numbers of categories and proposed an improved unbiased variable selection procedure for Random Forest.

Support vector machine (SVM) will be introduced in details in Section 1.2. Briefly speaking, SVM has better theoretical properties than tree-based methods. The classifier built from SVM has a smooth prediction surface. SVM can handle high dimensional data without increasing the learning complexity. In addition, the interpretation of the SVM model is easier than the ensemble tree method.

1.2 Support Vector Machine

The SVM paradigm was originally designed for binary classification problems. It has a nice geometrical interpretation of discriminating one class from the other by a hyperplane with the maximum margin. So, SVM is also known as hyperplane classifier. If the data cannot be separated by a hyperplane in the original input space, SVM performs classification by mapping the input data from original space to a higher dimensional (N-dimensional) feature space and then constructing an N-dimensional hyperplane that optimally separates the data into two categories. I will first give an overview about the standard binary SVM, followed by a discussion of the extension of binary SVM to multi-class SVM. For reference, see Boser et al. (1992), Vapnik (1998), Burges (1998) and Cristianini and Shawe-Taylor (2000).

1.2.1 Binary Support Vector Machine

Linear SVM

Separable Case

We fist introduce the idea of support vector machines in the simplest case, i.e. constructing a linear support vector machine for separable data. Figure 1.1 is a geometric figure for this case. For the separable case, there are infinite separating hyperplanes which can separate the training data into two categories with 100% accuracy. But the optimal hyperplane is the one that separates two classes perfectly and maximizes the distance to the closest points from either class (Vapnik, 1996). By this way, one can identify the unique optimal hyperplane as well as lead to better generalization error on future test data.

Suppose the training data are $\{\mathbf{x}_i, y_i\}, i = 1, \ldots, n, y_i \in \{-1, 1\}, \mathbf{x}_i \in \mathbf{R}^p$ and there is a hyperplane which can completely separate the two classes (Figure 1.2). Then the points on the separating hyperplane (H) satisfy the condition $\{\mathbf{x} : \mathbf{x}^T\beta + \beta_0 = 0\}$, where β is normal to the hyperplane, $|\beta_0|/\|\beta\|$ is the perpendicular distance from the hyperplane to the origin, and $\|\beta\|$ is the Euclidean norm of β (Burges, 1998). After the β and β_0 are rescaled, points on the hyperplane H_1 satisfy $\{\mathbf{x} : \mathbf{x}^T\beta + \beta_0 = -1\}$ and points on the hyperplane H_2 satisfy $\{\mathbf{x} : \mathbf{x}^T\beta + \beta_0 = 1\}$. The perpendicular distance from origin to H_1 is $|1 - \beta_0|/\|\beta\|$ and the perpendicular distance from origin to H_1 is $|-1 - \beta_0|/\|\beta\|$, therefore the margin is simply $2/\|\beta\|$. The goal of SVM learning is to maximize this margin since

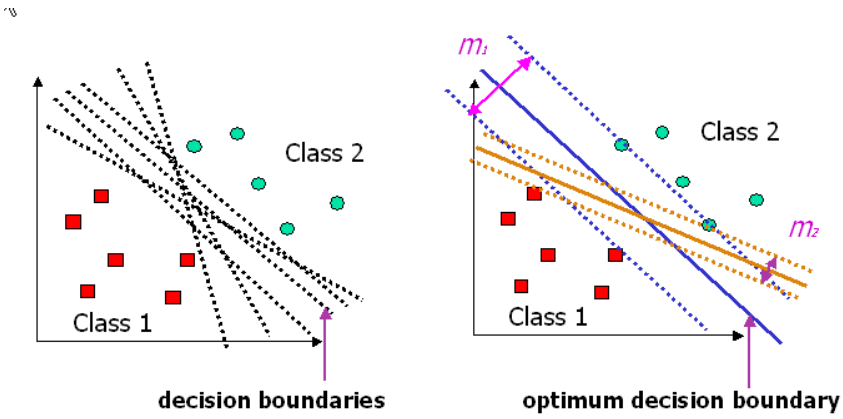


Figure 1.1: Identification of the optimum hyperplane for binary classification in separable case. The left plot shows infinite boundaries can separate two classes perfectly. The right plot shows that the decision boundary with the maximum margin m_1 is the optimum one.

a classifier with a larger margin tends to have better generalization ability on future test data. By this way, SVM is also called large margin classifiers. Actually, maximizing the margin $2/\|\beta\|$ is equivalent to minimizing the $\|\beta\|^2/2$. Besides maximizing the margin, the training data also need satisfy the following constraints:

$$\begin{aligned} \mathbf{x}_i^T\beta + \beta_0 &\geq 1 \quad \text{for} \quad y_i = +1, \\ \mathbf{x}_i^T\beta + \beta_0 &\leq -1 \quad \text{for} \quad y_i = -1. \end{aligned} \tag{1.1}$$

The above two constraints can be combined into one set of inequalities: $y_i(\mathbf{x}_i^T\beta + \beta_0) - 1 \geq 0, \forall i$. In summary, the SVM learning can be formulated as:

$$\begin{aligned} &\min_{\beta_0,\beta} \quad \frac{1}{2}\|\beta\|^2 \\ &\text{subject to} \quad y_i(\mathbf{x}_i^T\beta + \beta_0) - 1 \geq 0, \text{ for } i = 1, \ldots n \end{aligned} \tag{1.2}$$

The problem (1.2) is a convex optimization problem with quadratic objective function and linear inequality constraints. Quadratic programming techniques can be used to solve this problem. The Lagrangian function is obtained by introducing Lagrange multipliers $\alpha_i \geq 0$, for $i = 1, \ldots, n$, one for each of the inequality constraints in (1.2).

$$L_P(\beta, \beta_0, \alpha) = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{n} \alpha_i[y_i(x_i^T\beta + \beta_0) - 1]. \tag{1.3}$$

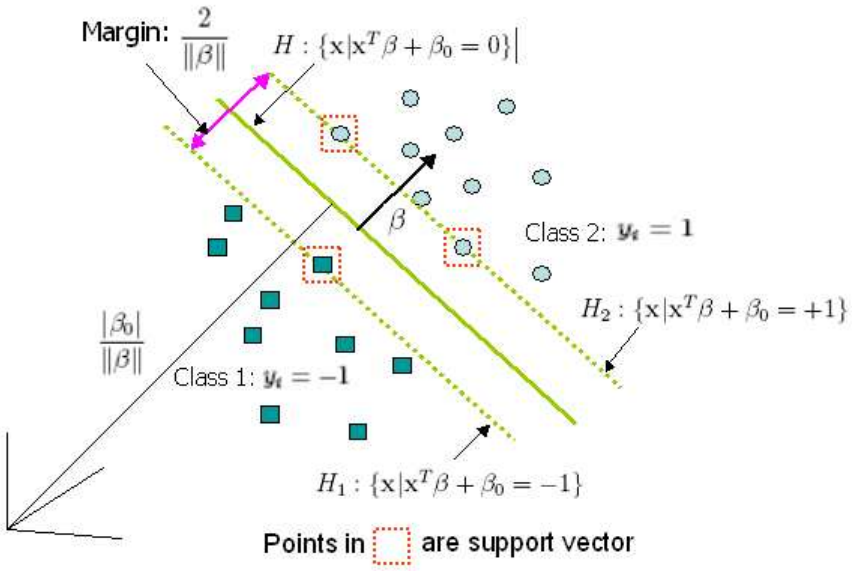


Figure 1.2: Binary SVM for separable case.

L_P has to be minimized with respect to the primal variables (β, β_0) and maximized with respect to the dual variables α_i's. At the saddle points, we have:

$$\frac{\partial}{\partial \beta_0} L(\beta, \beta_0, \alpha) = 0 \implies \sum_{i=1}^{n} \alpha_i y_i = 0, \tag{1.4}$$

$$\frac{\partial}{\partial \beta} L(\beta, \beta_0, \alpha) = 0 \implies \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \beta. \tag{1.5}$$

The above two equality constraints (1.4) and (1.5) can be used to substitute β_0 and β in (1.3), which gives the following dual problem:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{1.6}$$

$$\begin{aligned} \text{subject to:} \quad & \alpha_i \geq 0, i = 1, \ldots, n; \\ & \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned}$$

This particular dual formulation of the problem is also called the Wolf dual (Fletcher, 1987). Minimizing the primal problem L_P and maximizing the dual problem L_D give the same solution of $\hat{\beta}, \hat{\beta}_0$ and $\hat{\alpha}$.

The Karush-Kuhn-Tucker (KKT) conditions play an important role in the constrained optimization since KKT conditions are satisfied for any constrainted optimization problem (convex or not) if the constraints are linear. SVM is a convex problem with linear constraints. For SVM primal problem (1.3), the KKT conditions are:

$$\frac{\partial}{\partial\beta}L_P = \beta - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = 0, \tag{1.7}$$

$$\frac{\partial}{\partial\beta_0}L_P = \sum_{i=1}^{n}\alpha_i y_i = 0, \tag{1.8}$$

$$\alpha_i \geq 0, \text{ for }, i = 1, \ldots, n, \tag{1.9}$$

$$\alpha_i[y_i(\mathbf{x}_i^T\beta + \beta_0) - 1] = 0, \text{ for }, i = 1, \ldots, n, \tag{1.10}$$

$$y_i(x_i^T\beta + \beta_0) \geq 1, \text{ for }, i = 1, \ldots, n, \tag{1.11}$$

For a convex problem, such as SVM, KKT conditions are *necessary* and *sufficient* for $\hat{\beta}$, $\hat{\beta}_0$, $\hat{\alpha}$ to have a solution (Fletcher, 1987). Therefore, solving the SVM problem is equivalent to finding a solution to the KKT conditions.

According to the constraint (1.10), each observation satisfies either $y_i(\mathbf{x}_i^T\beta+\beta_0) - 1 = 0$ or $\alpha_i = 0$. If $y_i(\mathbf{x}_i^T\beta + \beta_0) - 1 \neq 0$, which means the observation is outside of the margin, α_i will be equal to 0. If $y_i(\mathbf{x}_i^T\beta + \beta_0) - 1 = 0$, which means the observation is on the two parallel hyperplanes (H_1 or H_2), the α_i may not be equal to 0. The support vectors (SVs) are all observations with $\alpha_i > 0$, which are located on the hyperplanes. By the constraint (1.7), $\beta = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i$, therefore, β is only defined by support vectors. Thus, the decision function, $\hat{f}(\mathbf{x}) = \hat{\beta}_0 + \sum_{i\in SVs}\hat{\alpha}_i y_i \mathbf{x}_i^T\mathbf{x}$, is only defined by support vectors. But the identification of the SV points requires the use of all the data. We can see the $\hat{\beta}$ can be explicitly expressed as $\sum_{i=1}^{n}\hat{\alpha}_i y_i \mathbf{x}_i$, but $\hat{\beta}_0$ cannot. By using the KKT condition (1.10), $\hat{\beta}_0$ can be solved with any of the support vectors or with an average of all solutions from all support vectors.

Non-separable Case

If the data is non-separable, it is impossible to find a separating hyperplane which satisfies the constraint $y_i(\mathbf{x}_i^T\beta + \beta_0) \geq 1$ for every observation. Therefore, it is necessary to relax the constraint (1.1). The relaxed constraints introduce a positive slack variable $\xi_i \geq 0$

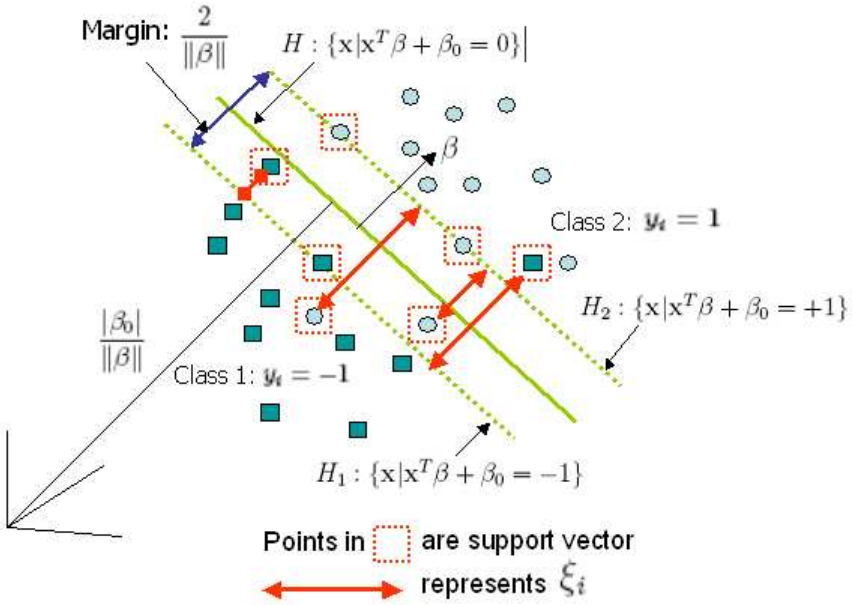


Figure 1.3: Binary SVM for non-separable case.

for each observation i (Figure 1.3):

$$\begin{aligned} \mathbf{x}_i^T\beta + \beta_0 &\geq 1 - \xi_i \quad \text{for} \quad y_i = +1, \\ \mathbf{x}_i^T\beta + \beta_0 &\leq -1 + \xi_i \quad \text{for} \quad y_i = -1. \end{aligned} \tag{1.12}$$

Correspondingly the optimization problem is updated from (1.2) to the following:

$$\begin{aligned} \min_{\beta,\beta,\xi} \quad & \frac{1}{2}\|\beta\|^2 + \gamma(\sum_{i=1}^{n}\xi_i) \\ \text{subject to} \quad & y_i(\mathbf{x}_i^T\beta + \beta_0) \geq 1 - \xi_i, \text{ for } i = 1, \ldots n, \\ & \xi_i \geq 0, \text{ for } i = 1, \ldots n, \end{aligned} \tag{1.13}$$

where $\gamma > 0$ is a penalty to errors.

ξ_i is a measurement about the degree of misclassification. If the point is correctly classified and locates outside of margin or on the H_1 or H_2 hyperplane, $\xi_i = 0$. If the point is correctly classified but locates within the margin, then $1 > \xi_i > 0$. Otherwise, if the point is misclassified, we have $\xi_i \geq 1$. The γ is a tuning parameter that balances the tuning error $\sum_i \xi_i$ and the reciprocal of margin width $\|\beta\|^2$. Large γ puts more penalty on the misclassification where small γ puts more penalty on the margin width. Too large γ will

make the model overfit the training sample, and too small γ will make the model underfit. Usually, cross-validations can be used to identify appropriate γ value.

In non-separable case, Lagrange multipliers $\alpha_i \geq 0$ and $\mu_i \geq 0$ are introduced for each constraint and the Lagrange formulation is:

$$L_P(\beta, \beta_0, \alpha, \xi, \mu) = \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \mu_i \xi_i - \sum_{i=1}^{n} \alpha_i [y_i(x_i^T \beta + \beta_0) - 1 + \xi_i]. \quad (1.14)$$

The optimization problem is to minimize L_p with respect to primal variables β, β_0, ξ and to maximize L_p with respect to dual variables α, μ. At the saddle points, we have

$$\frac{\partial}{\partial \beta_0} L_P = 0 \implies \sum_{i=1}^{n} \alpha_i y_i = 0, \quad (1.15)$$

$$\frac{\partial}{\partial \beta} L_P = 0 \implies \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \beta, \quad (1.16)$$

$$\frac{\partial}{\partial \xi_i} L_P = 0 \implies \alpha_i = \gamma - \mu_i, \text{ for } i = 1, \ldots, n. \quad (1.17)$$

Then substitute all equality constraints (1.15), (1.16) and (1.17) into L_P, we get the Wolfe dual problem:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad (1.18)$$

$$\text{subject to:} \quad 0 \leq \alpha_i \leq \gamma, i = 1, \ldots, n;$$
$$\textstyle\sum_{i=1}^{n} \alpha_i y_i = 0.$$

The KKT conditions for the primal problem in the non-separable case are:

$$\frac{\partial}{\partial\beta}L_P = \beta - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = 0, \tag{1.19}$$

$$\frac{\partial}{\partial\beta_0}L_P = \sum_{i=1}^{n}\alpha_i y_i = 0, \tag{1.20}$$

$$\frac{\partial}{\partial\xi_i}L_P = \alpha_i - \gamma + \mu_i = 0, \text{for } i = 1,\ldots,n, \tag{1.21}$$

$$\alpha_i \geq 0, \text{ for }, i = 1,\ldots,n, \tag{1.22}$$

$$\mu_i \geq 0, \text{ for }, i = 1,\ldots,n, \tag{1.23}$$

$$\alpha_i[y_i(\mathbf{x}_i^T\beta + \beta_0) - 1 + \xi_i] = 0, \text{ for }, i = 1,\ldots,n, \tag{1.24}$$

$$y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \text{ for }, i = 1,\ldots,n, \tag{1.25}$$

$$\mu_i\xi_i = 0, \text{ for }, i = 1,\ldots,n. \tag{1.26}$$

All above conditions uniquely characterize the solution to primal/dual problems.

In the non-separable situation, there are four cases. The first case is $y_i(\beta^T\mathbf{x}_i+\beta_0) > 1$, which means the $\mathbf{x}_i$ point is correctly classified and outside of the margin. In this case, $\xi_i = 0$ and $\alpha_i = 0$ according to constraint (1.10). The second case is $y_i(\beta^T\mathbf{x}_i + \beta_0) = 1$, which means the $\mathbf{x}_i$ point is correctly classified and just located on the parallel margin. In this case, $\xi_i = 0$, but α_i may be zero or may not be zero. If α_i is not zero, then the point $\mathbf{x}_i$ is a support vector and $0 < \alpha_i \leq \gamma$. The third case is $0 < y_i(\beta^T\mathbf{x}_i + \beta_0) < 1$, which means the $\mathbf{x}_i$ point is correctly classified but locates within the margin. In this case, we have $0 < \xi_i < 1$ and $\alpha_i = \gamma$. The fourth case is $y_i(\beta^T\mathbf{x}_i + \beta_0) \leq 0$, which means the $\mathbf{x}_i$ point is incorrectly classified and locates on the separate hyperplane or the other side of separate hyperplane. In this case, $\xi_i \geq 1$ and $\alpha_i = \gamma$. The support vectors are a set of points with $\alpha_i > 0$, which include some points on the parallel hyperplane ($0 < \alpha_i \leq \gamma$), all points within the margin area ($\alpha_i = \gamma$) and all points at the other side of the hyperplane (i.e. misclassified points, $\alpha_i = \gamma$). Similar to the separable case, $\hat{\beta}_0$ can be solved using the KKT condition (1.24) either with any support vector on the two parallel hyperplanes having $\alpha_i > 0, \xi_i = 0$, or with an average of all solutions from all support vectors having $\alpha_i > 0, \xi_i = 0$ for numerical stability.

Nonlinear Support Vector Machine

Sometimes, data cannot be linearly separated in the original input space, that is, we cannot find an optimal hyperplane to separate the data in the original input space. We can map the data from the original input space to higher dimensional "feature space", $\mathcal{F}$. In the higher dimensional "feature space" $\mathcal{F}$, a hyperplane is the optimum boundary to discriminate data. Kernel trick play an important role in transforming the data from input space to feature space without increasing the computational cost. The detailed discussion about nonlinear support vector machine and kernel trick is in Chapter 4.

SVM in Regularization Framework

Wahba (1998) and Evgenios et al. (1999) showed that the SVM paradigm could comfortably fit into the regularization framework where it has one component for data fitting and the other component for controlling the model complexity. Each pair of constraint SVMs, $\xi_i \geq 0$ and $y_i f(\mathbf{x}_i) \geq 1-\xi_i$, can be reformulated into one expression $\xi_i \geq \max\{0,\ 1 - y_i f(\mathbf{x}_i)\} = [1 - y_i f(\mathbf{x}_i)]_+$. Therefore, SVM has the hinge loss function $[1 - yf(\mathbf{x})]_+$. The $yf(\mathbf{x})$ is called the functional margin. If $yf(\mathbf{x}) > 0$, the point is correctly classified; if $yf(\mathbf{x}) < 0$, the point is misclassified. Then, the SVM can be equivalently expressed as:

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} [1 - y_i f(\mathbf{x}_i)]_+ + \lambda \|\beta\|^2 \tag{1.27}$$

where $\lambda = \frac{1}{2\gamma}$

$$[u]_+ = \begin{cases} u & \text{if } u \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

$$f(\mathbf{x}_i) = \begin{cases} \mathbf{x}_i^T \beta + \beta_0 & \text{for linear SVM,} \\ h(\mathbf{x}_i)^T \beta + \beta_0 & \text{for nonlinear SVM.} \end{cases}$$

The above SVM objective function has a regularization formulation format. The term $\sum_{i=1}^{n} [1 - y_i f(\mathbf{x}_i)]_+$ is a loss function to measure the summation of training error, $\|\beta\|^2$ is a penalty to control the model complexity, and λ is used to balance the training error and model complexity. It minimizes an average loss as well as controlling the overfitting by penalizing the model complexity. The regularization methods can not only solve the large p small n high dimensional data but also be used for model selection. Similar for SVM

with kernel K, the regularization problem in a reproducing kernel Hilbert space (RKHS) is $\min_{f\in\mathcal{H}_K}\sum_{i=1}^{n}[1-y_i f(\mathbf{x}_i)]_+ + \lambda\|h\|^2_{\mathcal{H}_K}$. Here $\|h\|^2_{\mathcal{H}_K}$ is regularization penalty component.

Some other example loss functions used in margin-based classifiers are:

- Cross entropy for logistic regression: $L(yf) = \log[1+e^{-yf(\mathbf{x})}]$,
- Exponential loss function for AdaBoost: $L(yf) = e^{-yf(\mathbf{x})}$,
- Misclassification loss: $L(yf) = I(yf<0)$,
- Square error loss: $L(yf) = [1-yf(\mathbf{x})]^2$,
- Hinge loss: $L(yf) = [1-yf(\mathbf{x})]_+$.

All above loss functions have the term yf. They all encourage large positive values of yf by penalizing negative margins more heavily than positive ones. Penalty for some loss function, such as cross entropy, hinge loss etc., increases linearly with increasingly negative margin, while penalty for some other loss functions, such as exponential loss, increases exponentially with increasingly negative margin.

Parameter Tuning

Similar to other regularization methods, the effectiveness of the proposed method depends on the tuned parameters, which control the trade off between the goodness of fit and model complexity (i.e. wiggliness of the fit). There are various tuning methods to find the best λ. If only training data are available, cross-validation is usually used to identify the best λ and approximately estimate the testing error. Either leave-one-out cross validation (LOOCV) or V-fold cross validation can be used. LOOCV is a special case of V-fold cross validation where V is equal to the number of training samples n. In V-fold cross validation, we first randomly split the training sample into V roughly equal parts. For each $v = 1,\ldots,V$, leave the vth part out of the training samples, fit the model using the other $V-1$ parts. Let $f_\lambda^{[-v]}$ be the minimizer of

$$\sum_{i=1, i\notin v}^{n}[1-y_i f(\mathbf{x}_i)]_+ + \lambda\|f\|^2$$

The prediction error of the $f_{\lambda}^{[-v]}$ on the vth part of the data is calculated. For each λ in the predefined grid search, there are V prediction errors. The best λ is the one that gives the minimum average prediction errors.

Relation to the Bayes Error

The success of SVM is justified by Vapnik (1998) based on the upper bounds of its generalization error in terms of Vapnik-Chervonenkis dimensions. Lin (2002) proved that the reason for that SVM outperformed other popular methods, such as logistic regression, is because SVM asymptotically estimate the Bayes rule in a more efficient and direct way.

The data fitting component usually approaches a limiting functional as $n \to \infty$, which can be used to identify the target function. In SVM, the asymptotic target function is $E[1-yf(\mathbf{x})]_+$. Lin (2002) showed that, the minimizer $f(\mathbf{x})$ of $E[1-yf(\mathbf{x})]_+$ is $sign[p(\mathbf{x})-\frac{1}{2}]$, which is the same as the Bayes rule $\phi_B(x)$ for binary classification. However, other popular statistical methods, such as logistic regression, implement the Bayes rule by estimating the probability of $P(Y = 1|X = x)$. Since support vector machine approximates the optimal rule by directly estimating the Bayes rule instead of estimating the $P(Y|X = x)$ first, it is more efficient for some complicated problems.

SVM Application in Bioinformatics

SVM has been successfully applied to the analysis of a lot of real-world data in many areas with demonstrated effective results compared with other techniques or methods. The well-known application areas of SVM are bioinformatics, image recognition and text categorization. The popular application related to image recognition includes handwriting identification (DeCoste and Scholkopf, 2002) and face recognition (Osuna et al., 1997). The text categorization is to classify documents into a predefined categories. Documents usually consist of thousands of words, which means the input data sets usually have extremely high dimensions. It has been shown that SVM outperforms other popular methods in text categorization, such as naive Bayes, Bayes nets and decision trees, in terms of prediction accuracy and computation time (Dumais et al., 1998). I will particularly introduce the application of SVM in bioinformatics areas in the next paragraph.

The SVM algorithm has demonstrated its success in many fields of medical and biology data analysis, such as DNA and protein sequence analysis, microarray gene expres-

sion analysis, mass spectrometry data analysis, translation start site recognition, tissue and sample classification, functional classification of promoter regions and prediction of protein subcellular location and secondary structure, etc. The SVM-Fisher method, which combines HMMs with support vector machines, can detect remote protein homogeneous and show significant improvement on previous methods for the classification of protein domains based on remote homologies (Jaakkola et al., 2000). Vert (2002) introduced a new class of kernels for strings, which were successfully used in the prediction of signal peptide cleavage site from the amino-acid sequence of a protein. In addition to sequence analysis, SVM can be used for gene function prediction as well. Pavlidis et al. (2001) applied SVM learning algorithm to infer gene function classifications from a heterogeneous data set consisting of DNA microarray expression measurements and phylogenetic profiles from whole-genome sequence comparisons. Furthermore, SVM has also been widely used in predicting the protein sub-cellular locations (Hua and Sun, 001a; Bhasin and Raghava, 2004; Garg et al., 2005), protein secondary structure (Hua and Sun, 001b; Pham et al., 2005; Nguyen and Rajapakse, 2005) and protein fold recognition (Ding and Dubchak, 2001; Shamim et al., 2007; Melvin et al., 2007). The development of high-throughput techniques, such as microarray and mass spectrometry, provide large amount of data and information, characterized with extremely large variable dimensions and very small sample size. SVM outperforms other machine learning methods in terms of capability in handling "large p small n" data sets, high prediction accuracy and fast computational speed. The expression levels of genes, proteins and metabolites provide an instantaneous "snapshot" of the status of that cell or tissue. Thus, their profile is widely used for cancer classification and sample prediction. If one searches the PubMed with classification and support vector machine together, around 600 papers come up. This shows that SVM has been successfully and widely used in the classification of cancers and prediction of samples. The details will not be listed here.

1.2.2 Multi-category Support Vector Machine

As described before, the goal of multi-class classification is to choose the best function from $\{\hat{f}(\mathbf{x}, \alpha), \alpha \in \Lambda\} : R^p \to \{1, 2, \ldots, K\}$. Suppose the $p_j = P(Y = j|\mathbf{X} = \mathbf{x})$ is the probability that $\mathbf{x}$ belongs to class j. The Bayes rule of multi-class classification assigns $\mathbf{x}$ to a class with the largest p_j. SVM was originally designed for binary classification problems. There are mainly two strategies in extending SVM to multi-class classification

problems. One is decomposing a multi-class classification problem into multiple binary classification problems and solving a series of binary classification problems. The other is considering all classes simultaneously and directly constructing a multi-class classification decision boundary. These two strategies will be presented in the next two sections.

Reducing Multi-class to Binary Classes

Allwein et al. (2000) presented a unified approach for studying the solution of multi-class categorization problems by decomposing a multi-class problem into multiple binary problems and then solving multiple binary problems using a margin-based binary learning algorithm. The two most popular approaches in decomposition strategy are to decompose a multicategory problem to multiple one-vs-one or one-vs-rest binary classification problems. For example, a three-class classification problem is demonstrated in Figure 1.4. Figure 1.4(a) shows that a three-class classification problem is decomposed into three one-vs-rest binary classification problems. Figure 1.4(b) shows that the same three-class classification problem is decomposed into three one-vs-one binary classification problems. When the testing data is applied, the predict label is a vote from the three binary classifiers. These one-vs-rest and one-vs-one strategies indirectly solve a multi-class classification problem using binary SVMs, which usually achieve a reasonably good prediction accuracy due to the generally good prediction performance of SVM. However, there are a lot of disadvantages for such decomposition strategies. First, Lee et al. (2004) demonstrated the one-versus-rest approach failed if there is no class dominates the union of the others for some $\mathbf{x}$. The Bayes rule for one-vs-rest binary problem is $\phi_B(x) = sign[p_j(\mathbf{x}) - 0.5]$. If a class j satisfies the $p_j(\mathbf{x}) > 0.5$ given $\mathbf{x}$, the algorithm can determine the correct class label. However, if a class j has the $p_j(\mathbf{x}) < 0.5$ given $\mathbf{x}$, it is hard for the algorithm to determine the right class label. Second, the one-versus-rest approach has an unbalanced number of samples in their binary problems if the sample number from one class is much smaller than the union of all other classes. Third, the one-versus-one has a much smaller samples size, which potentially increases the variance of the learned classifier. Fourth, it cannot capture the correlation between different classes since it breaks a multicategory problem into multiple independent binary problems (Crammer and Singer, 2001). For example, tumor subtypes are more correlated to each other than to normal samples. Finally, the generalization error of the learned classifier does not approach the Bayes error when $n \to \infty$.

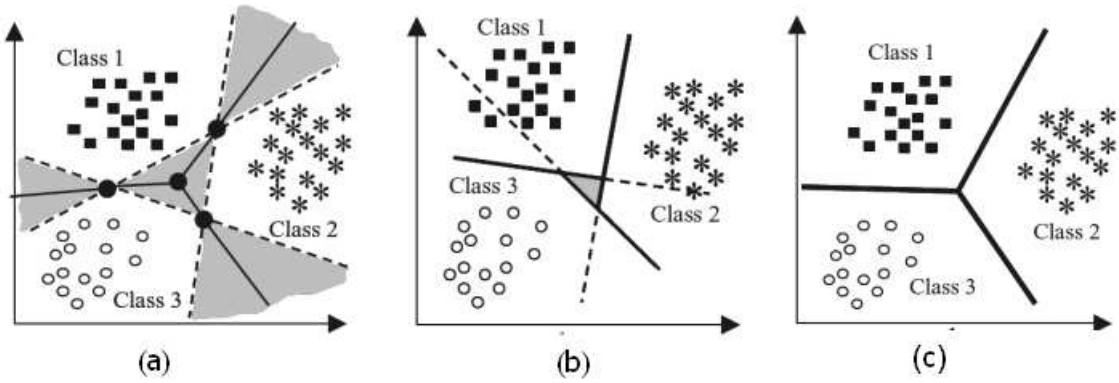


Figure 1.4: Three class classification problem. (a) One-vs-rest approach (b) One-vs-one approach (c) Simultaneous classification approach. This Figure is from Statnikov et al. (2005)

Therefore, a better way for extension of binary SVM to the multicategory case is highly desired, which can classify multiple classes simultaneously (Figure 1.4(c)).

Simultaneous Multiple Classification Needed

The multicategory SVM (Vapnik, 1996) is proposed to treat multiple classes simultaneously. It naturally extends the margin concept from binary SVM to the multi-class case and inherits the optimal properties of the binary SVM. In multi-category SVMs, multiple discriminant functions, $\mathbf{f} = (f_1, f_2, \ldots, f_K)$, need to be estimated instead of one decision function in a binary SVM. Each $f_k(\mathbf{x})$ is associated with a class k. The function $f_k(\mathbf{x})$ represents the strength of a sample$(\mathbf{x}, y)$ belonging to the class k. That is, $f_k(\mathbf{x}) \propto P(Y = k|\mathbf{x})$. Therefore, the classification rule in multi-category SVM is $\Phi(\mathbf{x}) = \arg\max_{k=1,\ldots,K} f_k(\mathbf{x})$ where the classification rule in binary SVM is $\Phi(\mathbf{x}) = sign(f)$. Furthermore, the decision boundary between class k and class l is $\{\mathbf{x} : f_k(\mathbf{x}) = f_l(\mathbf{x})\}$.

The accuracy measured by Generalization Error (GE) is $Error(f) = P(Y \neq \arg\max_{k=1,\ldots,K} f_k(\mathbf{X}))$. Therefore, a reasonable decision vector $\mathbf{f}$ should encourage a large value for $f_y(\mathbf{x})$ and generate small values for $f_l(\mathbf{x}), l \neq y$. A vector $\mathbf{g}$ with $K-1$ elements is defined as $\mathbf{g} = (f_y(\mathbf{x}) - f_1(\mathbf{x}), \ldots, f_y(\mathbf{x}) - f_{y-1}(\mathbf{x}), f_y(\mathbf{x}) - f_{y+1}(\mathbf{x}), \ldots, f_y(\mathbf{x}) - f_K(\mathbf{x}),)$. This $\mathbf{g}$ vector characterizes the correctness and strength of the classification of $\mathbf{x}$ by $\mathbf{f}$. For example, if $\mathbf{g}(\mathbf{f}(\mathbf{x}), y) > \mathbf{0}_{k-1}$, it indicates a correct classification of $(\mathbf{x}, y)$ (Liu and Shen, 2006).

To extend binary SVM to multicategory support vector machine (MSVM), the hinge loss $[1 - y_i f(\mathbf{x}_i)]_+$ in binary SVM needs to be generalized. There are several generalizations of the loss function for MSVM proposed by Weston and Watkins (1999); Crammer and Singer (2001); Lee et al. (2004); **?**.

The loss function proposed by Weston and Watkins (1999) is $L(y, \mathbf{f}(\mathbf{x})) = \sum_{l \neq y}[2 - (f_y(\mathbf{x}) - f_l(\mathbf{x}))]_+$. It gives a penalty/loss when $f_y(\mathbf{x}) < f_l(\mathbf{x}) + 2$ for $l \neq y$. The loss is defined as $\xi_i^l \geq f_l(\mathbf{x}) + 2 - f_y(\mathbf{x})$ for $l \neq y$. The total loss for each observation i is the summation of $\xi_i^l, l = 1, \ldots, K, l \neq y$. Comparing with reducing a multi-class problem to multiple binary problems, they demonstrated that their methods reduced the number of support vectors needed to describe the decision function. However, they found out that their learning algorithm is a little slow since the optimization problem of their method can be large sometimes. Hsu and Lin (2001) compared several methods to solve multi-class problems using SVM in terms of their performance and computational cost and found that the MSVM proposed by Weston and Watkins (1999) only solved the problem once, but the size of the problem is bigger than that of solving a series of binary problems.

The loss function proposed by Lee et al. (2004), $L(y, \mathbf{f}(\mathbf{x})) = \sum_{l \neq y}[f_l(\mathbf{x}) + 1]_+$, gives penalty when $f_l(\mathbf{x}) > -1$ for $l \neq y$. The class output code for a multi-class problem is designed (Lee et al., 2004). They defined v_j for $j = 1, \ldots, K$ as a K-dimensional vector for class j output code. v_j has 1 in the jth coordinate and $\frac{-1}{K-1}$ elsewhere. For their defined output code, a sum-to-zero constraint, $\sum_{j=1}^{K} f_j(\mathbf{x}) = 0$, must be enforced. They further showed that their loss function has good theoretical properties and proved that this extension ensures the solution to directly target the Bayes rule in the same fashion as in the binary case. They also derived a tuning criterion for the MSVM, an approximated leave-one-out cross validation function called Generalized Approximate Cross Validation.

The loss function proposed by Crammer and Singer (2001) is $L(y, \mathbf{f}(\mathbf{x})) = [1 - \min_l\{f_y(\mathbf{x}) - f_l(\mathbf{x})\}]_+$, which means if $\mathbf{x}_i$ belongs to class y, the $f_y(\mathbf{x}_i)$ has to be at least 1 margin bigger than any of other $f_k(\mathbf{x}_i)$, where $k = 1, \ldots, K$ and $k \neq y$. Different from the above two loss functions, this loss function only penalizes the minimum value of all losses associated with each observation i while the above two loss functions discussed penalize the summation of all losses $\xi_i^l, l = 1, \ldots, K, l \neq y$ for each observation i. They also discussed the technical details that yielded significant running time improvement for large data sets. Liu and Shen (2006) proposed an alternative strategy about multi-category learning, called ψ-learning. However, the loss function suggested by Liu and Shen (2006) is actually the

same as the loss function proposed by Crammer and Singer (2001). They further proved that this loss function was robust against extreme observations.

1.3 Variable Selection

Variable selection is a procedure to determine a small subset of variables with strong effects. For example, in regression problem, the mean squared error (MSE) is usually used as the measurement of prediction error. $MSE = Bias^2 + Var$. The least square estimates with full models tend to have low bias and high variance. The bias-variance tradeoff is as the model complexity increases, the variance tends to increase and the squared bias tends to decrease. It is feasible to trade a little bias with the large reduction in variance, thus achieving higher prediction accuracy. Therefore, one possible way of improving prediction accuracy is to shrink some coefficients to zeros. Thus, inclusion of extra variables can be detrimental. The simplest model is preferred that yields acceptable results. In this section, an overview of the classical approaches of variable selection as well as their pros and cons will be addressed followed by a discussion of shrinkage methods.

1.3.1 Classical Methods

There are several classical methods for variable selection. When the variable dimension is not large, the best subset selection can be used. There are 2^p possible subsets of variables. The residual sum-of-squares (RSS) is calculated for every model. The larger number of variables in the subset, the smaller RSS. To identify the optimal number of variables of the subset, the model selection criterion should be used to control bias-variance tradeoff.

The big limitation for the best subset selection method is that the number of subsets increases exponentially with the dimension of variables. It is unfeasible to evaluate all possible models for large p. Thus, seeking a good path through all the possible subsets is desired. So far, several sequential model selection methods have been proposed, including forward selection, backward selection and stepwise selection. Forward selection starts with intercept, sequentially adds the variable that most improves the model fit. The model before adding one variable is called the reduced model and the model after adding one variable is called the full model. An F-statistics is calculated to test the improvement of RSS of full model vs. reduced model after accounting for the sacrifice in the degree of freedom.

Usually there is large improvement in fit, i.e. large F-statistics when the first few variables are added. When more and more variables are added into the model, the improvement of the fit will become smaller and smaller. The algorithm stops when the improvement of the fit, i.e. F-statistics, cannot pass a predefined threshold for adding a variable, such as 0.10. The backward elimination starts with the full model with all variables in the model, then sequentially eliminates the variable with the smallest F-statistics, and stops when F-statistics cannot pass a predefined threshold for dropping a variable. The stepwise selection considers both forward and backward moves at each step and uses the thresholds to determine if the variable needs to be added or dropped or the move should be stopped. Note that the threshold for adding or dropping may be different.

Sequential model selection is usually used in the data set with the dimension smaller than the sample size. If the dimension is larger than the sample size, it will be ill-conditioned to fit all variables into the model. Therefore, in high dimensional data, researchers usually first rank the variables according to some univariate measurements, which measure the correlation of each variable with the outcome. Then the subset of variables with higher-ranked measurements will be used for model fitting, but the number of important variables in the subset is arbitrarily determined or determined by using the evaluation of the prediction accuracy.

All the above approaches for variable selection can improve the model interpretability due to the smaller number of variables and sometimes may achieve better prediction accuracy than the full model. However, there are a lot of limitations for the above variable selection approaches. First, the sequential model selection may end up with a local optimum model (not a best global sub-model) since variables are sequentially added/dropped and the evaluation is based on the selected model in the previous step. Second, the sequential selection is a discrete process, i.e. a variable is either retained in or discarded from the model. Thus, it is possible to have high variance. Third, the ranking method doesn't work well if variables are highly correlated. The role of variables should be measured relative to other variables in the model and the prediction accuracy is determined by the variables working together. However ranking methods measure the importance of each variable independently and doesn't consider the correlation or interaction among variables. To overcome the limitation of classical methods for variable selection, the shrinkage methods for variable selection have been proposed in literature (Hoerl and Kennard, 1970). The shrinkage methods can select variables in a continuous fashion without suffering from high variability

of sequential variable selection. In addition, the shrinkage methods consider all variables at the same time, which may lead to a better global submodel.

1.3.2 Shrinkage Methods

The shrinkage methods were first proposed in regression problems. The method of ridge regression was first proposed by Hoerl and Kennard (1970) as an alternative to least squares estimation of the coefficients in a linear model. The ridge regression shrinks the regression coefficients by imposing a penalty on their size as the following (Hastie et al., 2001):

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta_0,\beta} \Big\{ \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \Big\} \tag{1.28}$$

Here, λ is used to control the amount of shrinkage. The ridge penalty, $\sum_{j=1}^{p} \beta_j^2$, can shrink the variables toward to zeros. The larger the λ is, the bigger shrinkage is applied.

The ridge regression is useful in cases where the explanatory variables are non-orthogonal or highly correlated. In an ill-conditioned problem, adding an additional penalty term can solve the matrix inversion problem. When many variables are highly correlated, a large positive coefficient in one variable can be canceled by another negatively correlated variable. Therefore, the coefficients of variables can be poorly determined and have a large variance. The ridge penalty controls such situations by applying a constraint on the size of the coefficients. Ridge regression is equivalent to the weight decay in a neural network. Weight decay adds the same penalty term to the error function, which can help prune unimportant network connections. Ridge regression can shrink the coefficients towards zeros. However, it is hard to shrink the coefficients to exactly zeros. That means the learned model will retain all the variables. Another shrinkage method, called LASSO, can shrink some variables to exactly zero, which results in a more sparse and interpretable model.

LASSO is a short term for "Least Absolute Shrinkage and Selection Operator", which was proposed by Tibshirani (1996) for variable selection in linear regression. LASSO is defined similar to ridge regression, but replacing the L_2 ridge penalty $\sum_{j=1}^{p} \beta_j^2$ by the L_1

LASSO penalty $\sum_{j=1}^{p} |\beta_j|$.

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta_0,\beta} \Big\{ \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \Big\} \tag{1.29}$$

This LASSO penalty tends to shrink some coefficients to exactly zeros. Thus, it gives a better interpretable model with less variables. It also inherits the character of stability of ridge regression. In summary, LASSO combines the pros of model interpretability from subset selection and solution stability from ridge regression.

Why can LASSO produce some coefficients to exactly zeros but ridge penalty cannot? The ridge regression (1.28) can also be written as:

$$\begin{aligned} \hat{\beta}^{\text{ridge}} = \arg\min_{\beta_0,\beta} \quad & \textstyle\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \\ \text{subject to} \quad & \textstyle\sum_{j=1}^{p} \beta_j^2 \le t^2. \end{aligned}$$

The LASSO regression (1.29) can also be written as:

$$\begin{aligned} \hat{\beta}^{\text{lasso}} = \arg\min_{\beta_0,\beta} \quad & \textstyle\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \\ \text{subject to} \quad & \textstyle\sum_{j=1}^{p} |\beta_j| \le t. \end{aligned}$$

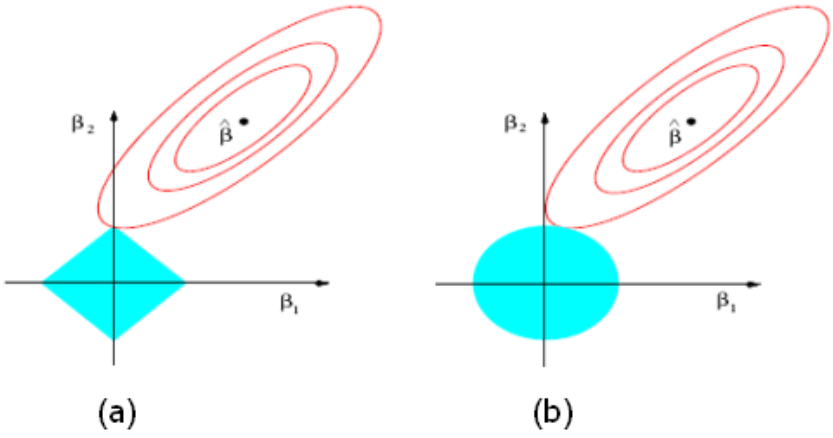


Figure 1.5: Estimation picture for the lasso (a) and ridge regression (b). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \le t(a)$ and $\beta_1^2 + \beta_2^2 \le t^2(b)$, respectively, while the red ellipses are the contours of the least squares error function. This figure is from Hastie et al. (2001).

The estimation of $\hat{\beta}^{\text{ridge}}$ and $\hat{\beta}^{\text{lasso}}$ can be illustrated in Figure 1.5 (Hastie et al., 2001). This figure is a special case of linear regression in two dimensions, where $\beta = (\beta_1, \beta_2)$. The training error contour in the linear regression is always an ellipse, which is colored in red in Figure 1.5. In ridge regression, the penalty contour is a circle, which is colored in solid blue. The point on the penalty contour that minimizes the training error lies where the two contours are tangential. Usually, at that point none of the weights are zero. However, in lasso regression, the penalty contour is a diamond. Because the vertices of the diamond tend to "stick out", they are usually the spot on the penalty contour where the smallest training error contour is attained. At those vertices, one of the β_1, β_2 is pushed to zero.

1.3.3 Variable Selection Methods in SVM

The general SVM has a different loss function from the ridge regression, but the same penalty function as the ridge regression, ridge penalty (i.e. L_2-norm). For linear SVM, the important variables have bigger coefficients while the unimportant variables have smaller coefficients due to the ridge penalty. However, the SVM utilizes all the variables in its model and cannot eliminate unimportant or redundant variables from the model. Furthermore, including a lot of redundant or noise variables can diminish the prediction accuracy of the model.

Several methods have been proposed for variable selection in SVM. Guyon et al. (2002) developed an automatic gene selection method, called recursive feature elimination (RFE), for classification using binary support vector machine. Instead of ranking genes using correlation between gene and phenotype, they ranked genes using the coefficient magnitude trained from the SVM; recursively eliminated the variables with the smallest coefficient magnitude in the learned SVM model followed by recursively training the updated data with decreasing number of variables to re-rank the rest of genes. They demonstrated that the RFE-SVM could eliminates gene redundancy automatically and yielded better and more compact gene subsets (Guyon et al., 2002). The RFE-SVM needs to train multiple SVM classifiers with subsets of features of decreasing size. Thus, the training time linearly increases with the number of genes for elimination, i.e. number of classifiers to be trained. For microarray data with extreme large number of genes/features, the computation cost is very expensive.

The subset selection method sets the coefficients of unimportant variables to zeros

and keeps the coefficients of important variables intact. However, the subset selection method is not continuous, therefore the result is unstable and has high variance. Bradley and Mangasarian (1998) suggested that the L_1-norm SVM can perform the feature selection in the binary SVM. The L_1-norm is defined as $\|\beta\|_1 = \sum_{i=1}^{p} |\beta_i|$, which is same as the LASSO penalty. The L_1-norm selects variables in a continuous fashion, which sets the small coefficients to zero and shrink the large coefficients by a constant. Therefore, the result from L_1-norm selection is stable due to the continuous fashion but will cause bias if the threshold parameter is large. Fan and Li (2001) proposed a new penalty form, called smoothly clipped absolute deviation (SCAD) penalty, for regression problems, which sets the small coefficients to zeros. It provides nearly unbiased estimates for large coefficients and gives a model continuous in data. SCAD penalty combines the pros from both L_1-norm and subset selection. In addition, Zhang et al. (2006) further extended this new form of the penalty to the binary SVM classification, called SCAD-SVM, and applied the SCAD-SVM to high-dimensional biological data.

All the above variable selection methods were developed using binary SVM. Very few methods were developed to select variables based on multi-category SVM. To date, there are only three publications discussed about variable selection in MSVM (Wang and Shen, 2007; Lee et al., 2006; Zhang et al., 2006). Zhu et al. (2003) suggested an algorithm to compute the whole solution path of L_1-norm binary SVM over a range of tuning parameters. Wang and Shen (2007) further computed the whole solution path of L_1-norm for multi-category SVM. The L_1-norm multi-class SVM developed by them can perform multi-class classification and variable selection simultaneously through an L_1-norm penalized sparse representation. Zhang et al. (2006) proposed a new penalty form, supnorm penalty, which is a good fit for multi-class classification. The supnorm MSVM penalizes the maximum value of K coefficients, which are associated with the same variable. Thus, if the maximum value of K coefficients is equal to zero, all K coefficients should be equal to zero. Therefore, the supnorm MSVM leads to a more sparse model than L_1-norm MSVM. Lee et al. (2006) proposed a nonlinear component selection in MSVM. Lee et al. (2006) applied ANOVA decomposition on the kernel for MSVM variable selection and added an additional penalty on the sum of weights of functional subspaces, which leads to a sparse solution.

1.4 Motivation and Organization of This Dissertation

As discussed above, there is not much work on variable selection in MSVM. All exising methods of variable selection for MSVM are based on the loss function proposed by Lee et al. (2004). As discussed in the MSVM case, the loss function proposed by Crammer and Singer (2001) naturally extends the concept of margins from binary SVM to MSVM. In this dissertation, the author developed several new approaches for variable selection in MSVMs. Several variable selection regularization forms using the loss function proposed by Crammer and Singer (2001) were investigated and ANOVA decomposition of kernel for non-linear MSVMs were developed. The rest of this dissertation is organized as the follows. In Chapter 2, a real biological problem about predicting a multi-category liver cell necrosis severity using the microarray data is presented. This is a multi-class classification problem with variable selection. Two existing methods, Random Forest and GEMS-SVM (Gene Expression Model Selector using Support Vector Machine), were applied on this real data and interesting genes related to the mechanism of liver cell necrosis were identified. The Random Forest method is a tree-based classification method which iteratively eliminating less important gene by sequentially fitting the Random Forest classifiers in multiple times. The GEMS-SVM uses the "filter" method to select genes. It ranks genes using the signal-to-noise ratio followed by a multi-class SVM by Weston and Watkins (1999). In Chapter 3, the general MSVM proposed by Crammer and Singer (2001) was re-formulated with L_1 penalty and supnorm penalty. An adaptive weighted L_1 penalty and adaptive weighted supnorm penalty were also investigated. The simulation and real data results showed that the newly developed methods have better finite sample performance in terms of both variable selection and classification accuracy in finite samples than existing methods. In Chapter 4, the non-linear classification problems were further tackled. Two approaches are developed for non-linear MSVMs via basis function transformation and non-linear MSVMs via kernel function respectively. The linear MSVMs developed in Chapter 3 can be directly extended to handle the variable selection of non-linear MSVMs via basis function transformation. For non-linear MSVMs via kernel function, a new MSVM method was developed which can perform ANOVA decomposition on the kernel, construct one kernel for each dimension, learn a scaling factor for each dimension and shrink some scaling factors of non-important dimensions to zeros. This new MSVM classifier can perform the variable selection on non-linear MSVM with different kernels, which is more general and flexible than the non-linear MSVM

with basis function transformation. Comparing with the two existing methods, Random Forest and GEMS-SVM used in Chapter 2, the new methods presented in this dissertation can perform variable selection and classification simultaneously and the important variables can be selected in one run instead of sequentially running multiple classifiers. Comparing with all existing methods of variable selection for MSVMs, the new methods in this dissertation can select the smaller number of genes and achieve a high prediction accuracy at the same time. Chapter 5 summarizes the dissertation and concludes with a discussion.

Chapter 2

Prediction of Liver Necrosis in Rats Exposed to a Compendium of Hepatotoxicants

2.1 Abstract

Liver injury caused by toxic insults is traditionally evaluated based on the analysis of serum enzymes such as ALT and AST. Discerning between the various types of liver injury using these biomarkers is an on-going investigation. Necrosis of the liver from exposure to toxicants is a type of liver injury characteristic of a degenerative process leading to death of hepatocytes. Some of the biochemical events that lead to necrosis of the liver are conjugation of reactive metabolites to proteins and unsaturated lipids, alteration of cellular Ca++ homeostasis, inhibition of protein synthesis and shifts in Na+ and K+ balance. However, the pathogenesis of necrosis of the liver from exposure to hepatotoxicants is a complex biological response to the injury. It is hypothesized that gene expression profiles can serve as a signature to predict, with a high degree of accuracy, the level of necrosis (as a class label for the phenotype) elicited by acute exposure of rats to a variety of hepatotoxicants. It is postulated that the expression profiles of the predictor genes in the signature can provide insight to some of the biological processes and molecular pathways that may manifest necrosis of the rat liver.

Male Fischer rats were treated with seven known hepatotoxicants (1,2-dichloroben-

zene, bromobenzene, monocrotaline, N-nitrosomorpholine, thioacetamide, galactosamine and diquat) and one control-isomer (1,4-dichlorobenzene), at low, mid or high doses, sacrificed at 6, 24 or 48 hours and analyzed for gene expression by microarray. Liver samples were grouped by the level of necrosis exhibited in the tissue. Analysis of significantly differentially expressed genes between adjacent necrosis levels revealed that inflammation follows programmed cell death in response to the agents. Using a Random Forest classifier with gene selection, the expression profiles of 21 informative genes were identified as a signature which achieved 90%, 80% and 60% prediction accuracies of necrosis against independent test data derived from the livers of rats exposed to acetaminophen, carbon tetrachloride, and allyl alcohol respectively. Pathway and gene network analysis of the genes in the signature revealed several gene interactions suggestive of apoptosis as a process possibly involved in the manifestation of necrosis of the liver from exposure of the hepatotoxicants in rats. TNF-α's cytotoxic effect and also a central regulatory role of JUN, TP53 and apoptosis-related genes are possible processes leading to necrosis.

The data analysis, gene selection and prediction approaches permitted a grouping of the classes of rat liver samples exhibiting necrosis to improve the accuracy of predicting the level of necrosis as a phenotypic end-point observed from the exposure. The strategy, along with pathway analysis and gene network reconstruction, led to the identification of 1) expression profiles of genes as a signature of necrosis and 2) perturbed regulatory processes that exhibited biological relevance to the manifestation of necrosis from exposure of rat livers to a compendium of hepatotoxicants.

2.2 Introduction

Hepatotoxicity is one of the most commonly observed adverse effects in response to many environmental and toxic exposures and is of major concern in the drug development industry (Kaplowitz and DeLeve, 2007). The liver's response to insults depends on the properties of the stressor, the dose received and if the exposure is acute or chronic. Examples of injury or damage are fatty liver, necrosis, cholestasis, cirrhosis or cancer. Traditionally, the detection of a stressor's toxicity relies on the evaluation of serum enzyme levels that are indicators of tissue damage (Casarett et al., 2001). For instance, elevations of ALT and AST are indicative of liver damage (Hamadeh and Afshari, 2004). They are associated with inflammation and/or injury to hepatocytes. Necrosis of the liver usually results in

hepatocellular plasma membrane leakage of AST and ALT into the bloodstream. However, although the levels of these serum enzymes are hallmarks of hepatocellular damage, they do not discern the severity of the liver injury. The ability to predict necrosis at the molecular level, the extent (level) of damage and the source of the insult is currently a challenge using classical toxicological assays, parameters and biomarkers.

Microarray analysis has evolved as a reliable technology to survey the expression of genes across an entire genome (Schena et al., 1995). Several efforts have shown that gene expression signatures can be "anchored" to the phenotype of biological samples (Bushel et al., 2007) and even characterize the genetic variability in individuals (Cheung and Spielman, 2002; Morley et al., 2004). The possibilities of phenotypic anchoring are promising and just beginning to take form in investigations from toxicogenomics and risk assessment to pharmacogenomics and personalized medicine. Ultimately, the success of these efforts relies on the identification of genes and gene products that are considered biomarkers of toxicity or candidates of susceptibility to health conditions. Recently, Bushel and Heinloth et al. (2007) have shown that rat blood gene expression signatures are predictive of the toxic exposure of acetaminophen and can be used to monitor the exposure of the toxicant reflected in the human blood and in the rat liver. The study was based on a single hepatotoxicant which targeted a specific region of the rat liver (centrilobular), addressed discrimination of non

sub-toxic vs toxic (two class) and overdose exposure levels and demonstrated the superiority of gene expression markers over traditional clinical parameters in predicting the exposure. A more comprehensive analysis of compound-induced liver injury was performed *ab initio* using samples exposed to hepatotoxicants or compounds without known liver toxicity (Dai et al., 2006). Expression profiles from 212 genes combined with a composite hepatotoxicity score were highly predictive of compound-induce liver injury. It would be useful to be able to identify gene expression patterns as a diagnostic signature for prediction of the (multi-class) level of necrosis as the general phenotypic response that is commonly manifested from toxic exposure to a compendium of stressors which targets various regions of the liver.

In this study, gene expression data from rat livers exposed to a compendium of hepatotoxicants (Lobenhofer et al. in press) was used to identify gene expression patterns as a diagnostic signature which predicts the level of necrosis of the liver with a high degree of accuracy. The eight chemical compounds in the compendium (1,2-dichlorobenzene, 1,4-dichlorobenzene, bromobenzene, monocrotaline, N-nitrosomorpholine, thioacetamide,

galactosamine and diquat) elicit some or no hepatotoxicity in male rat liver samples at one or more of the three time points and low, medium or high dose point exposures per compound. A form of necrosis was the major lesion observed in a region of the liver from the toxic doses of the chemicals. Preliminary analysis of the liver gene expression data using an SVM classifier within each dose/time group revealed compound-specific separation of the samples exposed to the hepatotoxicants and that SVM classifiers derived from the blood data performed better in higher dose groups at the later time points as compared to the liver data (Lobenhofer et al. in press). However, the blood data was not able to predict animals in some cases where the hepatotoxicant elicited a different phenotypic response with the animals of a particular dose/time group. To investigate the ability of liver genomic markers to predict the level of necrosis manifested in the livers of the animals exposed to the hepatotoxicants, a Random Forest classifier was utilized with an out-of-bag classification error and variable importance estimation procedure to select gene predictors of three classes of the level of necrosis that were derived according to 1) the five severity scores of the injury, 2) the differentially expressed genes from an ANOVA model and 3) the Gene Ontology biological processes enrichment shared by adjacent necrosis levels. From this strategy, gene expression profiles from 21 informative genes were identified as a diagnostic signature which achieved 90%, 80% and 60% prediction accuracies of the level of necrosis against independent test data derived from rats exposed to acetaminophen, carbon tetrachloride, and allyl alcohol respectively. In addition, it was determined that inflammation follows programmed cell death in response to the hepatotoxicants and TNF-α's cytotoxic effect as well as a central regulatory role of JUN, TP53 and apoptosis-related genes are possible processes leading to necrosis.

2.3 Materials and Methods

2.3.1 Experimental Design

The training data set is comprised of studies from the exposure of rats to eight compounds, including 1,2-dichlorobenzene, 1,4-dichlorobenzene, bromobenzene, monocrotaline, N-nitrosomorpholine, thioacetamide, galactosamine and diquat. All eight compounds were studied using standardized procedures, i.e. a common array platform, experimental procedures and data retrieving and analysis processes (Lobenhofer et al., 2006). For each

compound, four to six male, 12 week old F344 rats were exposed to a low dose, mid dose(s) and a high dose of the toxicant and sacrificed at 6, 24 and 48 hr later (Table 2.1). At necropsy, liver and blood were harvested for RNA extraction, histopathology, clinical chemistry and hematology assessments. For liver RNA, left liver lobes were flash frozen, pulverized, and RNA was extracted from a portion of the powder with the QIAGEN RNeasy Maxi Kits (QIAGEN, Valencia, CA). The test data is comprised of three compound data sets, acetaminophen, carbon tetrachloride, and ally alcohol. Acetaminophen data sets were collected from three different independent studies (NCT008, NTP and NCT informatics challenges). Studies used in the test data are not from the standardized procedures. For both the training and test samples, a time-matched vehicle control pool was made for each compound and each tissue by pooling equal amounts of RNA from each of the four control animals. Each treated animal was hybridized against a time matched control pool to the Agilent Rat Oligonucleotide Microarray (Agilent #G4130A) with a dye-swap technical replicate. Fluorescence intensities were measured with an Agilent DNA Microarray Scanner (Agilent g2565AA) and processed with the Agilent G2565AA Feature Extraction software.

2.3.2 Normalization of the Microarray Data

The log10-ratio intensity value for each gene feature on the array was retrieved from the raw file of each array. Each array was normalized by subtracting the sample-median value. Then the dye-swap arrays from the same biological replicate were merged by averaging. After dye-swap merging, there were a total of 318 arrays, one for each treated animal.

2.3.3 Histopathology

From the left liver lobes, two sections were taken and fixed in 10% formalin. After dehydration with ethanol, the liver sections were embedded in paraffin and H&E stained slides were made. These slides were evaluated by two independent pathologists and disagreements were resolved by a pathology working group review (Boorman et al., 2002). Hepatocyte necrosis was one of the observed lesions. The severities of necrosis were graded into 5 levels (none, minimum, mild, moderate and marked) by pathologists according to the percentage of hepatocytes that show necrosis (Table 2.2). The necrosis observation severity levels were then used as a class label for the samples in the training and test data sets.

Table 2.1: Experimental design of the training and test samples

Data Set	Compound	Sample Size	Time (hr)	Dose(mg/kg body weight) Low	Mid	High	Observed Hepatotoxicity
Training	1,2-dichlorobenzene	34	6/24/48	15	150	1500	centrilobular necrosis
	1,4-dichlorobenzene	36	6/24/48	15	150	1500	centrilobular necrosis
	bromobenzene	36	6/24/48	25	75	250	centrilobular necrosis
	diquat	72	6/24/48	5	10/20	25	centrilobular, midzonal and focal necrosis
	galactosamine	36	6/24/48	25	100	400	multifocal necrosis
	monocrotaline	32	6/24/48	10	50	300	centrilobular and midzonal necrosis
	N-nitrosomorpholine	36	6/24/48	10	50	300	centrilobular necrosis
	thioacetamide	36	6/24/48	15	50	150	centrilobular necrosis
Test	acetaminophen (NCT 008)	36	6/24/48	50	150	1500/2000	centrilobular necrosis
	acetaminophen (NTP)	64	6/18/24/48	50	150	1500/2000	centrilobular necrosis
	acetaminophen (informatics challenge)	108	3/6/12/24		150	1500/2500	centrilobular necrosis
	carbon tetrachloride	72	3/6/24/72	15	750	2000	centrilobular necrosis
	Ally Alcohol	95	6/24/48/72	10	20	40/50	periportal necrosis

Table 2.2: Necrosis severity and distribution in each compound study of the training data sets

Necrosis Observation (% of hepatocytes showing necrosis)	No Sign	<5%	5%-25%	26%-50%	>50%	Sample Size
Necrosis Level	0	1	2	3	4	
1,2-dichlorobenzene	17	8	5	2	2	34
1,4-dichlorobenzene	31	4	1	0	0	36
bromobenzene	16	7	5	0	8	36
diquat	50	10	6	4	2	72
galactosamine	18	7	8	2	1	36
monocrotaline	16	11	1	0	4	32
N-nitrosomorpholine	12	17	2	1	4	36
thioacetamide	4	18	1	6	7	36
Total sample size	164	82	29	15	28	318

2.3.4 Clinical Chemistry

At sacrifice, blood was collected into serum separation tubes (BD Microtainer Tubes, BD, Franklin Lakes, NJ) and serum was separated. Clinical chemistry analyses (albumin, cholesterol, creatinine, direct bilirubin, total bilirubin, total bile acid concentrations, triglycerides, and activities of alanine aminotransferase [ALT], alkaline phosphatase, aspartate aminotransferase [AST], lactate dehydrogenase [LDH] and sorbitol dehydrogenase [SDH]) were performed on all rats at study termination. Serum levels of the established liver injury markers ALT and AST increase when the liver shows inflammation or hepatotoxicity.

2.3.5 One-way ANOVA

To identify the genes that are significantly differentially expressed among the different levels of necrosis, an unbalanced one-way ANOVA was fitted for each gene,

$$
\begin{aligned}
Y_{ij} &= \mu + N_i + \varepsilon_{ij}, \quad \text{where} \\
i &= 0, \ldots, m \text{ (level of necrosis)} \\
j &= 1, \ldots, n_i, n_i \text{ is the number of rats in necrosis level } i \\
Y_{ij} &= \text{the gene expression value at the } j^{th} \text{ rat of } i^{th} \text{ necrosis level} \\
\mu &= \text{the average expression value of the gene for all rats from all necrosis levels} \\
N_i &= \text{the necrosis effect of the gene at level } i \\
\varepsilon_{ij} &= \text{the expression deviation of the gene at } j^{th} \text{ rat from the necrosis effect of level } i
\end{aligned}
$$

The number of rats in the different necrosis levels is different, leading to an unbalanced design with different numbers of replicates in each level of the necrosis factor. The significantly differentially expressed genes between two adjacent necrosis levels were identified by estimation statements. Bonferroni multi-test corrections are applied.

2.3.6 Biological Processes Analyses

The gene symbols for all genes features on the Agilent chip were retrieved from the S.O.U.R.C.E. database (http://smd.stanford.edu/cgi-bin/source/sourceBatchSearch) and used as the input of High-Throughput GoMiner (Zeeberg et al., 2005). GoMiner is used to test if selected genes in a gene list are over-represented for a particular biological process. Here, the total genes are all genes from the chip and the selected genes are the ones identified as significantly differentially expressed. The over-expressed genes and under-expressed genes of the selected differentially expressed genes were tested separately. For each gene list, every biological process was assessed for significance based on the Fisher's exact test and a p-value assigned based on the hypergeometric probability distribution. All p-values from each combination of a biological process and gene list were filtered using a false discovery rate (FDR) at 0.05, clustered and visualized using JAVA Treeview (Saldanha, 2004).

2.3.7 Gene Selection and Classification Methods

Random Forest: The Random Forest classification approach grows many single classification trees and chooses the most popular vote over all trees in the forest (Diaz-

Uriarte and Alvarez, 2006). Each tree uses the randomly selected samples (with replacement) as the training set. About one-third of the cases are left out of the selected samples, which is called the out-of-bag (OOB) data. The OOB data is used as the testing data to get an unbiased estimate of the classification error and to estimate variable importance. Two R packages, randomForest and varSelRf were used in the Random Forest classification.

GEMS: Gene Expression Model Selector provides several SVM-based binary or multi-category classification methods and several gene selection methods (Statnikov et al., 2005). The software constructs and estimates models from all combinations of gene selection methods and classification methods, then reports the models with the minimum cross-validation error. This is a filtering approach to rank genes by the selection method and then include them (step by step from the top ranked one to the bottom ranked one) into the model. Ten-fold nested cross-validation was used to provide an unbiased estimation of the model performance. A linear polynomial kernel was selected for the SVM.

2.3.8 Gene Network Reconstruction

Genes identified as significantly differentially expressed and then selected as a predictor from the Random Forest classifier were used in the Ingenuity Pathway Analysis (IPA) software (Ingenuity Systems, Redwood City, CA) to identify biological pathways that the genes are a component of. All genes within the network space from the pathway analysis were collected, their Agilent probes mapped and expression profiles retrieved. Thus, a total of 32 gene profiles were used for gene network reconstruction. Bayesian Network Inference with Java Objects (BANJO) developed by Dr. Alexander Hartemink (http://www.cs.duke.edu/ amink/software/banjo/)(Yu et al., 2002) was used to build Bayesian networks from the data. The 32 gene profiles across 318 arrays were discretized into three levels using the prior frequencies of each class. Only 84 samples from the high dose treatments of the hepatotoxicants (except for 1,4-dichlorobenzene) were used for the network reconstruction. In each run a heuristic algorithm based on simulated annealing searches for the network/model with the highest posterior probability of being generated from the data. The algorithm was run 500 times and the highest scoring network from each was gathered. The probability of the edges being presented was computed using weighted average scoring from all the models (Hartemink et al., 2002). If an edge is presented in all 500 runs, the probability of it is exactly 1.

Table 2.3: Number of DEGs between two adjacent necrosis levels after Bonferroni multi-test correction

Bonferroni p-value threshold	level 1 vs. 0	level 2 vs. 1	level 3 vs. 2	level 4 vs.3
0.0001	1592	14	336	31
0.001	1960	33	490	50
0.01	2404	90	695	91
0.05	2683	171	910	131

Total 20,500 genes from Agilent platform are tested.

2.4 Result

2.4.1 Manifestation of Necrosis

A data set containing 8 hepatotoxicants was used to identify genes related to necrosis level. Six out of eight compounds, including 1,2-dichlorobenzene, 1,4-dichloroben-zene, bromobenzene, Monocrotaline, N-nitrosomorpholine and thioacetamide, cause centrilobular hepatocyte necrosis, while galactosamine causes multifocal hepatocyte necrosis and diquat causes centrilobular, midzonal and focal hepatocyte necrosis (Table 2.1). Table 2.2 shows the animals' necrosis severity distribution in each compound studied. The necrosis severity included five levels: none, minimum, mild, moderate and marked. None represents no signs of necrosis, minimal represents less than 5% of hepatocytes show necrosis, mild represents 5%-25% of hepatocytes are necrotic, moderate means 26%-50% hepatocytes show necrosis and marked means more than 50% of liver cells are necrotic. A total of 318 animals were exposed, 164 of them showing no signs of necrosis, 82 with minimum necrosis, 29 expressed mild necrosis, 15 with moderate necrosis, and 28 showing marked necrosis.

2.4.2 Gene Expression Changes Transition with Severity of Necrosis

Given the groups of samples according to the manifestation of hepatocyte necrosis, genes was extracted from the microarray data that have expression levels that are significantly different between severity levels. The severity levels of necrosis were transformed to indicator variables 0, 1, 2, 3 and 4 denoting none, minimal, mild, moderate and marked necrosis respectively. An unbalanced one-way ANOVA was constructed with the level of necrosis as the only factor. To find significantly differentially expressed genes between two adjacent necrosis levels, four estimations of the comparison of necrosis score 0 vs. 1, 1 vs. 2, 2 vs. 3 and 3 vs. 4 were performed. Table 2.3 summarizes the number of differentially

expressed genes between two adjacent necrosis scores under a series of Bonferroni multi-test correction thresholds. The number of significant differentially expressed genes for comparisons of necrosis score 1 vs. 2 and score 3 vs. 4 are much fewer than that for comparison of necrosis score 0 vs. 1 and socre 2 vs. 3. This suggests that the expression of genes in the samples of the liver between minimal and mild necrosis and between moderate and marked necrosis is not much different respectively. The bulk of the difference is between liver samples with no necrosis and those with minimal necrosis and between those with mild necrosis and with moderate necrosis.

2.4.3 Inflammation follows Programmed Cell Death in Response to Hepatotoxicant Exposure

Gene Ontology analysis was performed on four lists of significantly differentially expressed gene from the comparisons of two adjacent necrosis scores which met the Bonferroni 0.05 threshold. Figure 2.1 shows GoMiner analysis of over-expressed genes from the four gene lists. The over-expressed genes in the selected gene lists for comparison of necrosis score 1 vs. 2 and score 3 vs. 4 do not show any significant biological processes, while the over-expressed genes in the selected gene lists for comparison of necrosis score 0 vs. 1 and 2 vs. 3 yielded several significant biological processes. One example is the biological process for inflammatory response. It is over-represented in the gene set discriminating necrosis level 2 and 3 but not when comparing level 0 and 1. The programmed cell death pathway is over-represented in this latter comparison. Therefore, it appears that after exposure of the liver to these hepatotoxicants in rats, programmed cell death is activated in the samples that exhibited mild or minimal necrosis while the inflammatory response is activated in the liver samples showing moderate or marked necrosis.

Both the ANOVA statistical analysis and GoMiner biological process analysis resulted in the conclusion that the gene expression, biological pathways and processes in the liver samples that manifest minimal or mild necrosis are very similar and difficult to differentiate from one another. The same is the case for liver samples revealing moderate or marked necrosis. However, the biological response in the liver of the samples with necrosis level comparisons of none vs. minimum and mild vs. moderate resulted in significant biological differences. Thus, for the purpose of this study, the five necrosis severity scores were combined into three necrosis levels, (no necrosis as level 0, minimal and mild necrosis

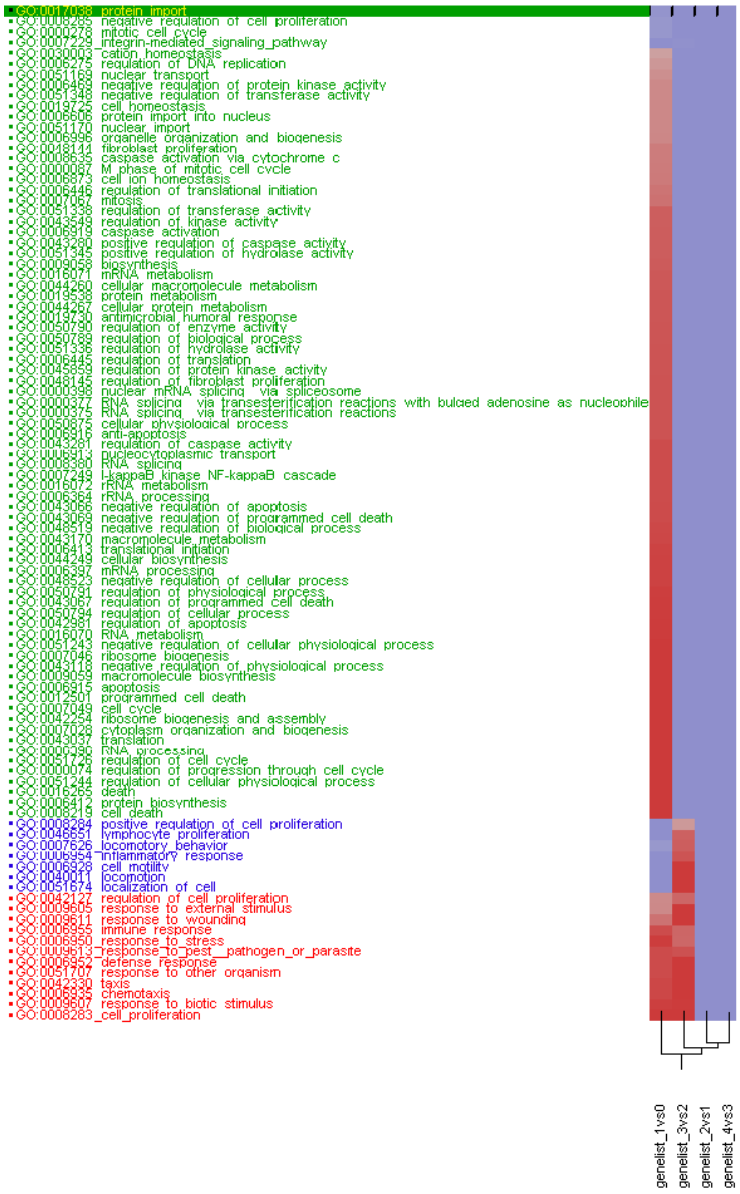


Figure 2.1: High-Throughput GoMiner analysis of 4 groups of over-expressed genes. There are 4 groups of significant differentially expressed gene lists, one for comparison of necrosis level 0 vs. 1 with 2683 significant genes, one for comparison of necrosis level 1 vs. 2 with 171 significant genes, one for comparison of necrosis level 2 vs. 3 with 910 significant genes, and one for comparison of necrosis level 3 vs. 4 with 131 genes. The four gene lists are labeled on the bottom of figure and over-represented biological processes are labeled at the left of the figure. The red color indicates that the p-value is smaller than the FDR rate of 0.05 whereas blue represents p-values larger than an FDR rate of 0.05. The smaller the p-value, the more intense the color.

as level 1, and moderate and marked necrosis as level 2) in order to identify genes that can predict the level of necrosis with a high degree of certainty from samples that share a common biological response. In the end, the sample size in each level increases and the statistical power to differentiate the newly defined levels of necrosis improves.

2.4.4 Gene Classifiers that Predict Necrosis

Selection of Predictor Genes

First, a one-way ANOVA was fitted to extract the significant differentially expressed genes among the three newly defined necrosis levels. The same one-way ANOVA model as described above was applied with the modification that the necrosis level now was from 0 to 2 instead of from severity score 0 to 4. Three contrasts were performed to identify differentially expressed genes between level 0 vs. 1, 1 vs. 2 and 0 vs. 2. From this analysis, 8561 genes were significantly differentially expressed either in the contrast of level 0 vs. 1 or in the contrast of level 1 vs. 2 using Bonferroni multi-test correction at p-value threshold of 0.05 (data not shown). Gene Ontology analysis performed on the two gene lists generated from the 0 vs. 1 and 1 vs. 2 level comparisons from the redefined groups reconfirms that inflammation follows programmed cell death when the samples are exposed to the hepatotoxicants (data not shown). Inflammation is significant only in the1 vs. 2 comparison while program cell death is more significant in 0 vs. 1 comparison.

For building a model for prediction, all the normalized data from the 318 arrays (treated animals) were used as a training set. The three necrosis levels were used as the class labels and the 8561 genes as the predictors. Random Forest and GEMS-SVM were two classification approaches used for prediction.

Prediction with Random Forest

The Random Forest classification approach grows many single classification trees and chooses the most popular vote over all trees in the forest. Each tree uses randomly selected samples (with replacement) as the training set. The Random Forest classification method selected 21 gene probes which have an out-of-bag (OOB) data error rate of 0.104 and standard deviation of 0.017. The names and annotation of the 21 selected gene probes are listed in Table 2.4. The list contains several genes related to inflammatory disease, cell-to-cell signaling and interaction, cell death, cellular movement, immune response, and cell organization.

Table 2.4: Annotation of the 21 selected Agilent probes using the Random Forest classification approach

Agilent Probe	Gene Accession #	Gene Name	Description
A_42_P458530	NM_139342	Ripk3	Rn receptor-interacting serine-threonine kinase 3
A_42_P487811	AW914054	Sema4g_predicted	(semaphorin) 4G (predicted)
A_42_P507284	NM_013111	Slc7a1	Rn solute carrier family 7, member 1
A_42_P517381	NM_133298	Gpnmb	Rn glycoprotein (transmembrane) nmb
A_42_P532103	NM_019905	Anxa2	Rn annexin A2
A_42_P594863	AI144754	Rnd1	Rho family GTPase 1 (predicted)
A_42_P621642	XM_341964	Lsp1	Rn similar to Lsp1 protein (LOC361680)
A_42_P695401	NM_031530	Ccl2, MCP-1	Rn chemokine (C-C motif) ligand 2
A_42_P710382	BF412297	TC466815	Transcribed locus
A_42_P730684	XM_214096	LOC289801	Rn similar to uridine phosphorylase (LOC289801)
A_42_P768467	BQ207775	RGD1305887	Similar to RIKEN cDNA (predicted)
A_42_P809565	NM_139324	Ehd4	Rn EH-domain containing 4
A_43_P10621	AI177116	Vasp_predicted	Vasodilator-stimulated phosphoprotein (predicted)
A_43_P11353	BC083855	Lcp1	Lymphocyte cytosolic protein 1 (predicted)
A_43_P11621	NM_012924	Cd44	Rn CD44 antigen
A_43_P12519	NM_031114	S100a10	Rn S100 calcium binding protein A10
A_43_P12698	NM_031832	Lgals3	Rn lectin, galactose binding, soluble 3
A_43_P12940	NM_053812	Bak1	BCL2-antagonist/killer 1
A_43_P13182	NM_133416	Bcl2a1	B-cell leukemia/lymphoma 2 related protein A1
A_43_P14045	AW914054	Sema4g_predicted	(semaphorin) 4G (predicted)
A_43_P15660	BC079312	Cxcl16	similar to chemokine (C-X-C motif) ligand 16

Rn represents Rattus norvegicus.

Prediction with GEMS

Classifiers used for prediction can be data dependent. Therefore, the Gene Expression Model Selector (GEMS) approach for prediction was also used. The best performing model selected by GEMS used a gene selection method consisting of a signal-to-noise ratio in a one-versus-rest fashion followed by a multicategory support vector machines method by Weston and Watkins (Weston and Watkins, 1999). The Weston and Watkins multicategory support vector machine can solve multi-class problems directly without decomposing it into multiple binary classification problems. Six genes were selected from the prediction model achieving a minimum cross validation error with prediction accuracy of 89.62% on training data (Table 2.5, 2.6). Five of the six genes were also selected as predictors using the Random Forest approach except for inosine monophosphate (IMP) dehydrogenase 1. This gene catalyzes the rate-limiting reaction of de novo GTP biosynthesis at the inosine monophosphate metabolic branch point and therefore is involved in the regulation of cell proliferation.

Comparison of Results Between Random Forest and GEMS

Using the 21 gene probes selected by Random Forest and the 6 gene probes selected by the GEMS-SVM classifier achieved the same average prediction accuracy of 89.62% on the training data (Table 2.6). Comparison of the prediction calls for the training data set from the two approaches indicates that the GEMS-SVM approach tends to misclassify the necrosis label towards the lower end of the true severity score class than the Random Forest approach (Table 2.7). Assessing the prediction accuracy for each compound in the training data set revealed that different compounds show different prediction accuracies, ranging from 83.33% to 100% (Table 2.6). The thioacetamide, monocrotaline, galactosamine, and diquat exposed samples always showed lower prediction accuracies (between 83%-89%) using both the Random Forest and GEMS-SVM approaches. The bromobenzene, 1,4-dichlorobenzene and N-nitrosomorpholine exposed samples were typically predicted with accuracies greater than 94% using either the Random Forest or GEMS-SVM approaches. The samples exposed to1,2-dichlorobenzene show better prediction accuracy using the Random Forest approach than the GEMS-SVM approach.

Both the Random Forest classifier and GEMS-SVM classifier misclassfied 33 animals from the training set. There are 24 animals in the overlap; thus, a total of 42 animals were misclassified either from the Random Forest approach or from the GEMS-SVM approach. Further verification using clinical chemistry data was performed for animals where

Table 2.5: Annotation of the 6 selected Agilent probes using the GEMS-SVM classification approach

Agilent Probe	Gene Accession #	Gene Name	Description
A_42_P458530 *	NM_139342	Ripk3	Rn receptor-interacting serine-threonine kinase 3
A_42_P507284 *	NM_013111	Slc7a1	Rn solute carrier family 7 member 1
A_42_P695401 *	NM_031530	Ccl2, MCP-1	Rn chemokine (C-C motif) ligand 2
A_42_P768467 *	BQ207775	RGD1305887	Similar to RIKEN cDNA 2310057H16 (predicted)
A_43_P11307	XM_342650	Impdh1_predicted	Rn similar to Impdh1 protein (LOC362329)
A_43_P11621 *	NM_012924	Cd44	Rn CD44 antigen

Rn represents Rattus norvegicus.

Overlap with the 21 selected Agilent probes using the Random Forest classification approach.

Table 2.6: Prediction accuracy of the training and test data sets using Random Forest and GEMS-SVM

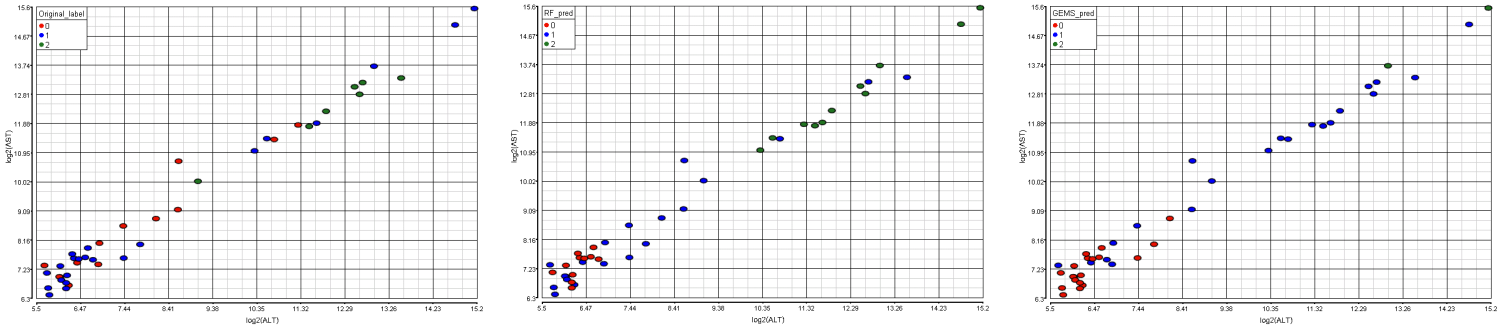
	Compounds	Sample Size	Prediction Accuracy	
			Random Forest	GEMS
Training	1,2-dichlorobenzene	34	91.18.%	85.29%
	1,4-dichlorobenzene	36	94.44%	94.44%
	bromobenzene	36	100.00%	94.44%
	diquat	72	86.11%	88.89%
	galactosamine	36	83.33%	86.11%
	monocrotaline	32	84.38%	84.38%
	N-nitrosomorpholine	36	94.44%	100.00%
	thioacetamide	36	86.11%	83.33%
	All training	318	89.62%	89.62%
Test	acetaminophen (NCT)	36	88.89%	88.89%
	acetaminophen (NTP)	64	87.5%	87.50%
	acetaminophen (Informatics Challenge)	108	90.74%	89.81%
	carbon tetrachloride	72	77.78%	77.78%
	Ally Alcohol	95	64.21%	58.95%

Table 2.7: Prediction accuracy for all the training data samples using Random Forest and GEMS-SVM

		Random Forest Predict			GEMS Predict		
		0	1	2	0	1	2
TRUE	0	151	12	1	155	9	0
	1	11	94	6	15	94	2
	2	0	3	40	0	7	36

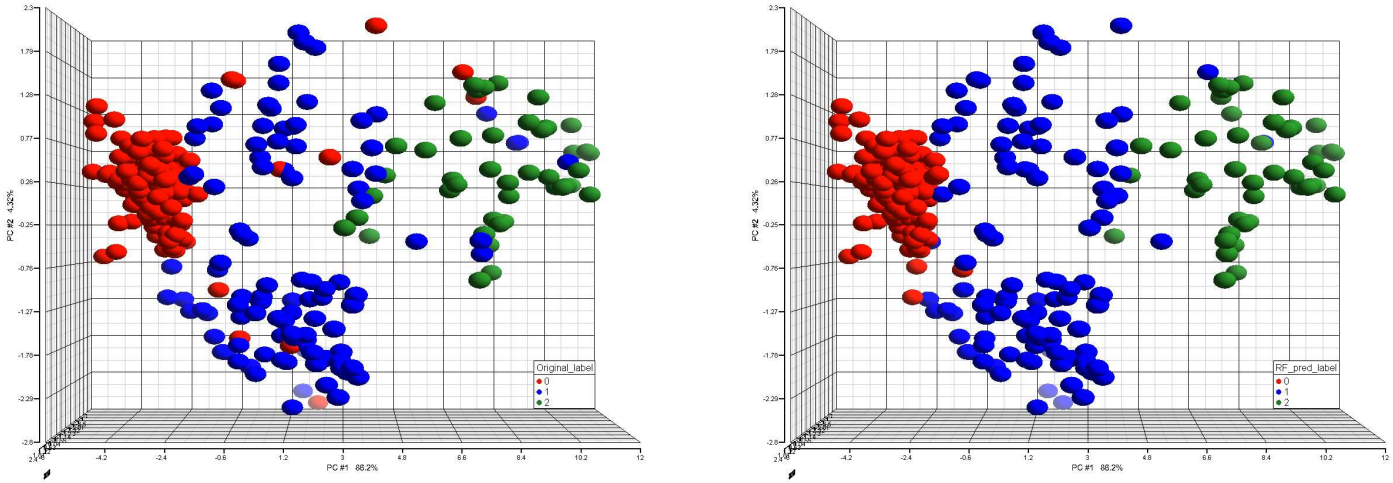
Table 2.8: Correlation analysis of ALT and AST with the necrosis class label or Random Forest predicted or GEMS-SVM predicted label

	Necrosis class label	GEMS_pred	RF_pred	log_2ALT	log_2AST
Necrosis class label	1				
GEMS_pred	0.095585	1			
RF_pred	0.146186	0.680875	1		
log_2ALT	0.448879	0.749506	0.758779	1	
log_2AST	0.417822	0.778307	0.743858	0.990228	1



(a) Redefined group from original label (b) Random Forest predicted label (c) GEMS-SVM predicted label

Figure 2.2: The scatter plot of log2 (ALT) and log2 (AST) levels of all 42 disagreement animals from Random Forest and GEMS-SVM classifier colored by the class label. (a) The scatter plot is colored by the necrosis class label according to the redefined groups; (b) The scatter plot is colored by the Random Forest predicted label; (c) The scatter plot is colored by the GEMS-SVM predicted label.



(a) Colored in the original class label

(b) Colored in the Random Forest predicted class label

Figure 2.3: PCA using the 21 selected genes. The necrosis levels increase from left to right. The red color represents animals with necrosis label 0, blue represents animals with necrosis label 1 and green represents necrosis level 2. The color in (a) represents the original class label, while in (b) the color represents the Random Forest predicted class label.

there was disagreement between the predicted necrosis level and the necrosis class label according to the redefined groups (data not shown). The correlation analysis (Table 2.8) and scatter plot (Figure 2.2) were performed on the data to identify the concordance of log_2ALT / log_2AST level, the predicted class label, and the necrosis class label based on the redefined groups. The Random Forest and GEMS-SVM predicted labels for the misclassified samples were more consistent with ALT / AST levels than the necrosis class label (Figure 2.2, Table 2.8). The Random Forest and GEMS-SVM predicted labels for the correctly classified samples were just as consistent (correlations $> +0.83$) with log_2ALT / log_2AST levels as the necrosis class label (Data not shown). This result corroborates the prediction call. Comparing the scatter plot colored by the Random Forest predicted label (Figure 2.2(b)) vs. colored by the GEMS-SVM predicted label (Figure 2.2(c)) on those misclassified samples revealed that the Random Forest method tends to predict the samples more towards a severe necrosis level while the GEMS-SVM approach tends to predict them towards a less severe necrosis level (see Table 2.7 as well). Figure 2.3 is a principal component analysis (PCA) on all 318 training animals using the 21 selected gene probes illustrating the level of necrosis separated along the PC1. Figure 2.3(a) is colored by the necrosis class label and figure 2.3(b) is colored by the predicted label. Both figures show that the level of necrosis increases from left to right along the PC1.

To validate the prediction using the selected genes, the classifier was applied to the test data. The test data sets include studies of agents that manifested centrilobular hepatocyte necrosis or periportal necrosis. The chemical compounds in the testing data sets include three different studies of rat livers exposed to acetaminophen, one to carbon tetrachloride and one to allyl alcohol. When the Random Forest classifier was applied to the acetaminophen-treated sample data, the prediction accuracies were 88.89%, 87.5% and 90.74% respectively for the data sets from the National Center for Toxicogenomics study #8 (NCT008), National Toxicology Program (NTP) and National Center for Toxicogenomics informatics challenge study (NCT informatics challenge). The Random Forest classifier produced 77.78% prediction accuracy for the carbon tetrachloride-treated test samples. Surprisingly, only 64.21% prediction accuracy was achieved with the classifier applied to the allyl alcohol-treated test samples (Table 2.6). Similarly, the GEMS-SVM classifier with the 6 predictive genes achieved prediction accuracies of 88.89%, 87.5% and 89.81% respectively for three acetaminophen data sets from NCT008, NTP and NCT informatics challenge. In the case of the GEMS-SVM classifier, it achieved the same prediction accuracy of 77.78%

as the Random Forest classifier on the carbon tetrachloride test samples. Not surprisingly, similar to the Random Forest prediction of the test data, the GEMS-SVM classifier model performed poorly on the allyl alcohol sample set achieving only 58.95% prediction accuracy. Generally speaking, the GEMS-SVM classifier model performed slightly worse than the Random Forest model on the test data, probably due to the smaller number of genes in the former model.

To test the significance of the 21 genes in the classifier 10,000 samples were generated by randomly selecting 21 genes from the array to use for training of the Random Forest classifier and testing on the test data. The median prediction accuracy for the 10,000 randomly generated samples is 0.701. About 4.7% of the random samples achieved a prediction accuracy greater than 0.764. Only 4 out of the 10,000 samples achieved a prediction accuracy greater than or equal to 0.808 (the accuracy for the 21 genes using the Random Forest classifier). Therefore, the significance of the prediction accuracy of the selected 21 genes in the Random Forest classifier has a p-value < 0.0005.

Biological Pathway and Gene Network Analysis

Pathway analysis of the 21 predictor genes revealed a central regulating role of tumor necrosis factor (TNF), Jun and TP53 (Figure 2.4). The majority of the predictor genes (17 out of 21) are regulated in their expression by those transcription factors. All genes within the network scope were retrieved and mapped to the corresponding probes on the Agilent chip. Bayesian gene networks were reconstructed using the discretized expression profile of all mapped probes from the training samples. The reconstructed Bayesian networks show the statistical dependence between the transcript levels of genes. The edges between the genes denote inferred interactions. Animals treated with a low dose (i.e. non-toxic dose) of the agents usually don't manifest necrosis. Only samples treated with a high dose (i.e. the more toxic dose) of the agents were selected (including the samples from training data set excluding 1-4-Dichlorobenzene) for reconstruction of the Bayesian networks. The networks with the highest posterior probability of the model given the data were collected from a large number of heuristic searches. The frequency distribution histogram (Figure 2.5) details the number of edges that reoccur in 500 networks that were reconstructed. There are potentially 1024 edges for the 32 genes used to generate the network (including the self-directed edge), 837 edges not presented (found zero times), 17 edges presented over 400 times, and 10 edges presented between 300 and 400 times. The probability of each edge being presented was calculated using weighted average scoring. Figure 2.6 shows

the consensus network recreated using the highest scoring network from each of the 500 runs of the algorithm, including all edges with a probability of being presented greater than 0.6. Four gene-to-gene interactions (S100 Calpactin [S100a10] with annexin A2 [Anxa2], Lectin, galactose binding, soluble 3 [Lgals3] with Epidermal Langerhans cell protein [Lcp1], Tumor protein p53 [TP53] with Cathepsin H [Ctsh] and Mitogen activated protein kinase 1 [Mapk1] with Caspase 8 [Casp8]) with probabilities greater than 0.8 are consistent with the interactions in the biological pathway generated from curation of scientific literature (Figure 2.4).

2.5 Discussion

Exposure of living organisms to environmental stressors typically results in the manifestation of phenotypic changes. Hepatotoxicants particularly target the liver and cause a variety of liver injuries. One type of damage is necrosis, a degenerative process leading to cell death. Several toxicants elicit necrosis of the hepatocytes from acute exposure to the liver. In this study, a compendium of gene expression data (standardized by way of a common array platform, experimental procedures and data analysis processes [Lobenhofer et al. in press]) was acquired from rat livers exposed to hepatoxicants and used in an attempt to define gene expression patterns as a signature that is highly predictive of the level of necrosis. Several of the agents in the compendium were known to cause centrilobular necrosis, midzonal necrosis, multifocal necrosis, apoptosis, cholestasis or sub-toxicity at high doses but exposure of the samples to many of them resulted in a general form of diffuse necrosis or were non-toxic (Table 2.1). This phenotype was used as an anchor to identify genes which predict the level of necrosis of the rat liver with a high degree of accuracy.

The eight compounds in the training data set provided sufficient representation of necrosis with enough diversity in the level of the phenotype to differentiate groups based on severity scores (Table 2.2). In addition, a prevailing advantage of this study was identifying genes related to necrosis which may be directly related to cell death caused by exposure to the compounds while filtering out genes related to other mechanisms from a single compound. This was accomplished by using the histopathological class label of the biological samples for necrosis as defined by board certified pathologists. Then ANOVA pairwise contrasts of the severity groups and GoMiner analysis of the gene expression data were leveraged to subset the samples into groups with similar phenotypic changes exhibited

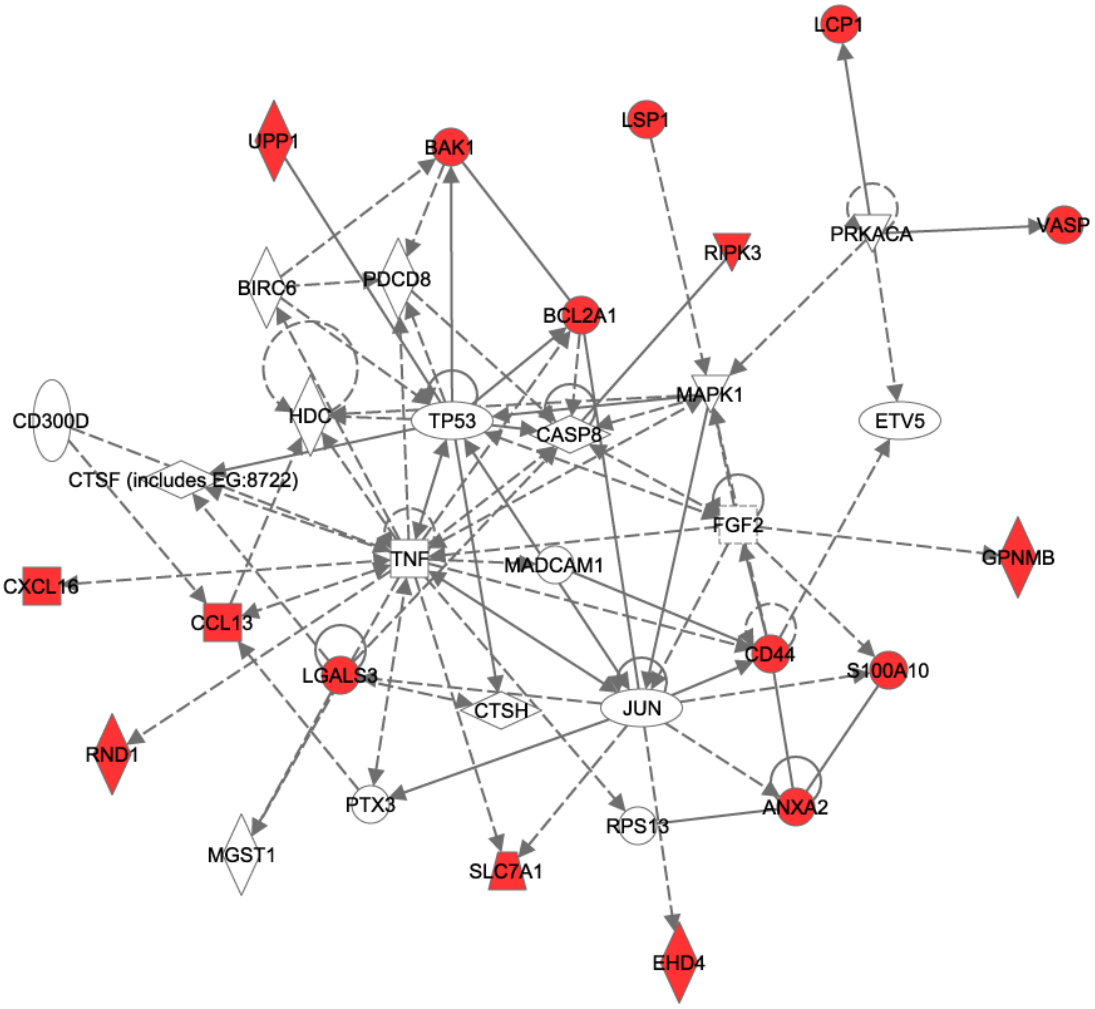


Figure 2.4: Pathway analysis. The 21 gene probes corresponding to 20 genes, were analyzed by Ingenuity Pathway Analysis software version 5.1. 18 out of 20 genes were annotated for gene network construction. 17 out of 20 over-expressed genes were mapped to the same network space. The red nodes represent the 17 selected genes. The pathway analysis revealed a central regulating role of tumor necrosis factor (TNF), Jun and TP53.

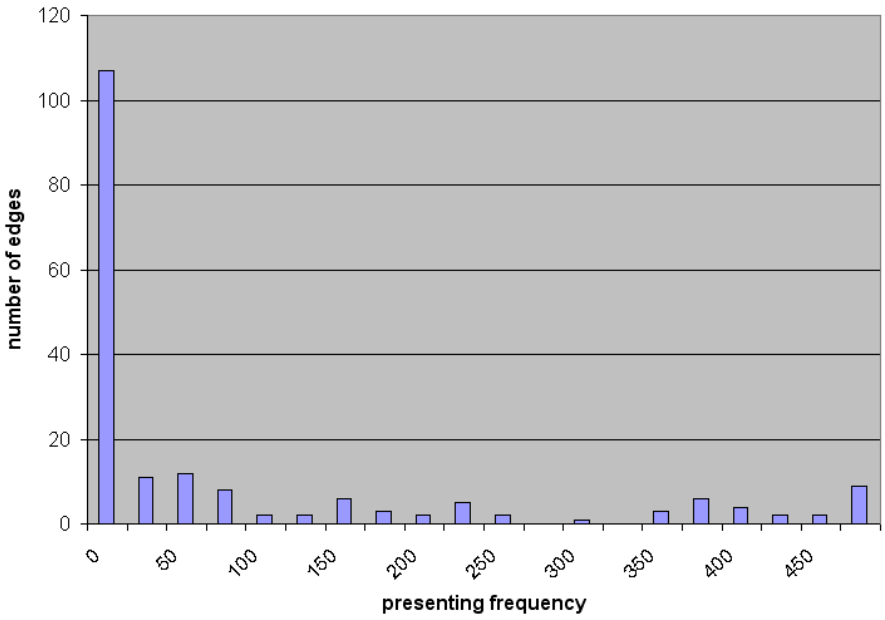


Figure 2.5: Histogram of the number of edges reoccurring in 500 networks. 187 possible edges present in at least one of 500 networks are plotted. 107 out of 187 edges are presented less than 25 times in 500 networks. The x-axis is the number of occurrences of the edges and the y-axis is the frequency.

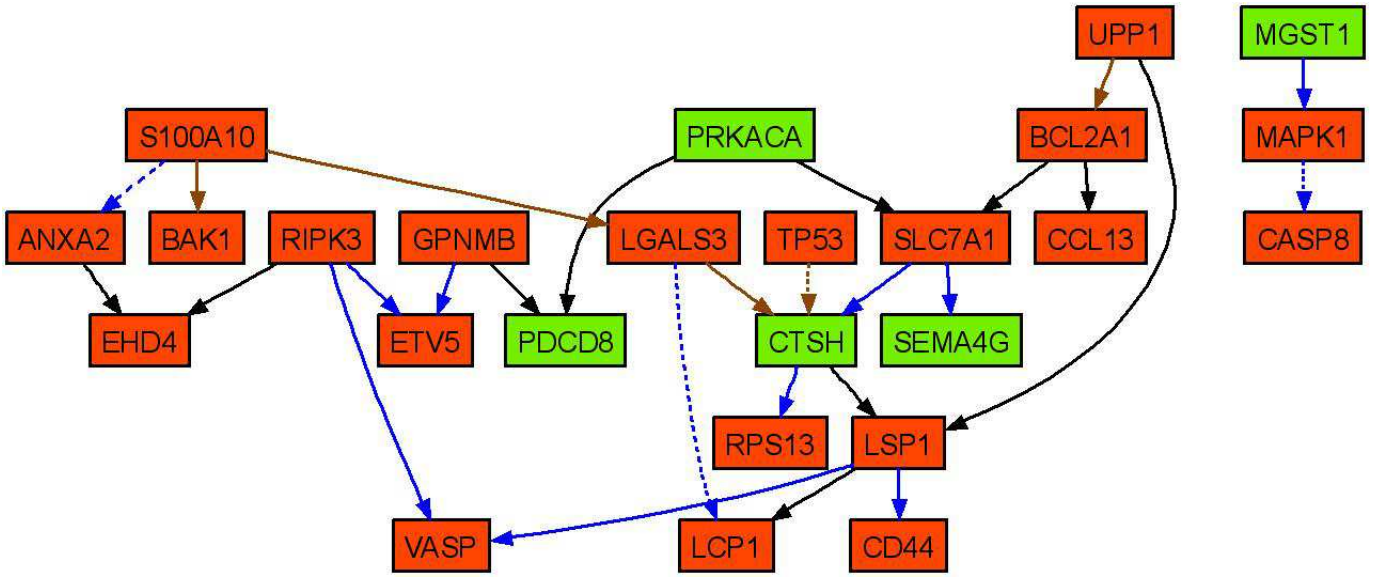


Figure 2.6: Reconstructed (consensus) gene network. The reconstructed Bayesian network was generated from the gene expression data from a total of 32 gene profiles. It depicts the statistical dependence between the transcript levels of the genes. The red nodes represent the up-regulated genes and green nodes represent the down-regulated genes both when the necrosis severity increases. Blue edges have a probability greater than 0.9, brown edges between 0.8 to 0.9, and black edges between 0.6 to 0.8. The dashed lines represent the edges consistent with the Ingenuity Pathway Analysis in Figure 4.

by necrosis of the liver and comparable overrepresentation of biological processes (Figure 2.1). Using these groups as class labels for the level of necrosis in the training set and two gene selection/classification approaches (Random Forest and GEMS-SVM), the author was able to identify subsets of genes which yielded a low prediction error rate during cross validation of the classifiers. Preliminary analysis of the liver gene expression data within each dose/time group by Lobenhofer et al. (in press) revealed compound-specific separation of the samples. Although classification of the blood data was better in higher dose groups at the later time points as compared to the liver data, classifiers derived from it was not able to predict animals in some cases where the hepatotoxicant elicited a different phenotypic response with the animals of a particular dose/time group. Interestingly, concordance analysis of ALT and AST enzyme levels with the class label of the level of necrosis and the predicted class labels revealed that the latter is more consistent with the enzyme levels. One potential reason for this could be that the microarray samples are from the whole liver whereas the histopathology samples used for scoring the extent of necrosis were only from two independent slices of the liver specimen. In a limited study, Heinloth et al. (2007) also showed that gene expression analysis is more informative than histopathological evaluations and offers unique advantages to liver biopsy evaluations. Another explanation could be that certain animals may develop the phenotype at a later time point after treatment than the time point when the samples were taken for analysis. Therefore, the histopathology samples may not completely represent the liver toxicity. However, the gene expression analysis in this study, redefining of the class labels for the level of necrosis exhibited in the samples and selection of predictor genes for necrosis are geared towards capturing the biological processes and mechanistic pathways that may govern the manifestation of the phenotype from a low level of necrosis to its highest level.

The hallmark of a good classifier is that it generalizes to other data sets. In other words, can an equal or better prediction accuracy be attained with a test data set than with the training data set? Using independent/non-standardized gene expression data sets acquired from the exposure of rat liver samples to a different set of hepatotoxicants (acetaminophen, carbon tetrachloride and allyl alcohol), It was shown that the prediction accuracies of either of the two classifiers were roughly 80% overall but approximately 90%, about 80% and around 60% for acetaminophen-, carbon tetrachloride- and allyl alcohol-exposed samples respectively with a p-value < 0.0005 for the significance of the prediction using the Random Forest classifier (Table 2.6). The dramatic difference in prediction accuracy

could be related to bioactivation mechanisms involved in the manifestation of centrilobular necrosis in the case of acetaminophen- and carbon tetrachloride-toxicity as opposed to periportal necrosis in the case of allyl alcohol-toxicity. In the former, the abundance of cytochrome P450 plays a critical role whereas in the latter, higher oxygen levels are responsible (Casarett et al., 2001). Another reason may be a site-specific batch effect since the allyl alcohol microarray data was generated at a different location than the acetaminophen and carbon tetrachloride microarray data. The training data and testing data are plotted together using PCA of the expression data from the signature-the 21 selected genes (data not shown). The testing data shows similar distributions of the samples as the training data and also indicates that the necrosis level increases from right to left along PC1. The acetaminophen- and carbon tetrachloride- exposed samples show a similar data dispersion range as the training data while the ally alcohol- exposed samples are more compressed along the first PC.

Previous reports suggest that acetaminophen (an analgesic) hepatotoxicity stems from the release of soluble proinflammatory and proapoptotic enzymes such as tumor necrosis factor (TNF)(Blazka et al., 1996). It is believed that at toxic doses, acetaminophen induces apoptosis in the liver which fails to go to completion and then degenerates into necrosis (Adams et al., 2001; El-Hassan et al., 2003; Jaeschke et al., 2004; Pierce et al., 2002). Presumably, at low doses of acetaminophen, cytokines and chemokines initiate an inflammatory response that alters the extent of acetaminophen-induced toxicity (Holt and Ju, 2006). In the case of carbon tetrachloride-induced toxicity, a release of cytokines and TNF-α mediated apoptosis precedes toxic doses (Weber et al., 2003) and with N-nitrosomorpholine -(NNM, a genotoxic hepatocarcinogen) induced toxicity, an inflammatory response possibly ensues prior to the injury and up-regulation of the p73 protein (a p53-related gene product), seems to lead to the initiation of apoptosis in the liver (Tudzarova-Trajkovska and Wesierska-Gadek, 2003).

Genes from the predictive models in this study (See Tables 2.4 and 2.5) have biological functions related to the regulation of apoptosis (Ripk3 and Bcl2a1) or are involved in a chemokine/inflammatory response (Ccl13 also known as CCL2/MCP-1), Cxcl16 and Lgals3. Pathway analysis of the predictor genes revealed a central regulating role of tumor necrosis factor (TNF), Jun and TP53 (Figure 2.4). The majority of the predictor genes in the signature (17 out of 21) are regulated in their expression by these transcription factors. Therefore, the results in this study are generally in agreement with the current

hypothesis that TNF mediates liver injury and genes such as Jun and TP53 are closely involved in necrotic changes in response to exposure to some hepatotoxicants. Surprisingly, Monocyte chemoattractant protein-1 (MCP-1), a serum factor gene and chemokine that is in the predictor gene list, was shown to have its protein product differentially expressed in acetaminophen-treated rats (Merrick et al., 2006) and is induced by TNF-α (Chen et al., 2004). This regulation might be a reflection of a reparative process following liver injury by acetaminophen-toxic exposure or could be a contributor to the insult. Although the role of MCP-1 in liver injury is controversial (Merrick et al., 2006), new evidence using MCP-1 deficient mice suggests that interference of the gene's expression is sufficient for altering the processes that lead to severe carbon tetrachloride-induced liver injury (Zamara et al., 2007). However, caution must be taken as a more complicated biological response to liver injury is likely since there are hepatotoxicants, such as monocrotaline- (MCT, a pyrrolizidine alkaloid plant toxin), where an inflammatory response ensues secondarily to injury of the liver and TNF-α appears to not be primarily responsible for the hepatotoxicity (Copple et al., 2003). In addition, transcription factors such as TNF-α and TP53 have both pro- and anti-apoptotic effects. TP53 keeps the cell from progressing through the cell cycle if there is damage to DNA but can also cause the cell to enter apoptosis if the damage cannot be repaired. Similarly, TNF-α can induce pro-apoptotic signaling mechanisms (Messmer et al., 1999) or induce resistance against apoptosis (Qin et al., 2007) dependent on the overall condition of the cell and its microenvironment.

The reconstructed Bayesian network from the toxic exposures of the hepatotoxicants (Figure 2.6) revealed several gene interactions that are consistent with interactions in the pathway that was generated from curated scientific literature (Figure 2.4) and points to apoptosis-related genes in necrosis-mediated toxicity. For instance, activation of mitogen-activated protein kinase 1 [Mapk1] is known to induce apoptosis and in the case of unrepairable DNA damage, tumor protein p53 has the capacity to prevent cells from dividing, leading to programmed cell death. In addition, caspase 8 [Casp8] is an apoptosis-related cysteine peptidase whose activation has been shown to be an essential step for inducing apoptosis. Bear in mind that the network is a consensus one, has only positive, one-way, acyclic interactions and was generated from microarray data alone using a limited number of genes. However, the confidence of each gene-to-gene edge (interaction) was calculated by performing 500 simulated annealing searches. Including the literature pathway as a prior assessment of the impact of a gene on its neighbors could potentially help to reconstruct

the relationships of the genes more precisely. Due to the limited time points and sample size, only a static Bayesian network was able to be reconstructed. Presumably, with more samples and times point along with additional biological information about the samples and the genes, a more informative and precise Bayesian gene network can be obtained to infer the gene interactions in a time and/or dose dependent manner.

2.6 Conclusions

In this study it has been demonstrated that gene expression signatures can predict, with a high degree of accuracy, the severity of necrosis of the liver elicited by acute exposure of rats to a variety of hepatotoxicants. First, the class discovery approach using ANOVA and GoMiner pathway analysis provided well-defined groups. This step is important since the redefined groups are more correlated with the liver injury as measured by ALT and AST enzyme levels. In addition, the gene selection strategy using Random Forest and GEMS-SVM improved the accuracy of predicting the severity of necrosis. Furthermore, the gene expression signature led to the identification of the molecular pathways that exhibited biological relevance to the manifestation of necrosis. Finally, pathway and gene network analyses revealed several gene interactions suggesting that apoptosis may be a process involved in the manifestation of necrosis of the liver from exposure of the hepatotoxicants in rats.

Chapter 3

Variable Selection for Linear MSVM

3.1 Abstract

Microarray techniques provide new insight into cancer diagnosis using gene expression profiles. However, molecular diagnosis based on high-throughput genomic data presents a major challenge due to the overwhelming number of variables versus the small sample size and complex multi-class nature of tumor samples. SVM have demonstrated superior performance in classifying high-dimensional and low sample size data. However, SVM was designed for binary classification problems. To extend the binary SVM to multi-class SVM, the traditional methods is to decompose a single multi-class problem into multiple independent binary problems using one-versus-one or one-versus-all approaches. But this decomposition does not consider the correlation between different classes, i.e. tumor subtypes. Further, the SVM original regularization formulation does not accommodate the variable selection, which provides a solution difficult to interpret. In the genomic application it is important to achieve a compact model for better interpretation besides the high prediction accuracy. Most existing approaches handle this issue by selecting genes prior to classification, which do not consider the correlation among genes when genes are selected by univariate rank. In this paper, new SVM methods are developed which can perform multi-class classification and variable selection simultaneously and learn an optimum classifier by consider all classes and all gene at the same time.

The original multi-class SVM (MSVM) proposed by Crammer and Singer (2001) does not provide the variable selection. Making use of the MSVM loss function by Crammer and Singer (2001), four new methods presented in this study can perform multi-class classification and variable selection simultaneously through adding four different sparse regularization terms. The methods are applied to a simulation data set and two real microarray data sets, acute leukemia study (Golub et al., 1999) and small round blue cell tumors (Khan et al., 2001). The results demonstrated that the methods can select a smallest number of genes (compared with other existing methods) that accurately predict the tumor subtypes. The compact model learned can identify which genes are the most important genes for discriminating which kind of tumor subtypes. Further, the identified genes from four different methods show big overlap, which demonstrates a reasonable stability in the marker selection. The combination of high accuracy and small number of genes should make the methods a powerful tool for developing diagnostics of disease from expression data and defining targets of the therapeutic intervention.

3.2 Introduction

With the boost of high biomedical development in the biological area, such as microarray technique, a lot of high-throughput data with "large p small n" feature has been generated. For example, a typical microarray data has the variables in the magnitude of tens or hundreds of thousands (p), which greatly exceeds the size of training samples n, i.e. $p \gg n$. Most classical multivariate analysis methods have difficulties in handling such data due to the curse of dimensionality and ill-conditioned parameter estimation. However, the Support Vector Machine (SVM; Boser et al., 1992; Cortes and Vapnik, 1995) has shown success in analyzing such "large p small n" data set. SVM is under the regularization framework, which can control the model complexity when fitting the data as well as solving ill-conditioned parameter estimation problems. Further, SVM outperforms most other popular methods in terms of prediction accuracy.

SVM is originally designed for binary classification (Cortes and Vapnik, 1995). Its geometric interpretation is to identify an optimum hyperplane with largest margin to separate two classes. In reality, biologists are not only interested in distinguishing tumor from normal, but often interested in distinguishing different severity levels of cancer or different tumor (sub)types. Therefore, much research has been done to extend the binary

SVM for multi-class problems by decomposing a single multi-class problem into multiple independent binary classification problems (Dietterich and Bakiri, 1995; Allwein et al., 2000). For example, a four-class classification problem can be decomposed into 4 one-versus-rest binary classification problems or into 6 one-versus-one binary classification problems including all pairwise comparisons of each class. There are a lot of disadvantages by such indirectly solving a multi-class problem, which has been explained in Chapter 1 in detail. A better direct way to solve a multi-class problem is to consider all classes simultaneously. To extend the binary SVM to the multi-class support vector machine (MSVM), the hinge loss $[1 - y_i f(\mathbf{x}_i)]_+$ in binary SVM needs to be generalized. There are several generalizations of the loss function for MSVM proposed by Weston and Watkins (1999), Crammer and Singer (2001), Lee et al. (2004), Liu and Shen (2006). The MSVM approach used by Crammer and Singer (2001) is better in terms of direct generalization of separating hyperplanes and concept of margins from binary SVM. The MSVM by Crammer and Singer (2001) directly estimates K decision functions $\mathbf{f} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_K(\mathbf{x}))$ simultaneously if $y \in \{1, \ldots, K\}$. The decision rule, $\Phi(\mathbf{x}) = \arg\max_{k=1}^{K} f_k(\mathbf{x})$, is to assign a new input data $\mathbf{x}$ a class label r if $f_r(\mathbf{x})$ gives the highest value. The loss function of MSVM by Crammer and Singer (2001) is detailed in the method section.

The general SVM has the L_2 penalty, which is also known as the ridge penalty. The ridge penalty can shrink the fitted coefficient $\hat{\beta}$ towards zero, which efficiently controls the variance of $\hat{\beta}$, hence improving the performance of the model, especially when a lot of variables are highly correlated (Zhu et al., 2003). However, L_2 penalty cannot shrink coefficients to exactly zero, hence all variables are utilized in the learned model. Besides classification, another important task is to identify which genes are most important for tumor (sub)types classification and are related to the disease mechanism. Further, models with too many redundant noise features diminishes the model performance and prediction accuracy (Zhu et al., 2003). Therefore, variable selection is very important and necessary in classification and model learning. Traditional variable selection approaches first rank genes using some univariate measurements, such as p-values from hypothesis tests or correlation between variables and end point, etc. According to the rank, genes are sequentially added or removed from the model and the models are selected using cross-validation error or test error. The traditional variable selection approaches don't take into account the interaction and correlation among genes and select genes solely by their individual measurement. In addition, the variable selection and classification are two separate procedures in traditional

approaches and variables are not selected for the improvement of the classification accuracy, which may lead to suboptimal models and performance. This approach is also widely used in most classical classification methods which cannot handle "large p small n" data set, such as K Nearest Neighbor (KNN), logistic regression, linear discriminant analysis, quadratic discriminant analysis etc.

Bradley and Mangasarian (1998) first introduced the L_1 penalty into the binary SVM, which can perform the classification and variable selection simultaneously and show better prediction accuracy especially when there are redundant noise variables. L_1 penalty can not only reduce the coefficient's variance like L_2 penalty, but also shrink small coefficients to exact zero, which results in a sparse model. Therefore, a "large p small n" data set can be directly fed into the model and the SVM classifier can select important variables as well as perform classification simultaneously without pre-screening important genes before learning the classifiers. By this way, the global optimal model can be achieved, which has most compact selected variables and show best prediction performance. Zhu et al. (2003) further suggested an algorithm to compute the whole solution path of L_1-norm binary SVM over a range of tuning parameters. There are a lot of other forms of shrinkage methods used in binary SVM, such as the SCAD penalty (Zhang et al., 2006), SVM recursive feature elimination (SVM-RFE; Guyon et al., 2002), L_0 penalty (Weston et al., 2003) etc. All above variable selection methods were developed using binary SVM.

Variable selection in multi-class classification problem is considered in this study. MSVM has much more coefficients than binary SVM since there are K decision functions in MSVM but only one decision function in binary SVM. Therefore, The variable selection in MSVM is much more challenging than binary SVM. Not much work has been done in this area. Currently, there are only three publications about variable selection in MSVM so far (Wang and Shen, 2007; Lee et al., 2006; Zhang et al., 2006). Similar to the solution path of L_1-norm binary SVM computed by Zhu et al. (2003), Wang and Shen (2007) further computed the whole solution path of L_1-norm for MSVM. Zhang et al. (2006) proposed a new penalty form, supnorm penalty, which is a good fit for multi-class classification and leads to the more sparse model than L_1-norm MSVM. Lee et al. (2006) applied ANOVA decomposition on the kernel for MSVM variable selection, which can perform a nonlinear component selection in MSVM. All above methods of variable selection for MSVM are based on the loss function proposed by Lee et al. (2004). In this paper, new approaches for variable selection in MSVM were developed and several variable selection regularization forms using

the MSVM loss function proposed by Crammer and Singer (2001) were investigated given that the original MSVM by Crammer and Singer (2001) does not provide the variable selection. An adaptive weighted L_1 penalty and adaptive weighted supnorm penalty were investigated as well as the L_1 penalty and supnorm penalty. These methods were applied to a simulation data set and two real microarray data sets, acute leukemia study (Golub et al., 1999) and small round blue cell tumors (Khan et al., 2001). The results using four different penalties were not only compared to each other but also compared with the original MSVM by Crammer and Singer (2001) and the other three existing MSVM classifiers with variable selection. The approaches show better performance in terms of selecting the smallest number of genes (compared with the other methods) which were able to achieve a high prediction accuracy.

3.3 Methods

Given a training data set with n pairs of observations $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, which are *i.i.d.* from $P(\mathbf{x}, y)$, where $\mathbf{x} \in R^p$ and $y = \{1, 2, \ldots, K\}$, the goal of multi-class classification is to learn the optimum decision function(s) from $\{\hat{\mathbf{f}}(\mathbf{x}, \alpha), \alpha \in \Lambda\} : R^p \rightarrow \{1, 2, \ldots, K\}$. For MSVM, multiple discriminant functions $f_k(\mathbf{x}), k = 1, \ldots, K$, need be estimated. Each $f_k(\mathbf{x})$ is associated with a class k. The function $f_k(\mathbf{x})$ represents the strength of a sample$(\mathbf{x}, y)$ belonging to the class k. That is, $f_k(\mathbf{x}) \propto P(Y = k|\mathbf{x})$. Therefore, the decision rule is given by $\Phi(\mathbf{x}) = \arg \max_{k=1,\ldots,K} f_k(\mathbf{x})$. And the classification boundary between class k and class l is $\{\mathbf{x} : f_k(\mathbf{x}) = f_l(\mathbf{x})\}$.

Binary SVM

$$\text{separable case:} \qquad y_i f(\mathbf{x}_i) \geq 1 \tag{3.1}$$

$$\text{non-separable case:} \quad y_i f(\mathbf{x}_i) \geq 1 - \xi_i \tag{3.2}$$

MSVM

$$\text{separable case:} \qquad f_y(\mathbf{x}_i) - \max_{\substack{k \neq y \\ k=1,\ldots,K}} f_k(\mathbf{x}_i) \geq 1 \tag{3.3}$$

$$\text{non-separable case:} \quad f_y(\mathbf{x}_i) - \max_{\substack{k \neq y \\ k=1,\ldots,K}} f_k(\mathbf{x}_i) \geq 1 - \xi_i \tag{3.4}$$

In linearly separable binary SVM, the decision function learned must satisfy the constraint (3.1) for each observation $\mathbf{x}_i$. It means the functional margin, $y_i f(\mathbf{x}_i)$, must be greater or equal than 1. Correspondingly, when Crammer and Singer (2001) extend the concept of margins in the binary SVM to multi-class SVM, they defined the constraint (3.3) must be satisfied for each observation in separable case of linear MSVM. The constraint (3.3) requires that if $\mathbf{x}_i$ belongs to class y, the $f_y(\mathbf{x}_i)$ has to be at least 1 margin bigger than all other $f_k(\mathbf{x}_i)$, where $k = 1, \ldots, K$ and $k \neq y$. In linearly non-separable case of binary SVM, it is impossible to find a decision function $f(\mathbf{x})$ satisfying the constraint (3.1) for all observations $\mathbf{x}_i, i = 1, \ldots, n$, thus the slack variable, $\xi_i \geq 0$, is introduced for each observation $\mathbf{x}_i$ to control an upper bound of the misclassification error. The decision function for non-separable binary SVM must satisfy the constraint (3.2) for every observation $\mathbf{x}_i$. Crammer and Singer (2001) proposed the similar extension on non-separable case MSVM and defined the constraint (3.4). Obviously, Crammer and Singer (2001) suggested a reasonable and direct extension of the margin definition from binary SVM to the MSVM and they demonstrated that their proposed MSVM can attain state-of-the-art accuracy (Crammer and Singer, 2001).

The regularization problem of the linear MSVM by Crammer and Singer (2001) can be expressed as:

$$\min_{\beta,\beta_0,\xi} \sum_{i=1}^{m} \xi_i + \frac{1}{2}\lambda \sum_{k=1}^{K} \sum_{j=1}^{p} \beta_{kj}^2 \tag{3.5}$$

$$\begin{aligned} \text{subject to:} \quad & f_y(\mathbf{x}_i) - \max_{k \neq y} f_k(\mathbf{x}_i) \geq 1 - \xi_i, \ \forall i = 1, \ldots, n, k = 1, \ldots, K, \\ & \xi_i \geq 0, \ \forall i = 1, \ldots, n, \\ \text{where} \quad & f_k(\mathbf{x}_i) = \sum_{j=1}^{p} x_{ij}\beta_{kj}. \end{aligned}$$

Because there are K decision functions in MSVM, the penalty function need to penalize a coefficient matrix with $K \times p$ dimensions. The above optimization problem with quadratic objective function and linear constraints can be solved using standard quadratic programming techniques. The objective function (3.5) consists of two parts: loss function and penalty function. The loss function is the sum of all training errors, which represents the model fitting. The penalty function controls the model complexity. λ is a factor that balances the trade off between the model fitting and model complexity. To make the MSVM by Crammer and Singer (2001) more general, the intercept to each decision function was introduced. Thus, $f_k(\mathbf{x}_i)$ is redefined as $\sum_{j=1}^{p} x_{ij}\beta_{kj} + \beta_{k0}$. Here they used

the L_2 penalty. As stated before, SVM model with L_2 penalty utilizes all variables and hurts the prediction accuracy especially when there exist many redundant noise variables. The loss function proposed by Crammer and Singer (2001) was utilized; but introduced the new penalty functions into the regularization problem, which can shrink small coefficients β to exactly zero and select important variables to build model. Therefore, a sparse model will be achieved with improved prediction accuracy. Four new regularization forms with the loss function proposed by Crammer and Singer (2001) were investigated through including four different penalty functions: L_1 penalty, Adaptive L_1 penalty, Supnorm penalty and Adaptive Supnorm penalty.

3.3.1 L_1 Penalty

The L_1 penalty is also called LASSO penalty. The new regularization form with L_1 penalty is:

$$\min_{\beta,\beta_0,\xi} \sum_{i=1}^{m} \xi_i + \frac{1}{2}\lambda \sum_{k=1}^{K}\sum_{j=1}^{p} |\beta_{kj}| \tag{3.6}$$

$$\begin{aligned} \text{subject to:} \quad & f_y(\mathbf{x}_i) - \max_{k \neq y} f_k(\mathbf{x}_i) \geq 1 - \xi_i, \ \forall i = 1, \ldots, n; k = 1, \ldots, K, \\ & \xi_i \geq 0, \ \forall i = 1, \ldots, n. \end{aligned}$$

To get rid of the absolute operation in optimization problem (3.6), it is defined that $|\beta_{kj}| = \beta_{kj}^{+} + \beta_{kj}^{-}$ and $\beta_{kj} = \beta_{kj}^{+} - \beta_{kj}^{-}$, where $\beta_{kj}^{+} = \beta_{kj}$ if $\beta_{kj} \geq 0$, or 0, otherwise; $\beta_{kj}^{-} = -\beta_{kj}$ if $\beta_{kj} \leq 0$, or 0, otherwise. Thus, the L_1-norm MSVM in (3.6) is converted to a linear programming (LP) problem and can be solved using standard LP techniques in polynomial time. In addition, a sum-to-zero constraint $\sum_{k=1}^{K} f_k(\mathbf{x}) = 0$ is enforced to ensure the uniqueness of the solution.

$$\begin{aligned} & \sum_{k=1}^{K} f_k(\mathbf{x}) = 0 \\ \Rightarrow \quad & \beta_{10} + \beta_{11}x_1 + \ldots + \beta_{1n}x_n + \ldots + \beta_{k0} + \beta_{k1}x_1 + \ldots + \beta_{kn}x_n + \ldots = 0 \\ \Rightarrow \quad & \sum_{k=1}^{K} \beta_{k0} = 0, \sum_{k=1}^{K} \beta_{kj} = 0, \forall j = 1, \ldots, p, k = 1, \ldots, K \\ \Rightarrow \quad & \sum_{k=1}^{K} \beta_{k0} = 0, \sum_{k=1}^{K} (\beta_{kj}^{+} - \beta_{kj}^{-}) = 0, \forall j = 1, \ldots, p, k = 1, \ldots, K \end{aligned}$$

Therefore, the (3.6) becomes

$$\begin{aligned}
\min_{\beta,\beta_0,\xi} \quad & \sum_{i=1}^{m}\xi_i + \frac{1}{2}\lambda\sum_{k=1}^{K}\sum_{j=1}^{p}(\beta_{kj}^{+}+\beta_{kj}^{-}) \\
\text{subject to:} \quad & \sum_{j=1}^{p}(\beta_{yj}^{+}-\beta_{yj}^{-})x_{ij} - \sum_{j=1}^{p}(\beta_{kj}^{+}-\beta_{kj}^{-})x_{ij} + \beta_{y0} - \beta_{k0} \geq 1-\xi_i, \\
& \qquad\qquad \forall i, k = 1,\ldots,K, k \neq y, \\
& \sum_{k=1}^{K}\beta_{k,0} = 0; \sum_{k=1}^{K}(\beta_{kj}^{+}-\beta_{kj}^{-}) = 0, \forall j = 1,\ldots,p, \\
& \beta_{kj}^{+} \geq 0, \beta_{kj}^{-} \geq 0, \forall k = 1,\ldots,K, j = 1,\ldots,p, \\
& \xi_i \geq 0, \ \forall i = 1,\ldots,n.
\end{aligned}$$

3.3.2 Adaptive-L_1 Penalty

Adaptive L_1 penalty is also called Adaptive-LASSO, which is used by Zou (2006) in regression models, Wang et al. (2007) in LAD model, and Zhang and Lu (2007) in Cox's proportional hazards model. L_1-norm applies equal penalty to all coefficients. Instead, adaptive L_1-norm gives different penalties for different coefficients. For large coefficients, small penalty is given, while for small coefficients, large penalty is given. Therefore, the large coefficients can be protectively preserved in the model selection procedure while the small coefficients are shrunk to zero much faster, which leads to more sparse model Zhang et al. (2006). In MSVM, the adaptive L_1-norm uses the penalty form $\lambda\sum_{k=1}^{K}\sum_{j=1}^{p} w_{kj}|\beta_{kj}|$, each coefficient β_{kj} has its own adjustable penalty $w_{kj} = \frac{1}{|\beta'_{kj}|}$, where β'_{kj} is solved from the L_2-norm MSVM Zhang et al. (2006). if $\beta'_{kj} = 0$, the corresponding β_{kj} will be set to zero Zhang et al. (2006). The estimates for the weights are data-driven penalties from the L_2-norm MSVM. Further, comparing to other estimates the ridge penalty in L_2-norm MSVM provides a stable and robust estimation especially if strong collinearity exists among covariates. If β'_{kj} is a large coefficient from L_2-norm MSVM, then w_{kj} is small and β_{kj} is given a small penalty; and vice versa. Therefore, a large coefficient is given a small penalty and a small coefficient is given a large penalty. The new optimization problem with adaptive L_1 penalty has the same constraints as L_1-norm MSVM, but different objective function as the following

$$\min_{\beta,\beta_0,\xi} \quad \sum_{i=1}^{m}\xi_i + \frac{1}{2}\lambda\sum_{k=1}^{K}\sum_{j=1}^{p}\frac{\beta_{kj}^{+}+\beta_{kj}^{-}}{|\beta'_{kj}|}.$$

3.3.3 Supnorm Penalty

Supnorm penalty for MSVM is proposed by Zhang et al. (2006). There are K decision functions in MSVM, therefore a coefficient matrix with $K \times p$ dimensions has K coefficients for each variable $x_j, j = 1, \ldots, p$. If a variable is not important, then all k coefficients associated with this variable should be zero. Motivated by this, Zhang et al. (2006) proposed to penalize the maximum absolute value of K coefficients for each variable. In the new regularization form presented in this study, instead of penalizing every coefficient separately, the maximum absolute value of K coefficients associated with the same variable is penalized. Thus, if the maximum value of K coefficients is equal to zero, all K coefficients should be equal to zero, which may lead to a more sparse model than L_1-norm MSVM. To implement this idea, $\eta_j \geq 0, j = 1, \ldots, p$ is introduced with $\eta_j = \max_{k=1,\ldots,K} |\beta_{kj}|$. Therefore the new regularization form with supnorm is:

$$
\begin{aligned}
\min_{\beta,\beta_0,\xi} \quad & \sum_{i=1}^{m} \xi_i + \frac{1}{2}\lambda \sum_{j=1}^{p} \eta_j \\
\text{subject to:} \quad & \sum_{j=1}^{p} (\beta_{yj}^{+} - \beta_{yj}^{-}) x_{ij} - \sum_{j=1}^{p} (\beta_{kj}^{+} - \beta_{kj}^{-}) x_{ij} \geq 1 - \xi_i, \\
& \qquad\qquad \forall i, k = 1, \ldots, K, k \neq y, \\
& \eta_j \geq \beta_{kj}^{+} + \beta_{kj}^{-}, \forall k = 1, \ldots, K, j = 1, \ldots, p, \\
& \sum_{k=1}^{K} \beta_{k,0} = 0; \sum_{k=1}^{K} (\beta_{kj}^{+} - \beta_{kj}^{-}) = 0, \forall j = 1, \ldots, p, \\
& \beta_{kj}^{+} \geq 0, \beta_{kj}^{-} \geq 0, \eta_j \geq 0, \forall k = 1, \ldots, K, j = 1, \ldots, p, \\
& \xi_i \geq 0, \ \forall i = 1, \ldots, n.
\end{aligned}
$$

3.3.4 Adaptive supnorm Penalty

Adaptive supnorm penalty shares the similar idea as adaptive L_1 penalty. Supnorm gives equal penalty to each variable, however, adaptive supnorm applies weighted-penalties to different variables. The important variables are given small penalties and the unimportant variables are given large penalties. In this way the important variables are protectively preserved in the model selection and unimportant variables shrink to zero faster and finally drop off from the model. The adaptive supnorm use the penalty form $\lambda \sum_{j=1}^{p} w_j \eta_j$, each variable has its own adjustable penalty $w_j = \frac{1}{\eta_j{}'}$, where $\eta_j{}' = \max_{k=1,\ldots,K} |\beta'_{kj}|$ and β'_{kj} is solved using L_2-norm MSVM. If $\eta_j{}'$ is a large coefficient from L_2-norm MSVM, then w_j is

small and η_j is given a small penalty; and verse-vice. The new optimization problem with adaptive supnorm penalty has the same constraints as the supnorm, but different objective function as the following

$$
\begin{aligned}
\min_{\beta,\beta_0,\xi} \quad & \sum_{i=1}^{m}\xi_i + \frac{1}{2}\lambda\sum_{j=1}^{p}\frac{\eta_j}{\eta'_j} \\
\text{subject to:} \quad & \sum_{j=1}^{p}(\beta^+_{yj}-\beta^-_{yj})x_{ij} - \sum_{j=1}^{p}(\beta^+_{kj}-\beta^-_{kj})x_{ij} \geq 1-\xi_i, \\
& \qquad\qquad \forall i, k = 1,\ldots,K, k \neq y, \\
& \eta_j \geq \beta^+_{kj} + \beta^-_{kj}, \forall k = 1,\ldots,K, j = 1,\ldots,p, \\
& \eta'_j \geq \beta'_{kj}, \forall k = 1,\ldots,K, j = 1,\ldots,p, \\
& \sum_{k=1}^{K}\beta_{k,0} = 0; \sum_{k=1}^{K}(\beta^+_{kj}-\beta^-_{kj}) = 0, \forall j = 1,\ldots,p, \\
& \beta^+_{kj} \geq 0, \beta^-_{kj} \geq 0, \eta_j \geq 0, \forall k = 1,\ldots,K, j = 1,\ldots,p, \\
& \xi_i \geq 0, \ \forall i = 1,\ldots,n.
\end{aligned}
$$

3.4 Parameter Tuning

λ is an important tuning parameter, which controls the trade off between training error and generalization, and determines the number of variables used in the classifier. To select the optimum tuning parameter λ, a V-fold cross validation is used to select λ via a grid search. A cross validation error given each $\log_2(\lambda)$ is calculated and used to identify the optimum λ which gives the best performance with the least cross validation error.

3.5 Simulation Study

In this section the author examined and compared of the prediction and variable selection performance of several new MSVMs with different variable selection penalties using the simulated data. Also comparison of the methods against the L2MSVM proposed by Crammer and Singer (2001) was done.

The simulation example is a four-class problem with $\mathbf{x}$ in 20 dimensions. For each class $k = 1, 2, 3, 4$, the first two dimensions of $\mathbf{x}$ are generated from a $N(\mu_k, \sigma^2 I_2)$, where $\mu_1 = (\sqrt{2}, \sqrt{3}), \mu_2 = (-\sqrt{3}, \sqrt{2}), \mu_3 = (-\sqrt{2}, -\sqrt{3}), \mu_3 = (\sqrt{3}, -\sqrt{2})$ and $\sigma = \sqrt{2}$. The remaining 18 dimensions of $\mathbf{x}$ are *i.i.d.*, generate from $N(0, 1)$. Evidently, only the first 2

Table 3.1: The average test error and model size for 100 simulations of linear example

Method	Test Error (S.D.)	Selected Variables: important 2 var.	Selected Variables: noise 18 var.	Model Size
L2MSVM	0.302(0.014)	2	18	20
L1MSVM	0.263(0.014)	2	4.29	6.29
SupMSVM	0.257(0.009)	2	2.71	4.71
Adapt-L1MSVM	0.260(0.008)	2	1.91	3.91
Adapt-SupMSVM	0.255(0.007)	2	1.24	3.24
Bayes	0.246(-)	2	0	2

C & S represents Crammer and Singer (2001)

components of x are relevant to classification, whereas the remaining 18 components are redundant. The optimum theoretical test error, called Bayes error, is 0.246 for this problem. 200 training samples, 200 tuning samples and 40,000 test samples were randomly simulated with equal sample size for each class. The tuning samples are used for choosing the optimum tuning parameter λ and the test samples are used for estimating the prediction error. λ is searched over a grid: $\log_2(\lambda) = -14 : 1 : 15$. This experiment is repeated for 100 times with 100 simulated data sets, the average test error and the average model size are summarized in Table 3.1. The number in the parentheses are the standard deviation of the estimates.

Table 3.1 compared the average test error and average model size for 100 simulations among the L2MSVM by Crammer and Singer (2001) and 4 new MSVM methods with 4 different variable selection penalties. Obviously, the L2MSVM by Crammer and Singer (2001) always include all variables in the model. However, all four methods can always select two important variables out and eliminate noise variables in different degrees. Adaptive-weighted penalties can achieve smaller average model size, i.e. more sparse model, than the equal-weighted penalties. In this example, supnorm penalty performs better than the L1 penalty in terms of variable selection and prediction accuracy. The MSVM with adaptive-supnorm penalty has the average model size, 3.24 variables, which is most close to the true model size, 2 variables. The error rate of the methods are also more close to the Bayes error rate than the L2MSVM by Crammer and Singer (2001) since models selected by the methods can retain the important variables but eliminate the noise variables which will hurt the prediction accuracy if included into the model. The MSVM with adaptive-supnorm penalty show the best prediction accuracy which may benefit from that the model size learned from the MSVM with adaptive-supnorm penalty is most close to the underline

true model.

Table 3.2 summarized the selection frequency of each variable in 100 simulations and compared the results from different methods. L2MSVM by Crammer and Singer (2001) always includes all 20 variables into the model for each simulations. For the methods, the two important variables x_1 and x_2 are always selected out and appear 100 times in 100 simulations, but the noise variables are selected less frequently. L1MSVM selects most noise variables around 20-30 times in 100 simulations where adaptive-supnorm MSVM performs best and selects most noise variables in less than 10 times from 100 simulations. Overall the MSVM with the adaptive-supnorm performs best in both classification accuracy and variable selection.

3.6 Real Data Application

One important application of the newly developed methods is classification and variable selection of microarray or "-omics" data. To evaluate the performance of the developed approaches, two well-known data sets were used for investigation. One was leukemia data set with 3 class levels (Guyon et al., 2002) and the other was small round blue cell tumor data set with 4 class levels (Khan et al., 2001). The L2-norm MSVM proposed by Crammer and Singer (2001) can only predict the cancer class label but does not automatically select relevant genes responsible for the classification. However sometimes, a primary goal in microarray or "-omics" study is to identify the genes responsible for the classification rather than class prediction since the selected small gene set provides a clue to the mechanism related to tumor and a guideline for biologist to perform additional confirmatory experiments. The microarray data has extreme high dimensional gene features, a popular way for genes selection is univariate ranking. The weakness for such ranking method is that (1) the classification and variable selection are performed in two-steps and (2) the correlation and interaction among genes are not considered when selecting genes for building classifier. The developed MSVM methods with variable selection has inherent gene selection property due to the penalty function. Hence, the developed MSVM methods can achieve the goals of classification of patients and selection of genes simultaneously. As stated previously, there are only three publications about variable selection in MSVM (Wang and Shen, 2007; Lee et al., 2006; Zhang et al., 2006). Wang and Shen (2007) showed their method performance on the the leukemia study. Lee et al. (2006) and Zhang et al.

Table 3.2: Variable selection frequency in 100 simulations of linear example

Method	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
L2MSVM(C & S)	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
L1MSVM	100	100	25	29	24	22	26	30	27	20	21	21	25	22	24	19	21	32	25	16
SupMSVM	100	100	16	15	18	15	18	13	20	14	13	17	13	16	9	10	16	18	15	15
Adapt-L1MSVM	100	100	7	9	11	11	13	13	11	12	10	8	13	10	10	10	15	12	11	5
Adapt-SupMSVM	100	100	7	10	11	6	10	7	7	8	5	4	6	4	5	5	9	4	8	8

C & S represents Crammer and Singer (2001)

(2006) evaluated their method performance using the small round blue cell tumor data set. Besides testing and comparing results among the newly developed four approaches, a comparison of the performance of the methods with the three existing MSVM with variable selection methods is done.

3.6.1 Leukemia Study

Leukemia study analyzed human bone marrow samples using oligonucleotide microarrays produced by Affymetrix. Leukemia data consists of 7219 genes and 72 patient bone marrow samples. The 72 patient bone marrow samples are categorized into three classes: acute myeloid leukemia (AML), T-cell acute lymphoblastic leukemia (ALL_T) and B-cell acute lymphoblastic leukemia (ALL_B). The data and sample information can be download from http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi. There are 38 training samples including 19 ALL_B samples, 8 ALL_T samples and 11 AML samples. The rest 34 samples are test data consisting of 19 ALL_B samples, 1 ALL_T samples and 14 AML samples. According to the description from Dudoit et al. (2002), all data including both training and testing samples are preprocessed by thresholding, filtering and log_{10}-transformation. Finally, it ends up with a data matrix with 3571 genes $\times$ 72 samples. For each gene, a one-way ANOVA is fitted and F-ratio test is performed on the training samples only. 742 out of 3571 genes are significant using the FDR 0.05. Only the significant 742 genes are used for model fitting because of limitation in computer memory. In model learning, cross-validation on the training data is used to identify the optimum λ. The cross-validation error is a little underestimated due to the selection of a small subset of significant genes using all training data. Fortunately, there is an independent test data, which can provide an unbiased estimation of prediction error. In the discussion section, I will talk about a stringent cross-validation approach which will provide an unbiased estimation of cross-validation error. However, the best λ selected from either way may not have much difference.

The original approach by Crammer and Singer (2001) and the four new approaches are applied to the training data matrix of 742 genes $\times$ 38 samples. The learned models are used to predict the unseen 34 test samples. The test error and gene selection results are summarized and compared in Table 3.3. From the Table 3.3, the L2-norm approach proposed by Crammer and Singer (2001) almost cannot select important variables out while

Table 3.3: Classification and variable selection results using top 742 genes in leukemia study

Penalty	Test Error	No. of Genes
L2MSVM (C & S)	1/34	429
L1MSVM	2/34	14
SupMSVM	2/34	14
Adapt-L1MSVM	3/34	9
Adapt-SupMSVM	3/34	4

[a] C & S represents Crammer and Singer (2001).

[b] Supnorm and L1 penalty achieves the same result for 3 class problem.

Table 3.4: The gene list selected using 4 new approaches in leukemia study

Label	Adapt-SupMSVM	Adapt-L1MSVM	SupMSVM /L1MSVM	F-test rank
X00437_s_at	1	1	1	1
X76223_s_at	1	1	1	3
M27891_at	1	1	1	12
X82240_rna1_at	1	1	1	19
X59871_at	-	1	1	8
M11722_at	-	1	1	157
U89922_s_at	-	1	1	324
Z14982_rna1_at	-	1	1	527
M21624_at	-	1	-	462
U05259_rna1_at	-	-	1	10
X58529_at	-	-	1	27
M74719_at	-	-	1	46
Y00787_s_at	-	-	1	58
M19507_at	-	-	1	112
U01317_cds4_at	-	-	1	390

MSVMs with Supnorm or L1 penalty has the same gene list for 3 class problem.

all four approaches can select a very small set of informative genes. It is a 3 class classification problem, hence the supnorm and L1-norm penalty lead to the same answer (Zhang et al., 2006). The new approaches misclassified 1-2 more samples than L2-norm approach by Crammer and Singer (2001) but greatly reduced the number of genes to accurately predict the classes. Those selected genes could be further used to explain the mechanism difference for different types of acute leukemia. The adaptive-weighted penalties can identify the smaller gene set than the equal weighted penalties. In Adaptive-supnorm penalty approach, a minimal number of informative genes ($n = 4$) are selected, which greatly saves the amount of time and costs for biologists to explore the further experimental confirmation.

In the Table 3.4, the gene lists selected from the four new developed approaches

are compared. In Table 3.4, the 4 genes selected from Adaptive-supnorm approach is a subset of any other gene lists from the other three approaches. Eight out of 9 selected genes using adaptive-L1 norm is also in the list of 14 genes selected from L1-norm/Supnorm methods. The big overlap among genes selected with different approaches demonstrated a reasonable stability in the marker selection. The last column shows the rank of genes using a univariate measurement, i.e. the F-test statistics. Apparently, not all selected genes are top rank genes since some high rank genes may be highly correlated. Different with the ranking method that select individual gene separately and ignore their correlation and interaction, the methods can select the smallest informative gene set using multivariate selection and considering the correlation among genes automatically.

To see how the selected genes serve as the legitimate markers for cancer classification, the model built by adaptive-supnorm approach are explained. The three discriminant functions learned by adaptive-supnorm correspond to three cancer classes, $f_{\text{ALL\_B}}, f_{\text{ALL\_T}}$ and f_{AML}.

$$\begin{aligned} f_{\text{ALL\_B}} &= -0.037 * \text{X00437\_s\_at} - 0.330 * \text{X76223\_s\_at} \\ &\quad -0.640 * \text{M27891\_at} + 0.091 * \text{X82240\_rna1\_at}; \\ f_{\text{ALL\_T}} &= 0.162 * \text{X00437\_s\_at} + 0.450 * \text{X76223\_s\_at}; \\ f_{\text{AML}} &= -0.124 * \text{X00437\_s\_at} - 0.121 * \text{X76223\_s\_at} \\ &\quad +0.640 * \text{M27891\_at} - 0.091 * \text{X82240\_rna1\_at}; \end{aligned}$$

The above three discriminant functions shows that (1) The first two genes X00437_s_at and X76223_s_at play major role in discriminating the class ALL_T from the other two classes, ALL_B and AML, since the coefficients of the first two genes for ALL_T class are positive while the coefficients of the first two genes for the other two classes, ALL_B and AML, are negative; (2) The last two genes M27891_at and X82240_rna1_at are further used for discriminating between ALL_B and AML since the coefficients of the last two genes for ALL_T class are zero. The last two genes play the same classification role from two different directions. The coefficient of M27891_at is negative for ALL_B, but positive for AML while the coefficient sign of X76223_s_at in ALL_B and AML classes are just opposite to the M27891_at. This result can be further confirmed by applying a hierarchical clustering to the expression data of the chosen four genes. Figure 3.1 is the visualization of clustering pattern. The red color represents the expression value is greater than the mean and the green

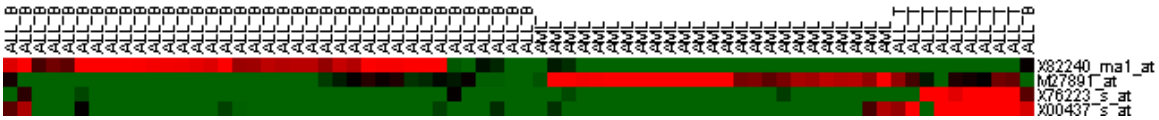


Figure 3.1: The hierarchical clustering result of all AML, ALL_T and ALL_B training and test samples based on 4 selected genes. Each column represents a sample and each row represents the expression profile of a gene across all samples. The expression value greater than the mean is colored in red and the expression value less than the mean is colored in green. The hierarchical clustering result is generated using the public software Cluster (http://rana.lbl.gov/EisenSoftware.htm) and viewed by the Java TreeView (http://jtreeview.sourceforge.net/)

color represents the expression value is less than the mean. Figure 3.1 illustrates the same pattern as what was inferred from the discriminating functions. The two genes, X00437_s_at and X76223_s_at, are colored in red for most ALL_T samples, but green for most ALL_B and AML samples. The M27891_at and X76223_s_at gene can further discriminate ALL_B and AML classes. The M27891_at shows low expression value in most ALL_B samples but high expression value in most AML samples, while the X76223_s_at shows the opposite pattern. In addition, Figure 3.1 shows that 4 selected genes can correctly clustering most samples except that one ALL_B sample is clustered together with a cluster of ALL_T sample.

The discriminant functions not only can explicitly illustrate each gene's responsibility in discriminating cancer classes but also can be used to infer the strength of prediction on each test sample. The Figure 3.2 shows the predicted decision value, $f_{\text{ALL\_B}}$, $f_{\text{ALL\_T}}$ and f_{AML}, for each test sample. Each discriminant function measures how strong a sample associated with each class. The classification rule is to assign a test data a class label which gives the highest decision value f. The bigger difference between the highest decision value with the second-highest decision value of a test sample may mean strong prediction away from the class boundary because the loss for the predicted class of this sample will be equal to zero or very small. Comparing the decision values of each test sample and its true class label in x-axis, 31 out of 34 samples are classified correctly while 3 out of 34 samples are misclassified.

Wang and Shen (2007) developed a L1MSVM method based on the loss function proposed by Lee et al. (2004). They applied their method to the leukemia study and compared the performance of their method with the one-versus-all and L2MSVM by Lee et al. (2004) on the same data set. To make the exactly equal comparison with L1MSVM

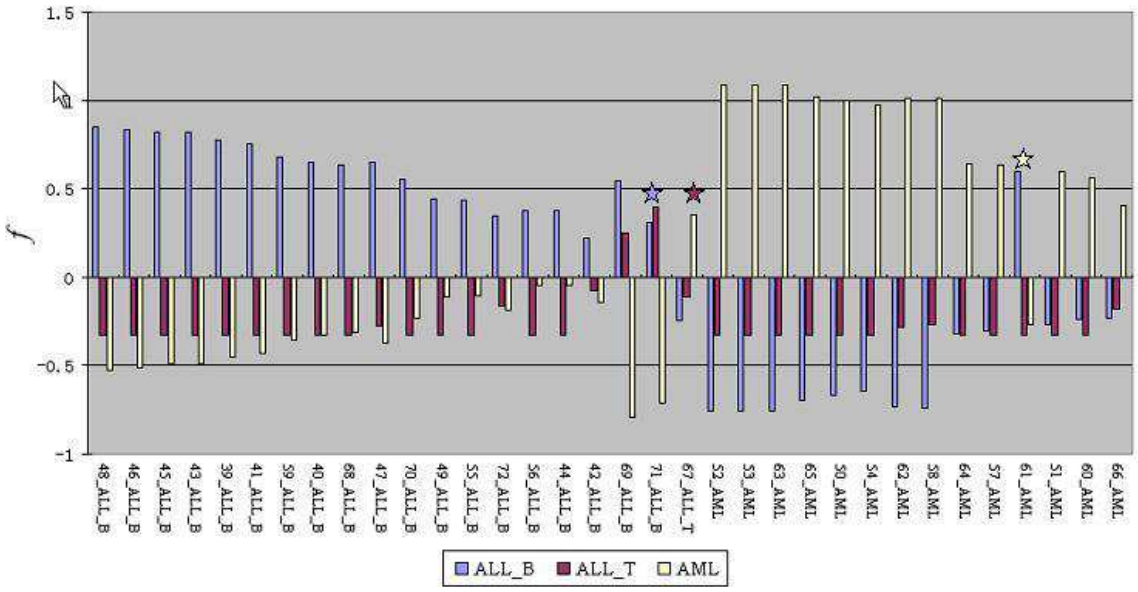


Figure 3.2: The predicted decision values, $f_{\text{ALL\_B}}, f_{\text{ALL\_T}}$ and f_{AML}, for each test sample. The $f_{\text{ALL\_B}}$ is in purple, the $f_{\text{ALL\_T}}$ is in red and f_{AML} is in yellow. The classification rule is to assign a sample a class label that gives the highest decision value f. The x-axis is the test sample index with true class label. Comparing the decision value with the true class label, 3 out of 34 test samples are misclassified, which is marked with star. The star are colored with the true class label.

Table 3.5: Leukemia data: Averaged test errors and their standard errors (in parentheses), as well as averaged number of genes selected and their standard error (in parentheses) over 100 runs. The results from this study are compared with those from Wang and Shen's study (2007)

Method	Penalty	Test Error (S.E.)	No. of Genes (S.E.)
Our	L2MSVM(C & S)	4.13% (0.03%)	291.88 (0.017)
Study	L1MSVM	5.88% (0.05%)	15.77 (0.024)
	SupMSVM	5.88% (0.05%)	15.77 (0.024)
	Adapt-L1MSVM	6.75% (0.04%)	11.98 (0.029)
	Adapt-SupMSVM	6.96% (0.04%)	11.18 (0.028)
Wang	L1MSVM	3.67% (0.51%)	20.92 (0.71)
& Shen	OVA	6.24% (0.45%)	26.73 (0.80)
(2007)	L2MSVM(Lee, 2004)	4.20% (0.33%)	3571

[a] C & S represents Crammer and Singer (2001).

[b] Test error rate is calculated from 24 test samples, 1 misclassified samples = 4.17% test error, 2 misclassified samples = 8.33% test error.

method by Wang and Shen (2007), the same procedure as Wang and Shen (2007) described in their paper was performed in this study: (1) Two-thirds of the 72 samples are randomly selected by stratification and the remaining one-third is used as test samples; (2) Five-fold cross-validation is used to select the best tuning parameter; (3) Repeat this process 100 times and calculate the average of the test error and the average number of selected genes over 100 data partitions. For each of the proposed methods, 100 data partitions were performed and learned 100 times by 5-fold cross-validation. To save the time, genes which p-value is greater than the Bonferroni 0.05 threshold in the F-ratio statistics were first filtered out. Therefore, total 295 genes are selected to fit the MSVM models developed in the study. Comparing the average prediction accuracy in 100 runs, the method proposed by Wang and Shen (2007) misclassify less than 1 sample in average (test error: 3.67%) since there are 24 test samples in each run where the methods misclassify less than 2 samples out of 24 samples in average (test error: 5.88%-6.96%)(Table 3.5). The methods show slightly larger test error than the method by Wang and Shen (2007). However, comparing the number of genes selected in average over 100 runs, the new methods can select much smaller number of genes with an ability to successfully classify samples. For biologists, smaller number of gene candidates can reduce large amount of works in the further experimental exploration. Compared with the result from the one-versus-all approach, the methods show a competitive prediction accuracy with much more compact model size.

3.6.2 Small Round Blue Cell Tumors Study

The small, round blue cell tumors (SRBCTs) of childhood can be categorized into 4 classes: neuroblastoma (NB), rhabdomyosarcoma (RMS), non- Hodgkin lymphoma (NHL) and the Ewing family of tumors (EWS) (Khan et al., 2001). Burkitt lymphoma (BL) is a subset of NHL. SRBCTs data set is a cDNA microarray data generated using standard protocol of National Human Genome Research Institute (NHGRI) (Khan et al., 2001). There are 63 training samples and 20 independent test samples. After filtering, the expression of 2308 genes out of 6567 genes are given at http://research.nhgri.nih.gov/microarray/ Supplement/. The data is log_2-transformed. For each gene, a one-way ANOVA is fitted and F-ratio test is performed on the training samples only. 333 out of 2308 genes are selected using the Bonferroni 0.05 threshold. To examine if the variable selection methods can eliminate the irrelevant noise genes, the bottom 300 genes were included as covariates

Table 3.6: Classification and variable selection results using 633 genes in SRBCTs study

Penalty	Test Error	Selected Genes: Top 333 genes	Selected Genes: Bottom 300 genes
L2MSVM (C & S)	0/20	194	124
L1MSVM	1/20	31	0
SupMSVM	0/20	36	0
Adapt-L1MSVM	0/20	31	0
Adapt-SupMSVM	0/20	28	0

C & S represents Crammer and Singer (2001).

besides the top 333 genes.

The original approach by Crammer and Singer (2001) and the four new different approaches were applied to the training data matrix of 633 genes $\times$ 63 training samples. The 633 genes include the top significant 333 genes and bottom non-significant 300 genes. The learned models were used to predict the unseen 20 test samples. Results are compared in the Table 3.6. All of the 4 new methods can exclude the non-significant 300 genes from the model. Comparing to the classification results of the same data from several MSVM variable selection methods in Zhang et al. (2006), their methods select around 47-62 genes out of 200 genes, including top 100 significant genes and bottom 100 non-significant genes. All of their methods misclassify 1 out of 20 test samples except that the method with adaptive-L1 penalty have 0 test error. However, the methods in this study select much smaller number of significant genes (28-36 genes) from larger number of covariates (633 genes) and all of the methods can achieve 0 test error except that the method with L1-norm penalty misclassify 1 out of 20 test samples. In summary, the presented methods outperform methods by Zhang et al. (2006) both in variable selection and prediction accuracy. Comparing to the classification results of the same data from the structured MSVM method by Lee et al. (2006), structured MSVM selects average 222 genes in 100 runs and achieves average 2.7% test error. Obviously the methods in this study outperform the structured MSVM in terms of selecting a minimal number of gene set which can achieve a higher prediction accuracy.

Looking into the 4 gene lists selected from the 4 different methods, there are big overlap among gene lists. Ten genes appear in all 4 gene lists and another 13 genes are selected at 3 out of 4 gene lists, which demonstrate again that the selected markers are reasonable stable and reliable. The adaptive-supnorm method selects the smallest number of genes comparing to other methods. Figure 3.3 shows the hierarchical clustering using

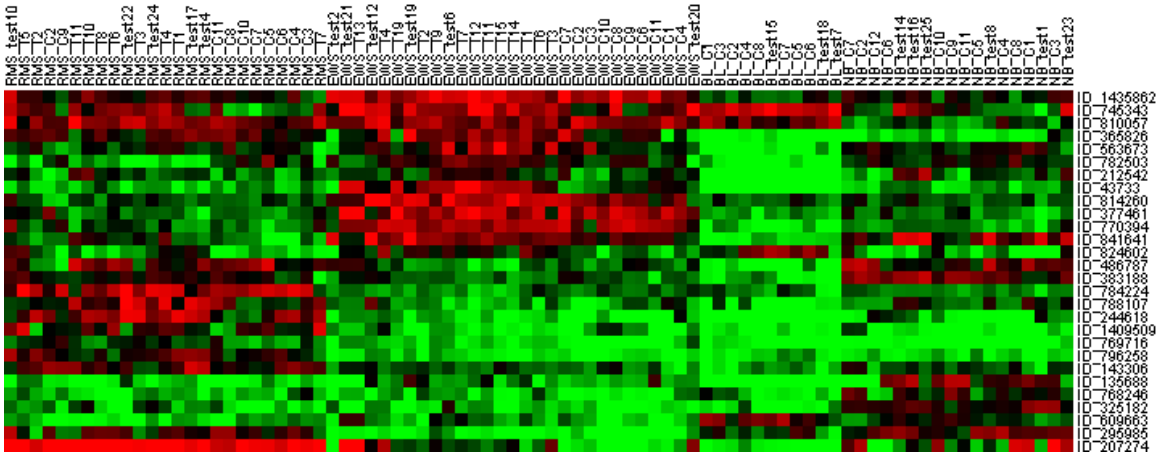


Figure 3.3: The hierarchical clustering result of all training and test samples based on 28 selected genes in SRBCTs study. Each column represents a sample from one of classes: NB, BL, RMS or EWS. Each row represents the expression profile of a gene across all samples. The log_2(relative red intensity) greater than the zero is colored in red and the log_2(relative red intensity) less than the zero is colored in green. The hierarchical clustering result is generated using the public software Cluster (http://rana.lbl.gov/EisenSoftware.htm) and viewed by the Java TreeView (http://jtreeview.sourceforge.net/)

28 genes selected from adaptive-supnorm penalty. It shows that the selected 28 genes can perfectly separate all training and test samples into 4 groups.

3.7 Discussion

The goal of this study is to provide an effective method for finding predictor genes that accurately discriminate multiple cancer (sub)types. The developed methods can carry the classification and variable selection simultaneously. The performance of the new methods has been demonstrated both in simulation study and real data application. In the simulation part, the new methods give better prediction accuracy and select more correct models than the L2MSVM by Crammer and Singer (2001). The simulated example also shows that the adaptive-weighted penalty functions can achieve more sparse and more correct models than the equal-weighted penalty functions and the adaptive-supnorm MSVM always outperforms the other three MSVM methods. The four methods were applied to two "large p small n" cancer data sets, Leukemia study and SRBCT study. In the real data, not only was a comparison of the result and performance among our four new approaches

done, but also a comparison of the performance of the methods with three existing MSVM methods which can also perform the variable selection simultaneously with learning the classification model. In Leukemia study, the genes selected by the four different MSVMs show big overlap. Further, the selected genes are not necessary all belonging to the top ranked genes since the top ranked genes may be highly correlated. Therefore, comparing to the univariate gene selection by ranking statistics, the methods can account for the correlation among genes and select the smallest number of genes which are able to achieve the optimum prediction accuracy. The learned discriminant functions explicitly illustrate the responsibility of each gene in discriminating several cancer subtypes. In addition, the discriminant functions can help to infer the strength of prediction on each test sample. Compared with the performance of L1MSVM by Wang and Shen (2007) on the leukemia study, the methods in this study can find a relatively smaller number of predictive genes which are able to achieve a good prediction accuracy. Compared with the one-versus-all performance on the leukemia study, the methods outperform in both prediction accuracy and variable selection. In SRBCT study, it is showed that the methods can select a very compact model including a small set of important genes and eliminating all noise genes. Compared with the performance of the MSVM methods by Zhang et al. (2006) and Lee et al. (2006) on the SRBCT study, it has been demonstrated again that the methods can select much smaller number of genes and achieve a 100% prediction accuracy at the same time. In summary, compared with the univariate gene ranking methods, the methods can account for the correlation among predictor genes; compared with the one-versus-all classification approach, the methods consider multiple classes at the same time and show better performance both in the prediction accuracy and variable selection; compared with the three existing MSVMs which can learn the classifier and perform the variable selection simultaneously, the methods can identify the smallest number of predictor genes which are able to achieve a high prediction accuracy. In addition, the methods can also be applied to binary classification problem (data not shown).

Compared with binary SVM, the MSVM methods need more memory since it estimates K decision functions by solving one big optimization problem. The size of the MSVM problem is bigger than that of solving a series of binary problems. This is a common shortcoming of the approaches that consider all the classes at the same time. To make the computation amenable to large data sets, a univariate statistics test is usually performed and the irrelevant and noise genes are eliminated before learning a classifier using MSVMs.

However, filtering genes before learning a classifier may provide an underestimate of cross-validation error since the gene filtering process make use of all training data. Fortunately, an independent test data set was used which can provide an unbiased estimation of prediction error. In order to provide an unbiased estimation of cross-validation error when there is no test data, gene filtering should be performed within each cross-validation. Therefore, one may train a model starting from different gene subsets in each fold of cross-validation. The optimum λ is the one that gives the smallest average cross-validation error. Then, perform the gene filtering again on all training data set and apply the identified best λ on it to learn a final classifier for prediction on the future data. The best λ selected from either way may not have much difference.

In summary, MSVMs with automatic variable selection features was developed by incorporating the sparse model regularization terms into the original MSVM formulation. The efficiency of selecting a relative small number of genes using the methods is very attractive for searching and identifying the diagnostic molecular marker. The combination of high accuracy and small number of genes should make the methods a powerful tool for cancer biomarker discovery and subsequent molecular cancer diagnosis.

Chapter 4

Variable Selection for Non-linear MSVM

4.1 Introduction

Sometimes in practice, data couldn't be linearly separated in the original input space. One cannot find an optimal hyperplane to separate the data in the original input space and a nonlinear curve is the optimum decision boundary to separate classes. In this situation, one can map the data from the original input space to higher dimensional "feature space", $\mathcal{H}$. In the higher dimensional "feature space" $\mathcal{H}$, the data may be completely separable or non-separable (overlapping), but the hyperplane is the optimum decision boundary at the "feature space". The mapping function is called $\Phi : \mathbf{R}^p \to \mathcal{H}$. One way for such mapping is to select some basis functions of $\mathcal{H}$: $h_m(\mathbf{x}), m = 1, \ldots, M$. Thus the new features after mapping, $h(\mathbf{x}_i) = (h_1(\mathbf{x}_i), \ldots, h_m(\mathbf{x}_i), \ldots, h_M(\mathbf{x}_i))$, are used to fit SVM model. The decision function for the separating hyperplane in the feature space is $\hat{f}(\mathbf{x}) = h(\mathbf{x})^T\hat{\beta} + \beta_0$, which is nonlinear for $\mathbf{x}$, but linear for $h(\mathbf{x})$. The classification rule is same as the linear SVM case, $\hat{y} = sign(\hat{f}(\mathbf{x}))$ for binary case or $\hat{y} = \arg\max_{k=1,\ldots,K} \hat{f}_k(\mathbf{x})$. The dimension of the feature space can be very large or as large as infinite.

$h(\mathbf{x})$ in the feature space can be treated as $\mathbf{x}$ in the linear SVM case, therefore, one can replace $\mathbf{x}$ by $h(\mathbf{x})$ in all the formulation that is derived for linear SVM case. The

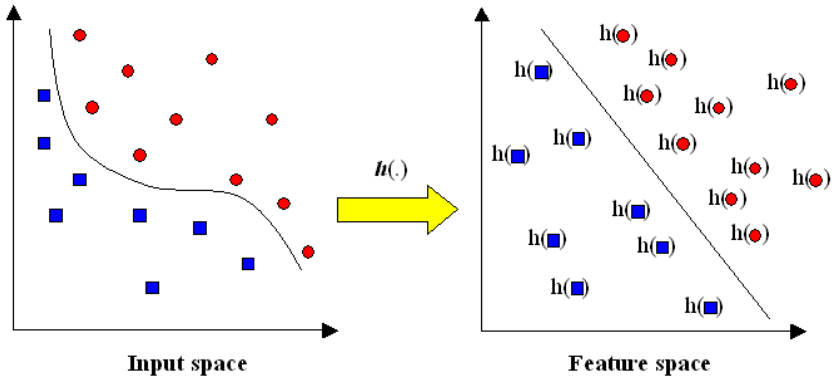


Figure 4.1: Feature mapping: transforming data from the original input space to a higher-dimensional feature space can make it linearly separable.

Lagrange dual problem is:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j < h(\mathbf{x}_i), h(\mathbf{x}_j) >, \tag{4.1}$$

$$\text{subject to:} \quad 0 \leq \alpha_i \leq \gamma, i = 1, \ldots, n,$$
$$\textstyle\sum_{i=1}^{n} \alpha_i y_i = 0.$$

$< h(\mathbf{x}_i), h(\mathbf{x}_j) >$ represents the inner product of $h(\mathbf{x}_i)$ and $h(\mathbf{x}_j)$. Therefore the decision function is $f(\mathbf{x}) = h(\mathbf{x})^T \beta + \beta_0 = \sum_{i=1}^{n} \alpha_i y_i < h(\mathbf{x}), h(\mathbf{x}_i) > + \beta_0$. Given α_i, β_0 can be determined by solving $f(\mathbf{x}_i) = 0$. In the Lagrange dual problem (4.1), $\alpha_i, i = 1, \ldots, n$ is the only variable and the number of variables α_i is the number of training points. Therefore, SVM can solve the nonlinear optimization problem and construct the optimal hyperplane in feature space $\mathcal{H}$ with the same calculation cost as that in original input space even if the original input data has been mapped to a infinity feature space $\mathcal{H}$.

The point of the above discussion is that both the Lagrange dual problem L_D and the decision function $f(\mathbf{x})$ only depend on the data through the dot product of $< h(\mathbf{x}), h(\mathbf{x}') >$ in the feature space. Therefore, if there were a "kernel function" K such that $K(\mathbf{x}, \mathbf{x}') =< h(\mathbf{x}), h(\mathbf{x}') >$, one only needs to use K without knowing the explicit mapping function $h(\mathbf{x})$. In addition, another advantage of introducing such kernel trick is the efficiency in the computational cost.

Not every function can be used as the kernel function. $K(\mathbf{x}, \mathbf{x}')$ must be a sym-

metric positive (semi-) definite function. There are many types of kernel functions, which are symmetric positive semi-definite. For examples:

$$
\begin{aligned}
d\text{th Degree polynomial:} &\quad K(\mathbf{x},\mathbf{x}') = (1+ <\mathbf{x},\mathbf{x}'>)^d \\
\text{Gaussian Radial basis:} &\quad K(\mathbf{x},\mathbf{x}') = exp(\frac{-\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}) \\
\text{Exponential Radial basis:} &\quad K(\mathbf{x},\mathbf{x}') = exp(\frac{-\|\mathbf{x}-\mathbf{x}'\|}{2\sigma^2}) \\
\text{Neural network:} &\quad K(\mathbf{x},\mathbf{x}') = tanh(\kappa_1 <\mathbf{x},\mathbf{x}'> +\kappa_2) \\
\text{Multiquadric:} &\quad K(\mathbf{x},\mathbf{x}') = (\|\mathbf{x}-\mathbf{x}'\|^2 + c^2)^{\frac{1}{2}} \\
\text{Inverse Multiquadric:} &\quad K(\mathbf{x},\mathbf{x}') = (\|\mathbf{x}-\mathbf{x}'\|^2 + c^2)^{-\frac{1}{2}} \\
\text{Wave kernel:} &\quad K(\mathbf{x},\mathbf{x}') = \frac{\theta}{\|\mathbf{x}-\mathbf{x}'\|}\sin\big(-\frac{\|\mathbf{x}-\mathbf{x}'\|}{\theta}\big) \\
\text{Rational quadratic kernel:} &\quad K(\mathbf{x},\mathbf{x}') = 1 - \frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\|\mathbf{x}-\mathbf{x}'\|^2 + \theta}
\end{aligned}
$$

Each kernel function corresponds to a unique mapping function $h(\mathbf{x})$. For example, a training sample has 2 dimensions in the original input space (x_1, x_2). A 2^{nd} polynomial kernel can be applied,

$$
\begin{aligned}
K(\mathbf{x},\mathbf{x}') &= (1+ <\mathbf{x},\mathbf{x}'>)^2 \\
&= (1 + x_1x_1' + x_2x_2')^2 \\
&= 1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_1'x_2x_2'
\end{aligned}
$$

Thus, it can be derived that the mapping function, $h(\mathbf{x}) = (h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_M(\mathbf{x}))$, has $M = 6$ dimensions in the feature space. Each dimension in the feature is a transformation of the original input space: $h_1(\mathbf{x}) = 1, h_2(\mathbf{x}) = \sqrt{2}x_1, h_3(\mathbf{x}) = \sqrt{2}x_2, h_4(\mathbf{x}) = x_1^2, h_5(\mathbf{x}) = x_2^2, h_6(\mathbf{x}) = \sqrt{2}x_1x_2$. $< h(\mathbf{x}), h(\mathbf{x}') >$ with the above mapping is equivalent to a 2^{th} polynomial kernel $K(\mathbf{x},\mathbf{x}')$.

A large γ puts more penalty on the misclassification and makes the model overfitting to the training samples. Furthermore, it will have a small margin width in the "feature space" and gives a wiggly nonlinear boundary in the original space. Small γ puts more penalty on the margin width and results in a model with larger margin in the "feature space". The model with larger margin in the feature space will lead to a smoother boundary in the original space.

However, the kernel function is the dot product of $< h(\mathbf{x}), h(\mathbf{x}') >$, which is the summation of the data from all dimensions. Therefore, it is hard to provide the relation-

ship between covariates and classes. With the kernel trick, one doesn't need to know the mapping function explicitly. Thus, the resulting classifier is like a "black box" function which does not allow for a clear interpretation of the importance of each variable to the final classifier (Lee et al., 2006). A lot of the variable selection research has been done for linear SVMs, which has been discussed in Chapter 3. For nonlinear SVMs using the kernel trick, the variable selection is a challenging area. Weston et al. (2001) and Chapelle et al. (2002) suggested to introduce a scale factor for each dimension. They treated the scale factor as tuning parameters and chose them by minimizing generalization error bounds via gradient descent (Weston et al., 2001; Chapelle et al., 2002). Grandvalet and Canu (2003) further added constraints to encourage the sparsity of scale factors and assign zero weights to irrelevant variables. Lee et al. (2006) proposed a structured learning through analysis of variance decomposition. Although Wahba et al. (1994) proposed structure learning for soft classification via smoothing-spline analysis of variance, Lee et al. (2006) first used ANOVA decomposition in the "hard" classification. The "soft" classification methods classify samples through estimating the probability of a sample belonging to each class. Logistic regression belongs to "soft" classification but SVM belongs to "hard" classification since SVM assigns classes without directly estimating the probability.

In this chapter, the author investigates variable selection for two different situations, non-linear MSVM via basis function transformation or non-linear MSVM via kernel functions. If the basis function is known, the approaches developed in Chapter 3 for linear MSVM can be directly extended for non-linear MSVM. If the basis function is unknown and kernel function is used for mapping data to higher dimensional feature space, the author developed a new MSVM method which can perform ANOVA decomposition on the kernel, construct one kernel for each dimension and learn a scaling factor for each dimension. To achieve the model sparsity, as suggested by Lee et al. (2006), one extra L1-penalty term is added to penalize the scale factor so that the irrelevant/nonimportant variables have the zero weights.

4.2 Non-linear MSVM via Basis Function Transformation

4.2.1 Method

The approaches developed in Chapter 3 can be easily extended to the non-linear MSVMs if the mapping functions, i.e. the basis functions, are known. Suppose the $\mathbf{h}(\mathbf{x}) = \{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_q(\mathbf{x})\}$ is a dictionary of basis functions transformed from original input variable $\mathbf{x}$. Therefore, the decision function is $f_k(\mathbf{x}_i) = \sum_{j=1}^{q} \mathbf{h}_j(\mathbf{x}_i)\beta_{kj} + \beta_{k0}$ where $j = 1, \ldots, q$ in non-linear case. The above decision function is non-linear in terms of $\mathbf{x}$, but linear in terms of $h(\mathbf{x})$. The design matrix is $\mathbf{H} = \Big(h_j(\mathbf{x}_i)\Big)_{n\times q}$ instead of $\mathbf{X} = (x_{ij})_{n\times p}$. Therefore, one can treat $(\mathbf{h}(\mathbf{x}))_{1\times q}$ the same as $\mathbf{x}_{1\times p}$ and plug in $\mathbf{h}(\mathbf{x})$ into the four regularization forms developed in Chapter 3 for non-linear case, however, variables for selection are $\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_q(\mathbf{x})\}_{1\times q}$ instead of $\mathbf{x}_{1\times p}$. Even if the basis function is unknown, one can include the quadratic or interaction terms into the MSVMs developed in Chapter 3 to capture some non-linear effect such as interaction effect or quadratic effect.

4.2.2 Simulation Study

A data set was simulated with a "large p small n" three-class example for the non-linear case. Then two covariates $\mathbf{x} = (x_1, x_2)$ were generated, which are uniformly distributed on the unit square $[-3, 3] \times [-3, 3]$. Then three functions were defined:

$$\begin{aligned} f_1 &= -2x_1 + 0.2x_1^2 - 0.4x_2^2 + 0.2; \\ f_2 &= -0.4x_1^2 + 0.8x_2^2 - 0.4; \\ f_3 &= 2x1 + 0.2x_1^2 - 0.4x_2^2 + 0.2; \end{aligned}$$

For each observation $\mathbf{x}$, the class label is given using the multinomial sampling of $(p_1(\mathbf{x}), p_2(\mathbf{x}), p_3(\mathbf{x}))$ with $p_k(\mathbf{x}) \propto f_k(\mathbf{x}), k = 1, 2, 3$. Evidently the classification is defined by x_1, x_1^2 and x_2^2. Another 18 *i.i.d.* covariates were generated from $N(0, 2)$. To achieve the non-linear classification, a fit to the non-linear MSVM was done by including 20 main effects, their quadratic effects and their 2-way interaction effects as the basis functions, which lead to total 230 variable terms in the model. Then, 100 training samples, 100 tuning samples and 40000 test samples were simulated. Similarly to the linear case, λ is searched over the grid of $\log_2(\lambda) = -14 : 1 : 15$ and this experiment was repeated 100 times with 100 simulated data sets. The average test error and the average model size for 100 simulations are summarized

Table 4.1: The average test error and model size for 100 simulations of non-linear example

Method	Test Error (S.D.)	Selected Variables: important 3 var.	Selected Variables: noise 227 var.	Model Size
L2MSVM(C & S)	0.441(0.021)	3	218.87	221.87
L1/SupMSVM	0.160(0.024)	3	18.34	21.34
Adapt-L1MSVM	0.152(0.021)	2.98	6.08	9.06
Adapt-SupMSVM	0.147(0.019)	3	5.58	8.58
Bayes	0.120(-)	3	0	3

C & S represents Crammer and Singer (2001)

in Table 4.1.

Proved by Zhang et al. (2006), L1MSVM and Supnorm MSVM have the same classification and variable selection result when the class number is equal to three. Therefore, in this study the result of L1MSVM and SupnormMSVM are presented together in Table 4.1. From Table 4.1, the L2MSVM by Crammer and Singer (2001) includes almost all variables in the model for each simulation since the average model size of 100 simulations is 221.87 out of 230 total variables. However, the four new methods achieve much smaller average model size. In 100 simulations, L1/Supnorm MSVM and Adaptive-supnorm MSVM select all 3 important variables 100 times in 100 runs and Adpative-L1 MSVM averagely select 2.98 out of 3 important variables in 100 runs. The L1/Supnorm MSVM select average 18.34 out of 227 non-important variables in 100 simulations. The MSVM with Adaptive-weight penalties performed better and select average 5-7 out of 227 non-important variables in 100 simulations. The L2MSVM by Crammer and Singer (2001) has the prediction accuracy close to 0.5, which is probably due to the number of non-important variables (227) that is over-dominant comparing to the number of important variables (3). However, the prediction accuracy of the method is much better than the L2MSVM, especially that the Adaptive-Supnorm MSVM has the average test error 0.147, which is very close to the Bayes error rate of this problem (0.120). Table 4.2 summarized the selection frequency of each variable in 100 simulations using Adaptive-Supnorm MSVM. Here the frequency tables of other methods (Adaptive-L1MSVM and L1/Supnorm MSVM) were skipped due to the limitation of pages. There are total 230 variable terms in the models including 20 main effects which frequencies are presented as the first row in Table 4.2, 20 quadratic terms which frequencies are presented on the diagonal in Table 4.2 and 190 two-way interaction effect which frequencies are presented in the lower triangle in Table 4.2 with each frequency repre-

senting the interaction of the 2 variables from the corresponding row and column. The three important variables, x_1, x_1^2 and x_2^2, are selected 100 times in 100 simulations. Obviously, the non-important variables are selected with much few times over 100 simulations. In this example, the Adaptive-Supnorm MSVM selected the smallest number of variables which leads to the smallest prediction error.

4.2.3 Application to the SRBCTs Study

The data set from the small round blue cell tumors study is used to test the performance of non-linear extension of linear MSVMs. As the author described it in Chapter 3, small round blue cell tumor study is a four-class classification problem with 63 training samples and 20 independent testing samples. After necessary preprocessing and filtering, the expression of 2,308 genes out of 6,567 genes are given at http://research.nhgri.nih.gov/microarray/Supplement/. First, fit the ANOVA for each gene and use the F-test as a measure of marginal association between each gene with the class label in the training data only to generate the p-value. Second, 333 out of 2308 genes are selected using the Bonferroni 0.05 as the p-value cutoff threshold. The selected top 333 genes and their quadratic effect are used to learn the model, thus, total 666 variables are fitted into the model. The learned model are used to predict the unseen 20 test samples. Result are compared in the Table 4.3. All 4 methods can achieve zero test error and select much smaller gene set at the same time. The number of selected genes is fewer than the number of selected variables since both main effect and quadratic effect are important for some genes, thus, both the main effect and quadratic effect terms for these genes are selected out. The 4 selected variable lists from 4 methods show a big overlap. Nine variables appear in all 4 variable lists, 10 variables are selected at 3 out of 4 variable lists and 22 variables are appear at 2 out of 4 variable lists.

Compared with the result in Chapter 3 Table 3.6, the number of genes selected by adaptive-L1 MSVM and adaptive-supnorm MSVM are about same no matter the quadratic term is included into the model or not. However, the number of genes selected by L1 MSVM and supnorm MSVM in Table 4.3 is larger than number of genes selected in Chapter 3 without quadratic terms. Gene lists selected in Chapter 3 Table 3.6 and in Table 4.3 also show a big overlap in selected genes. Adaptive-supnorm MSVMs selected 28 genes in linear case and 29 genes in non-linear case. There are 12 overlap genes. Adaptive-L1 MSVMs

Table 4.2: The variable selection frequencies in 100 simulations of non-linear example using MSVM with Adaptive-supnorm penalty

Variable	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
main effect	**100**	1	0	2	0	3	0	1	0	0	1	0	2	2	2	0	1	0	0	1
x_1	**100**	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
x_2	9	**100**	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
x_3	0	5	6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
x_4	1	4	3	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
x_5	4	3	3	1	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
x_6	1	1	1	2	1	7	.	.	.	.	.	.	.	.	.	.	.	.	.	.
x_7	3	1	1	4	2	3	5	.	.	.	.	.	.	.	.	.	.	.	.	.
x_8	3	4	5	2	2	0	0	5	.	.	.	.	.	.	.	.	.	.	.	.
x_9	2	6	2	2	1	2	2	2	6	.	.	.	.	.	.	.	.	.	.	.
x_{10}	1	4	4	2	4	2	1	3	2	7	.	.	.	.	.	.	.	.	.	.
x_{11}	1	8	1	4	2	2	1	4	1	1	7	.	.	.	.	.	.	.	.	.
x_{12}	2	3	3	1	3	0	4	2	2	2	1	9	.	.	.	.	.	.	.	.
x_{13}	2	5	2	0	3	2	2	4	2	3	0	4	9	.	.	.	.	.	.	.
x_{14}	2	3	3	2	1	5	1	4	1	4	3	3	3	5	.	.	.	.	.	.
x_{15}	2	2	3	0	6	3	1	4	3	3	3	1	2	1	5	.	.	.	.	.
x_{16}	0	6	0	5	4	2	0	4	2	4	0	2	4	0	3	4	.	.	.	.
x_{17}	1	5	0	2	1	4	2	2	2	1	3	1	1	1	3	2	4	.	.	.
x_{18}	0	4	3	2	1	0	2	3	2	2	4	3	3	1	2	0	2	4	.	.
x_{19}	0	4	3	4	6	3	3	0	1	3	1	2	2	3	0	1	3	2	9	.
x_{20}	1	3	2	4	3	1	3	3	3	0	0	0	3	1	2	0	2	1	1	6

[a] The diagonal is the quadratic effect.

[b] The lower triangle is the 2-way interaction effect of the variables from the corresponding row and column.

Table 4.3: Classification and variable selection results using 666 genes in SRBCTs study

Penalty	Test Error	# Selected Variables (666)	# Selected genes (333)
L2MSVM (C & S)	1/20	325	241
L1MSVM	0/20	63	59
SupMSVM	0/20	58	56
Adapt-L1MSVM	0/20	31	30
Adapt-SupMSVM	0/20	29	29

C & S represents Crammer and Singer (2001).

selected 31 genes in linear case and 30 genes in non-linear case. There are 16 genes overlap genes. Supnorm MSVMs selected 36 genes in linear case but 56 genes in nonlinear case. There are 20 genes overlap. L1 MSVMs selected 31 genes in linear case but 59 genes in non-linear case. There are 13 genes overlap. Therefore, it demonstrates again that the selected markers are reasonable stable and reliable.

4.3 Non-linear MSVM via Kernel Function

4.3.1 Method

If the mapping function is unknown, the kernel function will be used to map data from the low dimensional input space to higher dimensional feature space for better separation. Given a training data set with n pairs observations $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, which are *i.i.d.* from $P(\mathbf{x}, y)$, where $\mathbf{x} \in R^p$ and $y = \{1, 2, \ldots, K\}$, the general MSVM proposed by Crammer and Singer (2001) can be formulated with a kernel function, as the following:

$$
\begin{aligned}
\min_{b,\mathbf{c}_j,\xi_i} \quad & \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2} \sum_{j=1}^{k} (\mathbf{c}_j)^T \cdot K(\mathbf{X}, \mathbf{X}) \cdot \mathbf{c}_j \\
\text{subject to:} \quad & \forall i, r, \quad f_y(\mathbf{x}_i) - \max_{r \neq y} f_r(\mathbf{x}_i) \geq 1 - \xi_i, \\
& \sum_{j=1}^{k} f_j(\mathbf{x}) = 0, \\
& \forall i, \quad \xi_i \geq 0, \\
\text{where:} \quad f_j(\mathbf{x}) \quad & = b_j + \sum_{i=1}^{n} (\mathbf{c}_j)^T K(\mathbf{x}_i, \mathbf{x}), \\
K(\mathbf{X}, \mathbf{X}) \quad & = \Big(K(\mathbf{x}_i, \mathbf{x}_j) \Big)_{ij}, \forall i = 1, \ldots, n, j = 1, \ldots, n,
\end{aligned}
$$

$\mathbf{c}_j$ is a vector with each element equal to $\alpha_i y_i, \forall i = 1, \ldots, n$. As the author discussed in the Chapter 1, α_i is a Lagrange multiplier introduced for each sample i and only samples with $\alpha_i > 0$ are called "support vectors".

First, here is a brief introduction to the ANOVA decomposition of a function. Assume there is a $f(\mathbf{x})$ representing the relationship between the response variable y and p covariates $\mathbf{x} = (x_1, x_2, \ldots, x_p)$. The ANOVA decomposition of function $f(\mathbf{x})$ can be expressed as:

$$f(\mathbf{x}) = b + \sum_{\alpha=1}^{p} f_\alpha(x_\alpha) + \sum_{\alpha<\beta} f_{\alpha\beta}(x_\alpha, x_\beta) + ...$$

where b is a constant, f_α can be interpreted as the main effect of x_α and $f_{\alpha\beta}$ can be interpreted as the two-factor interaction effect of x_α and x_β, etc. For simplicity and interpretability, the higher order interaction effect can be ignored. Even if the higher order interaction effect is included, it probably will be very hard to estimate all functional components.

Kernel is a function defined in the reproducing kernel Hilbert space. Similar to the functional ANOVA decomposition that the author discussed above, the reproducing kernel Hilbert space can be decomposed and kernel function in the space will be decomposed correspondingly. The detailed discussion about the decomposition of reproducing kernel Hilbert space can be found at Lee et al. (2006). For simplicity and elucidation, all interaction components were ignored in the model, thus the original kernel function will be decomposed to an additive model format as the following.

$$K(\mathbf{x}_i, \mathbf{x}_j) = 1 + \sum_{v=1}^{p} K_v(x_{iv}, x_{jv})$$

where K_v represents a kernel constructed with only x_v covariate. A scale factor θ_v is introduced to give different weights to different component kernel spaces. Thus,

$$K(\mathbf{x}_i, \mathbf{x}_j) = 1 + \sum_{v=1}^{p} \theta_v K_v(x_{iv}, x_{jv})$$

Therefore, the MSVM with structured decomposition is

$$\begin{aligned}
\min_{b,\mathbf{c}_j,\xi_i} \quad & \sum_{i=1}^{n}\xi_i + \frac{\lambda}{2}\sum_{j=1}^{k}\big(\mathbf{c}_j\big)^T\cdot\big(\sum_{v=1}^{p}\theta_v K_v(\mathbf{X},\mathbf{X})\big)\cdot\mathbf{c}_j \\
\text{subject to:} \quad & \forall i,r, \quad f_y(\mathbf{x}_i) - \max_{r\neq y} f_r(\mathbf{x}_i) \geq 1-\xi_i, \\
& \sum_{j=1}^{k} f_j(\mathbf{x}) = 0, \\
& \forall i, \quad \xi_i \geq 0, \\
& \theta_v \geq 0, v = 1,\ldots,p, \\
\text{where:} \quad f_j(\mathbf{x}) \quad & = b_j + (\mathbf{c}_j)^T\sum_{v=1}^{p}\theta_v K_v(\mathbf{X},\mathbf{x}).
\end{aligned}$$

To make the weight (scale factor) of non-important component to be zero, a L1-penalty on the sum of all weights is introduced so that some small/negligibal weights are forced to be zero. Therefore the final training problem can be formulated as

$$\begin{aligned}
\min_{b,\mathbf{c}_j,\xi_i} \quad & \sum_{i=1}^{n}\xi_i + \frac{\lambda}{2}\sum_{j=1}^{k}\big(\mathbf{c}_j\big)^T\cdot\big(\sum_{v=1}^{p}\theta_v K_v(\mathbf{X},\mathbf{X})\big)\cdot\mathbf{c}_j + \lambda_\theta\theta_v \qquad (4.2) \\
\text{subject to:} \quad & \forall i,r, \quad f_r(\mathbf{x}_i) - f_y(\mathbf{x}_i) - \xi_i \leq \delta_{y,r} - 1 \\
& \sum_{j=1}^{k} f_j(\mathbf{x}) = 0 \\
& \theta_v \geq 0, v = 1,\ldots,p, \\
& \forall i, \quad \xi_i \geq 0 \\
\text{where:} \quad f_j(\mathbf{x}) \quad & = b_j + (\mathbf{c}_j)^T\sum_{v=1}^{p}\theta_v K_v(\mathbf{X},\mathbf{x})
\end{aligned}$$

4.3.2 Algorithm

Structured Kernel Construction

A radial basis function (RBF) kernel, $K(\mathbf{x},\mathbf{x}') = exp(\frac{-\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2})$ is used as an example to explain the difference between the general kernel and structured Kernel.

General Kernel: A general kernel uses all d dimensions to construct the kernel. (1) For two observations (i.e. two vectors) $\mathbf{x}$ and $\mathbf{x}'$, kernel $K(\mathbf{x},\mathbf{x}')$ is a scaler, $K(\mathbf{x},\mathbf{x}') = \|\mathbf{x}-\mathbf{x}'\|^2 = (x_1-x_1')^2 + (x_2-x_2')^2 + \ldots + (x_p-x_p')^2$. (2) For n observations (i.e. a matrix with $n\times p$ dimensions) $\mathbf{X}$ and m observations $\mathbf{X}'$, kernel $K(\mathbf{X},\mathbf{X}')$ is a matrix of dimension

$n \times m$ with each element (i, j) is a kernel for i^{th} observation in $\mathbf{X}$ and j^{th} observation in $\mathbf{X}'$.

Structured Kernel: A structured kernel uses each dimension independently to construct the kernel. (1) For two observations (i.e. two vectors) $\mathbf{x}$ and $\mathbf{x}'$, $K_v(\mathbf{x}, \mathbf{x}')$ is a scaler, $K_v(\mathbf{x}, \mathbf{x}') = \|\mathbf{x}_v - \mathbf{x}'_v\|^2 = (x_v - x'_v)^2$. There are p $K_v(\mathbf{x}, \mathbf{x}')$, so the ANOVA Kernel has dimension of $[1, 1, p]$. (2) For n observations (i.e. a matrix with $n \times p$ dimensions) $\mathbf{X}$ and m observations $\mathbf{X}'$, kernel $K_v(\mathbf{X}, \mathbf{X}')$ is a matrix with dimension of $n \times m$ with each element (i, j) is a kernel for i^{th} observation v^{th} dimension in $\mathbf{X}$ and j^{th} observation v^{th} dimension in $\mathbf{X}'$. There are p $K_v(\mathbf{X}, \mathbf{X}')$, thus the ANOVA Kernel has dimension of $[n, m, p]$.

Iterative One-way Grid Search

Different from the linear MSVM discussed in Chapter 3, optimization formulation (4.2) has two tuning parameters, λ and λ_θ, instead of one parameter. The two dimensional search usually has high computational cost. Fortunately, a two-dimensional search can be decomposed as iteratively updating processes, which is suggested by Lee et al. (2006). First initialize all $\theta_v = 1, v = 1, \ldots, p$, every component of kernel space is given a same weight. λ is the only parameter need be tuned. One can find the best λ which minimizes the average cross-validation error. Given the best λ, the b_j and $\mathbf{c}_j$ can be solved for $\forall j = 1, \ldots, K$. The step of tuning the λ and finding the solution for b_j and $\mathbf{c}_j$ is called c-step. Then, the λ_θ is tuned at the specified best λ given the b_j and $\mathbf{c}_j$. The λ_θ can identify the scale parameter $\theta_v, v = 1, \ldots, p$ and shrink some small scale factor to zero. This step is called θ-step. The θ-step is used to identify the scale factor and give different weights to different components of kernel space. After obtaining the solution of the scale factor θ_v, one needs to rerun the c-step to tune the λ at the updated θ_v value and get the new solution for b_j and $\mathbf{c}_j$ at the updated θ_v value.

The problem (4.2) can be solved in three steps, which can be summarized as the follows:

1. C-step for initialization: The first c-step uses the initialized values of $\theta = (1, \ldots, 1)$, which give the equal weight to every component of kernel space. In c-step, one

doesn't consider the penalty term $\lambda_\theta \theta_v$. Thus, the optimization problem is:

$$\begin{aligned}
\min_{b,\mathbf{c}_j,\xi_i} \quad & \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2} \sum_{j=1}^{k} (\mathbf{c}_j)^T \cdot (\sum_{v=1}^{d} \theta_v K_v(\mathbf{X}, \mathbf{X})) \cdot \mathbf{c}_j \\
\text{subject to:} \quad & \forall i, r, \quad f_r(\mathbf{x}_i) - f_y(\mathbf{x}_i) - \xi_i \leq \delta_{y,r} - 1 \\
& \sum_{j=1}^{k} f_j(\mathbf{x}) = 0 \\
& \theta_v \geq 0, v = 1, \ldots, p, \\
& \forall i, \quad \xi_i \geq 0 \\
\text{where:} \quad f_j(\mathbf{x}) & = b_j + (\mathbf{c}_j)^T \sum_{v=1}^{d} \theta_v K_v(\mathbf{X}, \mathbf{x})
\end{aligned}$$

λ is the only parameter tuned via minimizing the cross-validation error. After the optimum $\lambda^{(0)}$, which has the minimum cross-validation error, is identified, the $(\hat{b}_j^{(0)}, \hat{\mathbf{c}}_j^{(0)})$ for each class $j = 1, \ldots, K$ can be solved.

2. θ-step: The θ-step is to tune λ_θ at the specified $\lambda^{(0)}, \hat{b}_j^{(0)}, \hat{\mathbf{c}}_j^{(0)}, j = 1, \ldots, K$. The goal is to find the optimum λ_θ and the solution for $\theta_v, v = 1, \ldots, p$. It is worthwhile noting that, when one does the cross-validation, one shouldn't directly use the $\hat{b}_j^{(0)}, \hat{\mathbf{c}}_j^{(0)}$ obtained from step 1 since $\hat{b}_j^{(0)}, \hat{\mathbf{c}}_j^{(0)}$ in step 1 are solved using all training data. Therefore, in θ-step cross validation procedure, one needs to first run the c-step at the optimum $\lambda^{(0)}$ to get the $\hat{b}_j^{(0)}, \hat{\mathbf{c}}_j^{(0)}$ for each cross-validation sample; then run the θ-step at the $\lambda^{(0)}, \hat{b}_j^{(0)}, \hat{\mathbf{c}}_j^{(0)}$ for the same sample. The optimization problem can be formulated as the follows:

$$\begin{aligned}
& \min_{\theta_v,\xi_i} \quad \textstyle\sum_{i=1}^{n} \xi_i + \lambda_\theta \sum_{v=1}^{d} \theta_v + \frac{\lambda^{(0)}}{2} \sum_{v=1}^{d} \theta_v \sum_{j=1}^{k} (\mathbf{c}_j^{(0)})^T \cdot K_v(\mathbf{X}, \mathbf{X}) \cdot \mathbf{c}_j^{(0)} \\
\Longrightarrow \quad & \min_{\theta_v,\xi_i} \quad \textstyle\sum_{i=1}^{n} \xi_i + \sum_{v=1}^{d} \big(\lambda_\theta + \frac{\lambda^{(0)}}{2} \sum_{j=1}^{k} (\mathbf{c}_j^{(0)})^T \cdot K_v(\mathbf{X}, \mathbf{X}) \cdot \mathbf{c}_j^{(0)}\big) \theta_v \\
\text{subject to:} \quad & \forall i, r, \quad f_r(\mathbf{x}_i) - f_y(\mathbf{x}_i) - \xi_i \leq \delta_{y,r} - 1 \\
& \xi_i \geq 0 \quad for \;\; i = 1, \ldots, n \\
& \theta_v \geq 0 \quad for \;\; v = 1, \ldots, k \\
\text{where:} \quad & f_j(\mathbf{x}) = b_j + \sum_{v=1}^{d} \theta_v (\mathbf{c}_j)^T \cdot K_v(\mathbf{X}_v, \mathbf{x}_v)
\end{aligned}$$

After the optimum λ_θ, which shows the minimum cross-validation error, has been identified, the $\theta_v, v = 1, \ldots, p$ can be solved and different components of kernel space will be given different weights.

3. C-step for updating: This c-step re-tunes the λ at the values of $\theta_v, v = 1, \ldots, p$ which have been already obtained in the step 2. This c-step is the same as the c-step at step 1 except that θ_v is the weighting factor that one obtained in step 2 instead of $(1, \ldots, 1)$. After the best $\lambda^{(1)}$ is identified, the $(\hat{b}_j^{(1)}, \hat{\mathbf{c}}_j^{(1)})$ for each class $j = 1, \ldots, K$ will be updated.

In this iterative way, a two-dimensional scan/search for the optimum λ and λ_θ has been reduced to three one-dimensional searches. It greatly reduces the computation cost.

4.3.3 Simulation Study

In this section, a toy example was used to evaluate the performance of the developed variable selection method for non-linear MSVM with kernel function. It is a three-class example with two covariates $\mathbf{x} = (x_1, x_2)$. $\mathbf{x} = (x_1, x_2)$ are uniformly distributed on the unit square $[0, 1] \times [0, 1]$. Only x_1 is relevant to the response variable $y, y = 1, 2, 3)$. Then, three probability functions are defined:

$$\begin{aligned} p_1 &= 0.97exp(-3x_1); \\ p_3 &= exp\{(-2.5(x_1 - 1.2)^2\}; \\ p_2 &= 1 - p_1 - p_3; \end{aligned}$$

For each observation $\mathbf{x}$, the class label is given based on the multinomial sampling of $(p_1(\mathbf{x}), p_2(\mathbf{x}), p_3(\mathbf{x}))$. Evidently the class is only defined by x_1. The underline three target functions are three non-linear functions shown in Figure 4.2. The Bayes error rate for this example is 0.3941, which is a hard discrimination problem. Next, 50 runs were simulated, each with 100 training points and 200 test points. The λ is searched over a grid: $log_2(\lambda) = -10 : 1 : 10$ and the λ_θ is searched over a grid: $log_2(\lambda_\theta) = -15 : 1 : 10$.

The Table 4.4 reports the test error and the weights of x_1 and x_2 in 50 runs. The correct model can be learned for 34 times out of 50 runs. The average test error in 50 runs is 0.431 with the standard deviation as 0.043. The figure 4.3 shows the simulated data and the learned decision boundary. In summary, the newly developed method can perform the multi-class classification and variable selection simultaneously.

4.3.4 Application to the SRBCTs Study

One of the important application areas of the newly developed method is cancer classification and biomarker discovery. For better comparison, the small round blue cell

Table 4.4: Variable selection results for two dimensional three-class simulation example

Seed	θ_1	θ_2	Test Error	Seed	θ_1	θ_2	Test Error
1	0.75	0	0.37	26	0.61	0	0.505
2	0.62	0	0.445	27	0.55	0	0.435
3	0.93	0.79	0.45	28	0.5	0	0.43
4	0.57	0	0.42	29	0.56	0	0.45
5	0.73	0	0.545	30	0.9	0.7	0.405
6	1	0.87	0.34	31	0.99	0.97	0.515
7	0.47	0	0.44	32	1.09	1.14	0.405
8	0.94	0.64	0.495	33	0.81	0	0.445
9	1	1	0.44	34	1	1	0.36
10	0.71	0.05	0.46	35	1	1	0.45
11	0.74	0	0.455	36	0.79	0	0.385
12	1.03	0	0.47	37	1	1	0.445
13	1.01	0.99	0.395	38	1	0	0.425
14	0.92	0	0.44	39	0.81	0.62	0.4
15	2.7	3.46	0.44	40	0.85	0	0.43
16	0.86	0	0.4	41	0.78	0	0.385
17	0.55	0	0.425	42	0.66	0	0.385
18	0.84	0	0.415	43	0.92	0	0.395
19	0.79	0	0.44	44	0.88	0	0.425
20	0.76	0.09	0.465	45	0.72	0	0.38
21	0.92	0	0.445	46	0.83	0	0.35
22	0.94	0	0.445	47	0.64	0	0.435
23	0.79	0	0.41	48	0.37	0	0.415
24	1	1	0.53	49	1.09	0.96	0.38
25	1	1	0.475	50	0.35	0	0.445

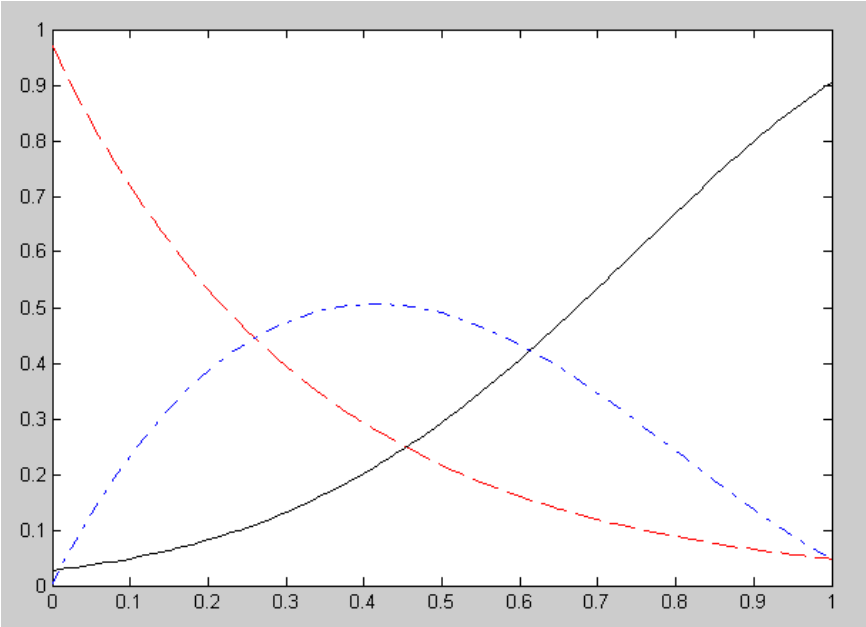


Figure 4.2: MSVM target functions for the three-class examples. $p_1(x)$ is colored in red, $p_2(x)$ is colored in blue, and $p_3(x)$ is colored in black.

tumors is used to test the performance of the newly developed method. The top 333 gene selected by F-test are used to learn the model (please refer to Chapter 3.6.2 and Chapter 4.2.3 for details). In this study, a radial basis function (RBF) kernel was used. The optimal λ is searched over a grid: $log_2(\lambda) = -10 : 1 : 10$ and the optimal λ_θ is searched over a grid: $log_2(\lambda_\theta) = -15 : 1 : 10$. The 10 fold cross-validation is used to select the optimum λ and λ_θ.

The classifier learned by the newly developed method, ANOVA-kernel L1MSVM, can achieve 100% prediction accuracy on the testing data with selected 34 genes. Table 4.5 summarizes the θ values for the selected 34 genes. It also indicates whether the selected genes are in the other gene lists generated by the other methods developed in Chapter 3. If any selected gene is in the other gene lists, it is marked as 1 and otherwise marked as 0. The adaptive-supnorm MSVM selects 28 genes into the model and shares 14 common genes with the 34 gene list select by ANOVA-kernel L1MSVM. The adaptive-L1norm MSVM selects 31 genes into the model and also shares 14 common genes with the selected 34 gene list. The supnorm MSVM selects 36 genes into the model and shares 14 common genes with the 34 gene list. The L1-norm MSVM selects 31 genes into the model and shares 12 common genes with the 34 gene list. The last column in Table 4.5 shows the rank of genes using a

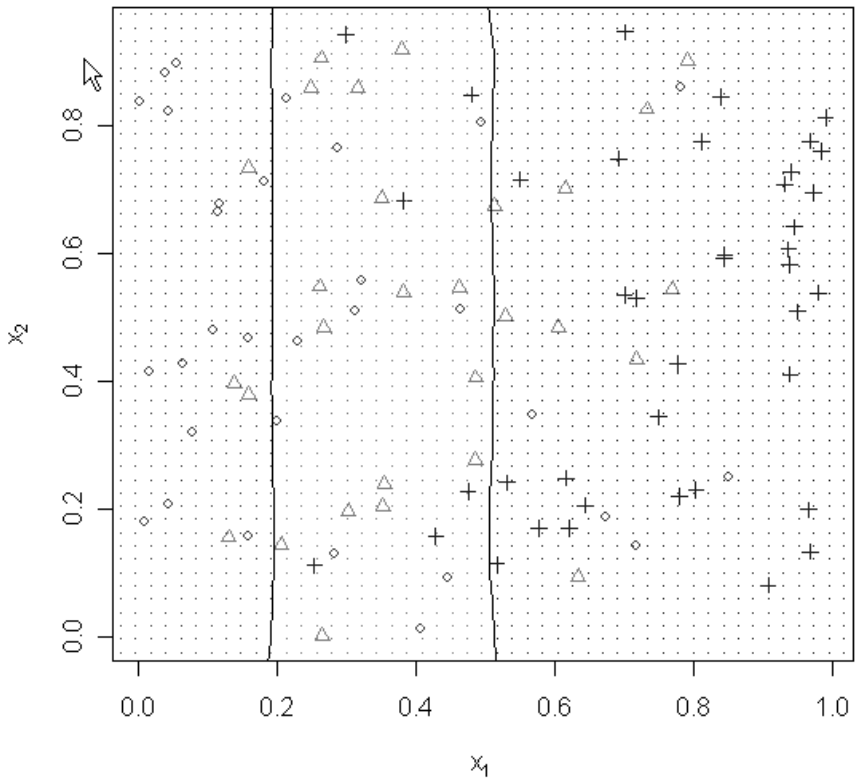


Figure 4.3: The simulated samples and learned decision boundary.

univariate measurement, the F-test statistics. Evidently, not all selected genes are top rank genes since some highly ranked genes may be highly correlated. Different from the ranking method that selects individual genes separately and ignores their correlation, the methods can select the smallest informative gene set by using multivariate selection and considering the correlation among genes automatically.

The method developed by Lee et al. (2006) was also tried on the same training and test data set with the top ranked 333 genes. The results show that 317 genes are selected by their method and the learned classifier can achieve 100% prediction accuracy. Actually, Lee et al. (2006) reported the performance of their method on this data set using the average performance in 100 bootstrap samples. Their structured MSVM selected 222 genes on average in 100 runs and achieved 2.7% test error on average. Obviously, the new method outperforms the structured MSVM by Lee et al. (2006) in terms of selecting a minimal number of gene set which can achieve an equal or higher prediction accuracy.

Table 4.5: Performance of variable selection for non-linear MSVM with kernel function in SRBCTs study

Image ID	θ-value	Adapt-SupMSVM	Adapt-L1MSVM	SupMSVM	L1MSVM	F-test Rank
784224	6.79038	1	1	1	1	1
770394	4.7939	1	1	0	1	2
365826	7.60111	1	1	1	1	3
377461	9.48367	1	1	0	0	4
814260	1.26837	1	1	1	1	6
796258	6.04749	1	1	1	1	7
1435862	5.13135	1	1	0	1	8
859359	2.01556	0	0	0	0	9
298062	7.24	0	1	1	1	12
810057	9.00213	1	1	1	1	16
43733	11.5153	1	1	1	1	18
308163	1.41253	0	0	1	0	20
841641	1.8107	1	0	1	1	21
769716	6.10996	1	1	0	0	24
563673	11.7289	1	1	1	0	25
435953	0.16268	0	0	0	0	31
812105	0.472383	0	0	0	0	33
784593	18.3761	0	0	1	0	36
364934	9.258	0	0	0	0	41
325182	3.06085	1	1	1	1	43
154472	8.59959	0	0	0	0	44
236282	10.7243	0	1	0	0	48
183337	24.5905	0	0	0	0	50
143306	4.91879	1	1	0	0	51
784257	5.52893	0	0	0	0	60
897177	6.74608	0	0	0	0	72
839552	1.95057	0	0	0	0	75
609663	2.05408	1	1	1	1	77
812965	5.66638	0	0	1	0	93
788107	3.52509	1	0	0	0	97
741831	2.4807	0	0	0	0	104
289645	3.77692	0	0	0	0	110
745019	0.490688	0	0	0	0	163
79022	1.66302	0	0	0	0	196

4.4 Discussion

SVM can select a limited number of observations (i.e. support vectors) for classifier construction. However, the original SVM formulation cannot handle the variable selection. In Chapter 3, the variable selection of linear MSVM was discussed by including the different sparse penalty functions into the MSVM formulation. However, sometimes data cannot be linear separated in the original input space. A mapping to the higher dimensional feature space is necessary in order to find an optimum hyperplane to discriminate classes. There are two mapping approaches, one is via basis function transformation and the other is using kernel function. Therefore, in this study two variable selection approaches were developed for two different non-linear MSVMs.

For non-linear MSVMs via basis function transformation, the variable selection methods developed in Chapter 3 can be directly extended to handle it. The basis function is unknown usually, thus, one can include the quadratic or interaction terms into the MSVMs to capture some non-linear effect such as interaction effect or quadratic effect. The performance of extended non-linear MSVMs using a non-linear three-class simulation data set was demonstrated. In the simulation study, it was found that when there are too many noise variables, the L2MSVM by Crammer and Singer (2001) got very poor prediction accuracy with a model including almost all noise variables, however, the four methods can achieve the prediction accuracy pretty close to the Bayes error and build a model close to the underline true model. In both simulation and real data application, the adaptive-weighted methods selected smaller gene set than the equal-weighted methods and adaptive-supnorm MSVM always outperforms the other three MSVM methods. The real data application used the same data set as the Chapter 3, i.e. the small round blue cell tumors study data set. Compared with the result in Chapter 3 without quadratic term, the extended non-linear MSVMs with adaptive-weighted penalties selected around the same number of genes while the extended non-linear MSVMs with equal-weighted penalties selected a larger gene set than the linear MSVMs. Both linear MSVMs and extended non-linear MSVMs can achieve 100% prediction accuracy.

Variable selection for non-linear MSVM via kernel function is a big challenge since kernel functions take use of values from all dimensions during calculation. Further, kernel functions are like "black-box" since the mapping function corresponding to the kernel function is unknown. In this Chapter, a further exploration of the ANOVA decomposition

of kernel function was done. Therefore, the kernel function of all dimensions can be approximated by summation of kernel functions of each dimension, kernel functions of every two dimensions, kernel functions of every three dimensions, and so on. For simplicity and interpretability, the higher order terms are ignored in this Chapter and the kernel function is decomposed as the additive effect with each kernel constructed from each dimension. A scale factor, θ, is learned for each kernel constructed from each dimension. L1-norm penalty is used to select the important scale factors out and shrink the non-important scale factors to zero. The performance of the newly developed L1MSVM with ANOVA-decomposed kernel has been tested in both simulation study and the real cancer study. The target functions for the simulation study are three non-linear functions shown in figure 4.2. The class label are defined by x_1 through three non-linear functions. The model can successfully select the correct model in 34 times out of 50 simulations and achieve the test error, 0.431, which is pretty close to the Bayes error, 0.3941. If the training sample size is increased, the model can achieve better prediction accuracy. Application of the method to the real microarray study, namely small round blue cell tumor study, was also done. By using the RBF kernel, the method can select 34 genes for classifier construction and the learned classifier can achieve 100% prediction accuracy on the independent test data set. The L1MSVM with ANOVA-decomposed kernel can achieve the competitive result as the the linear MSVM with variable selection method in terms of the number of selected genes and prediction accuracy. The gene lists selected by different methods show a big overlap, which implies the robustness and stability of the newly developed variable selection methods.

Suppose the variables that are selected are orthogonal, n samples can definitely be discriminated using n orthogonal variables. The microarray data usually has very few sample size but extremely high dimensions. Due to the very few sample size, the linear MSVM usually can find enough genes to discriminate all samples. A lot of noise variables are introduced if the kernel function is used to map data to higher dimensional feature space in the situation where linear MSVM can well separate all classes in lower dimensional input space. Thus, the performance will decrease for non-linear MSVMs since large number of noise variables are introduced in. Therefore, in the microarray study, it is not surprising that the variable selection in linear SVM generally outperform than the one in non-linear kernel SVM. In the SRBCTs study, it was also found that the L1MSVM with ANOVA-decomposed kernel can achieve the competitive result as the linear MSVM, but does not outperform the linear MSVM. However, in the situation of large sample size with fewer

dimensions, it is expected that the L1MSVM with ANOVA-decomposed kernel will perform better than the linear MSVM since linear combinations of all dimensions are not enough to discriminate all samples and the non-linear transformation to higher dimensional feature space is a big help for discriminating all samples.

In summary, the original SVM formulation does not accommodate the variable selection and the kernel trick brings more challenges for the variable selection in SVM. In this Chapter, the linear MSVMs developed in Chpater 3 was extended to handle the non-linear classification problem and further constructed a variable selection method for MSVM with kernel trick by developing an ANOVA decomposition of the kernel function. The L1MSVM with ANOVA-decomposed kernel is able to perform variable selection for MSVMs with different kernels, which is more general and flexible than extending the linear MSVMs for solving non-linear classification problems. The L1MSVM with ANOVA-decomposed kernel can select a relatively small number of genes and achieve a high prediction accuracy at the same time.

Chapter 5

Conclusions and Discussion

In this dissertation, a discussion about the variable selection issue – one of the most challenging problems for microarray and other high-dimensional "-omic" data sets was presented. First a multi-class problem related to liver toxicity severity prediction using the Random Forest and GEMS-SVM was addressed. Although SVM has demonstrated superior performance in classifying high-dimensional and low sample size data, most SVM methods, including GEMS-SVM, decompose a single multi-class problem into multiple independent binary problems using one-versus-one or one-versus-rest approaches, and do the classifier learning after the genes selection step using the univariate measurement. There are two drawbacks. First, the decomposition approaches for multi-class classification problem do not consider the correlation between different classes. Second, the gene selection using univariate measurement before the classification does not consider the correlation and interaction among genes. Therefore, the new multi-class SVM approaches were developed, which can perform multi-class classification and variable selection simultaneously. Chapter 3 and 4 talked about the variable selection in linear MSVM and kernel MSVM respectively. The variable selection in linear MSVM is achieved by incorporating sparse penalty functions into the MSVM formulation. Four different MSVM approaches, L1-norm MSVM, supnorm MSVM, Adaptive-L1 MSVM and Adaptive-supnorm MSVM, have been developed with different sparse penalty functions. The adaptive weighted penalty functions can select smaller number of genes than the equal weighted penalty functions. In most cases, the adpative-supnorm MSVM outperforms than all other methods. The gene lists selected by different methods show big overlap in the selected genes, which implies the robustness and stability in the newly developed variable selection methods. By including the quadratic and

interaction term, the four linear MSVMs are also extended to handle non-linear classification problems. Furthermore, the variable selection for non-linear MSVM with kernel function is achieved by ANOVA decomposition of the kernel function. The newly developed L1MSVM with ANOVA-decomposed kernel shows good performance in both simulation study and real data study. After the L1MSVM method with ANOVA-decomposed kernel was applied into the SRBCTs study, the results showed that it achieved the competitive result as the linear MSVM variable selection approaches in terms of the number of genes selected and the prediction accuracy. The gene lists selected by linear MSVM approaches and non-linear MSVM approaches show a big overlap in the selected genes, which implies the stability of variable selection in the methods. In addition, a comparison of the methods with currently existing variable selection methods for MSVM show that the methods can identify the smallest number of predictor genes with which a high prediction accuracy can be achieved.

SVM algorithm is pretty sensible to the outlier or wrong class labels. The liver injury data used in Chapter 2 shows some variability in the class labels in both training and test data, therefore, the methods were not applied to this data set but to other public available data sets in Chapter 3 and 4. In the future, it is planned to extend the method so that they are able to robustly handle the outlier data and reduce the influence of outlier data on the classifier learning.

Bibliography

Adams, M., R. Pierce, M. Vail, C. White, R. Tonge, T. Kavanagh, N. Fausto, S. Nelson, and S. Bruschi (2001). Enhanced acetaminophen hepatotoxicity in transgenic mice overexpressing bcl-2. *Mol Pharmacol 60(5)*, 907–915.

Allwein, E. L., R. E. Schapire, and Y. Singer (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research 1*, 113–141.

Bhasin, M. and G. Raghava (2004). Eslpred: Svm-based method for subcellular localization of eukaryotic proteins using dipeptide composition and psi-blast. *Nucleic acids research 34*, W414–9.

Blazka, M., M. Elwell, S. Holladay, R. Wilson, and L. MI (1996). Histopathology of acetaminophen-induced liver changes: role of interleukin 1 alpha and tumor necrosis factor alpha. , :. *Toxicol Pathol 24(2)*, 181–189.

Boorman, G., J. Haseman, M. Waters, J. Hardisty, and R. Sills (2002). Quality review procedures necessary for rodent pathology databases and toxicogenomic studies: the national toxicology program experience. *Toxicol Pathol 30(1)*, 88–92.

Boser, E., M. Guyon, and V. Vapnik (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Conference on Computational Learning Theory*.

Bradley, P. and O. Mangasarian (1998). Feature seleciton via concave minimization and support vector machines. In *Proceedings of the 13th International Conference on Machine Learning*.

Breiman, L. (1996). Bagging predictors. *Machine Learning 26*, 123–140.

Breiman, L. (2001). Random forests. *Machine Learning 45*, 5–32.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Bureau, A., J. Dupuis, K. Falls, K. Lunetta, B. Hayward, T. Keith, and P. Van Eerdewegh (2005). Identifying snps predictive of phenotype using random forests. *Genetic Epidemiology 28*, 171–182.

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

Bushel, P., R. Wolfinger, and G. G (2007). Simultaneous clustering of gene expression data with clinical chemistry and pathological evaluations reveals phenotypic prototypes. *BMC System Biology 1*, 15.

Casarett, L., l. J. Doul, and C. Klaassen (Eds.) (2001). *Casarett and Doull's toxicology: the basic science of poisons, 6th edn.* New York: McGraw-Hill Medical Pub. Division.

Chapelle, O., V. Vapnik, O. Bousquet, and S. Mukherjee (2002). Choosing multiple parameters for support vector machines. *Machine Learning 46*, 131–159.

Chen, Y., W. Chiang, S. Lin, K. Wu, T. Tsai, and H. BS (2004). Dual regulation of tumor necrosis factor-alpha-induced ccl2/monocyte chemoattractant protein-1 expression in vascular smooth muscle cells by nuclear factor-kappab and activator protein-1: modulation by type iii phosphodiesterase inhibition. *J Pharmacol Exp Ther 309(3)*, 978–986.

Cheung, V. and R. Spielman (2002). The genetics of variation in gene expression. *Nat Genet 32 Suppl*, 522–525.

Copple, B., P. Ganey, and R. Roth (2003). Liver inflammation during monocrotaline hepatotoxicity. *Toxicology 190(3)*, 155–169.

Cortes, C. and V. Vapnik (1995). Support-vector networks. *Machine Learning 20*, 273–279.

Crammer, K. and Y. Singer (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research 2*, 265–292.

Cristianini, N. and J. Shawe-Taylor (2000). *An introduction to Support Vector Machines*. Cambridge University Press.

Dai, X., Y. He, H. Dai, P. Lum, C. Roberts, J. Waring, and R. Ulrich (2006). Development of an approach for ab initio estimation of compound-induced liver injury based on global gene transcriptional profiles. *Genome Inform 17(2)*, 77–88.

DeCoste, D. and B. Scholkopf (2002). Training invariant support vector machines. *Machine Learning 46*, 161–190.

Devroye, L., L. Gyrfi, and G. Lugosi (1996). *A probabilistic theory of pattern recognition.* Springer-Verlag.

Diaz-Uriarte, R. and S. Alvarez (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics 7*, 3.

Dietterich, T. and G. Bakiri (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research 2*, 263–286.

Ding, C. and I. Dubchak (2001). Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics 17*, 349–58.

Duda, R. O., P. E. Hart, and D. G. Stork (2001). *Pattern classification.* Wiley Interscience.

Dudoit, S., J. Fridlyand, and T. Speed (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association 97*, 77–87.

Dumais, S., J. Platt, D. Heckerman, and M. Sahami (1998). Inductive learning algorithms and representations for text categorization. In *In Proc. 7th International Conference on Information and Knowledge Management.*

El-Hassan, H., K. Anwar, P. Macanas-Pirard, M. Crabtree, S. Chow, V. Johnson, P. Lee, R. Hinton, S. Price, and K. GE (2003). Involvement of mitochondria in acetaminophen-induced apoptosis and hepatic injury: roles of cytochrome c, bax, bid, and caspases. *Toxicol Appl Pharmacol 191(2)*, 118–129.

Evgenios, T., M. Pontil, and T. Poggio (1999). A unified framework for regularization networks and support vector machines. Technical report, AI Memo 1654, MIT.

Fan, J. and R. Li (2001). Variable selection via penalized likelihood. *Journal of the American Statistical Association 96*, 1348–1360.

Fletcher, R. (1987). *Practical methods of optimization*. John Wiley and Sons, Inc.

Garg, A., M. Bhasin, and G. Raghava (2005). Support vector machine-based method for subcellular localization of human proteins using amino acid compositions, their order, and similarity search. *The Journal of biological chemistry 280*, 14427–32.

Golub, T., D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science 286*, 531–537.

Grandvalet, Y. and S. Canu (2003). *Advances in Neural Information Processing Systems*, Chapter Adaptive Scaling for Feature Selection in SVMs, pp. 553–560. Cambridge, MA: MIT Press.

Greer, B. and J. Khan (2004). Diagnostic classification of cancer using dna microarrays and artificial intelligence. *Annals of the New York Academy of Sciences 1020*, 49–66.

Gunther, E., D. Stone, R. Gerwien, P. Bento, and M. Heyes (2003). Prediction of clinical drug efficacy by classification of drug-induced genomic expression profiles in vitro. *Proceedings of the National Academy of Sciences of the United States of America 100*, 9608–9613.

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

Hamadeh, H. and C. Afshari (Eds.) (2004). *Toxicogenomics: principles and applications*. Hoboken, N.J.: Wiley-Liss.

Hartemink, A., D. Gifford, T. Jaakkola, and Y. RA (2002). Combining location and expression data for principled discovery of genetic regulatory network models. *Pac Symp Biocomput*, 437–449.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The elements of statistical learning: Data mining, inference and prediction*. Springer-Verlag.

Heinloth, A., G. Boorman, J. Foley, N. Flagler, and P. RS (2007). Gene expression analysis offers unique advantages to histopathology in liver biopsy evaluations. *Toxicol Pathol 35(2)*, 276–283.

Hettick, J., M. Kashon, J. Slaven, Y. Ma, J. Simpson, P. Siegel, G. Mazurek, and D. Weissman (2006). Discrimination of intact mycobacteria at the strain level: a combined maldi-tof ms and biostatistical analysis. *Proteomics 6*, 6416–6425.

Hoerl, A. and R. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*, 55–67.

Hoffmann, K., M. Firth, A. Beesley, N. de Klerk, and U. Kees (2006). Translating microarray data for diagnostic testing in childhood leukaemia. *BMC Cancer 6*, 229.

Holt, M. and C. Ju (2006). Mechanisms of drug-induced liver injury. *Aaps J 8(1)*, E48–54.

Hsu, C.-W. and C.-J. Lin (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks 13*, 415–425.

Hua, S. and Z. Sun (2001a). Support vector machine approach for protein subcellular localization prediction. *Bioinformatics 17*, 721–8.

Hua, S. and Z. Sun (2001b). A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *Journal of molecular biology 308*, 397–407.

Izmirlian, G. (2004). Application of the random forest classification algorithm to a seldi-tof proteomics study in the setting of a cancer prevention trial. *Annals of the New York Academy of Sciences 1020*, 154–174.

Jaakkola, T., M. Diekhans, and D. Haussler (2000). A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology 7*, 95–114.

Jaeschke, H., J. Gujral, and M. Bajt (2004). Apoptosis and necrosis in liver disease. *Liver Int 24(2)*, 85–89.

Kaplowitz, N. and L. DeLeve (Eds.) (2007). *Drug-induced liver disease, 2 edn.* New York: Informa Healthcare USA.

Khan, J., J. Wei, M. Ringner, L. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. Antonescu, C. Peterson, and P. Meltzer (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine 7*, 673–679.

Kohavi, R. and G. H. John (1997). Wrappers for feature subset selection. *Artificial Intelligence 97*, 273–324.

Lee, Y., Y. Kim, S. Lee, and J.-Y. Koo (2006). Structured multicategory support vector machines with analysis of variance decomposition. *Biometrika 93*, 555–571.

Lee, Y., Y. Lin, and G. Wahba (2004). Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association 99*, 67–81.

Li, L., D. Umbach, P. Terry, and J. Taylor (2004). Application of the ga/knn method to seldi proteomics data. *Bioinformatics 20*, 1638–1640.

Li, L., C. Weinberg, T. Darden, and L. Pedersen (2001). Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the ga/knn method. *Bioinformatics 17*, 1131–1142.

Lin, Y. (2002). Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery 6*, 259–275.

Linder, R., D. Dew, H. Sudhoff, D. Theegarten, K. Remberger, S. Pppl, and M. Wagner (2004). The 'subsequent artificial neural network' (sann) approach might bring more classificatory power to ann-based dna. *Bioinformatics 20*, 3544–3552.

Liu, C., C. Lin, K. Li, W. Chen, J. Chen, M. Yang, P. Yang, P. Chang, and J. Chen (2007). Genome-wide identification of specific oligonucleotides using artificial neural network and computational genomic analysis. *BMC Bioinformatics 8*, 164.

Liu, Y. and X. Shen (2006). Multicategory psi-learning. *Journal of the American Statistical Association 101*, 474–509.

Lobenhofer, E., G. Boorman, K. Phillips, A. Heinloth, D. Malarkey, P. Blackshear, C. Houle, and H. P (2006). Application of visualization tools to the analysis of histopathological data enhances biological insight and interpretation. *Toxicol Pathol 34(7)*, 921–928.

Lunetta, K., L. Hayward, J. Segal, and P. Van Eerdewegh (2004). Screening large-scale association study data: exploiting interactions using random forests. *BMC Genetics 5*, 32.

Melvin, I., E. Ie, R. Kuang, J. Weston, W. Stafford, and C. Leslie (2007). Svm-fold: a tool for discriminative multi-class protein fold and superfamily recognition. *BMC Bioinformatics 8*, Suppl 4:S2.

Merrick, B., M. Bruno, J. Madenspacher, B. Wetmore, J. Foley, R. Pieper, M. Zhao, A. Makusky, A. McGrath, and J. e. a. Zhou (2006). Alterations in the rat serum proteome during liver injury from acetaminophen exposure. *J Pharmacol Exp Ther 318(2)*, 792–802.

Messmer, U., V. Briner, and J. Pfeilschifter (1999). Tumor necrosis factor-alpha and lipopolysaccharide induce apoptotic cell death in bovine glomerular endothelial cells. *Kidney Int 55(6)*, 2322–2337.

Morley, M., C. Molony, T. Weber, J. Devlin, K. Ewens, R. Spielman, and C. VG (2004). Genetic analysis of genome-wide variation in human gene expression. *Nature 430(7001)*, 743–747.

Narayanan, A., E. Keedwell, and B. Olsson (2002). Artificial intelligence techniques for bioinformatics. *Appl Bioinformatics 1*, 191–222.

Nguyen, M. and J. Rajapakse (2005). Two-stage multi-class support vector machines to protein secondary structure prediction. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*.

O'Neill, M. and L. Song (2003). Neural network analysis of lymphoma microarray data: prognosis and diagnosis near-perfect. *BMC Bioinformatics 4*, 13.

Osuna, E., R. Freund, and F. Girosi (1997). Training support vector machines: An application to face detection. In *In Proc. IEEE Conference on Computer Vi- sion and Pattern Recognition*.

Pal, N., K. Aguan, A. Sharma, and S. Amari (2007). Discovering biomarkers from gene expression data for predicting cancer subgroups using neural networks and relational fuzzy clustering. *BMC Bioinformatics 8*, 5.

Pan, F., B. Wang, X. Hu, and P. W. (2004). Comprehensive vertical sample-based knn/lsvm classification for gene expression analysis. *Journal of Biomedical Informatics 37*, 240–248.

Pavlidis, P., J. Weston, J. Cai, and W. N. Grundy (2001). Gene functional classification from heterogeneous data. In *In Proc. Fifth Annual International Conference on Computational Biology*.

Pham, T., K. Satou, and T. Ho (2005). Support vector machines for prediction and analysis of beta and gamma-turns in proteins. *Journal of bioinformatics and computational biology 3*, 343–58.

Pierce, R., C. Franklin, J. Campbell, R. Tonge, W. Chen, N. Fausto, S. Nelson, and B. SA (2002). Cell culture model for acetaminophen-induced hepatocyte death in vivo. *Biochem Pharmacol 64(3)*, 413–424.

Qin, Y., S. Auh, L. Blokh, C. Long, I. Gagnon, and K. Hamann (2007). Tnf-alpha induces transient resistance to fas-induced apoptosis in eosinophilic acute myeloid leukemia cells. *Cell Mol Immunol 4(1)*, 43–52.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning representations by back-propagating errors. *Nature 323*, 533–536.

Saldanha, A. (2004). Java treeview–extensible visualization of microarray data. *Bioinformatics 20(17)*, 3246–3248.

Schena, M., D. Shalon, R. Davis, and P. Brown (1995). Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science 270(5235)*, 467–470.

Schwender, H., M. Zucknick, K. Ickstadt, H. Bolt, and G. network. (2004). A pilot study on the application of statistical classification procedures to molecular epidemiological data. *Toxicology letters 151*, 291–299.

Shamim, M., M. Anwaruddin, and H. Nagarajaram (2007). Support vector machine-based classification of protein folds using the structural properties of amino acid residues and amino acid residue pairs. *Bioinformatics 23*, 3320–7.

Statnikov, A., C. Aliferis, I. Tsamardinos, D. Hardin, and L. S. (2005). A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics 21*, 631–43.

Statnikov, A., I. Tsamardinos, Y. Dosbayev, and C. Aliferis (2005). Gems: a system for automated cancer diagnosis and biomarker discovery from microarray gene expression data. *Int J Med Inform 74(7-8)*, 491–503.

Strobl, C., A. Boulesteix, A. Zeileis, and T. Hothorn (2007). Bias in random forest variable importance measures: illustrations, sources and a solution. *BMC Bioinformatics 8*, 25.

Tarca, A., J. Cooke, and J. Mackay (2005). A robust neural networks approach for spatial and intensity-dependent normalization of cdna microarray data. *Bioinformatics 21*, 2674–2683.

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

Tudzarova-Trajkovska, S. and J. Wesierska-Gadek (2003). Strong induction of p73 protein in vivo coincides with the onset of apoptosis in rat liver after treatment with the hepatocarcinogen n-nitrosomorpholine (nnm). *J Cell Biochem 90(4)*, 837–855.

Vapnik, V. (1996). *Computational learning and probabilistic reasoning*, Chapter Structure of Statistical Learning Theory, pp. 33–41. John Wiley and Sons.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Network 10(5)*, 117–128.

Vert, J. (2002). Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*.

Wahba, G. (1998). *Advances in kernel methods: Support vector learning*, Chapter Support Vector Machines, Reproducing Kernel Hilbert Spaces, and randomized GACV, pp. 69–87. MIT Press.

Wahba, G., Y. Wang, C. Gu, D. Xiang, R. Klein, and B. Klein (1994). *Advances in Neural Information Processing Systems*, Chapter Structured Machine Learning for 'Soft' Classification with Smoothing Spline ANOVA and Stacked Tuning, Testing and Evaluation, pp. 415–422. San Francisco, CA: Morgan Kaufmann.

Wang, H., G. Li, and G. Jiang (2007). Robust regression shrinkage and consistent variable selection via the lad-lasso. *Journal of Business and Economics Statistics 25*, 347–355.

Wang, L. and X. Shen (2007). On l1-norm multi-class support vector machines: methodology and theory. *Journal of the American Statistical Association 102*, 583–594.

Weber, L., M. Boll, and A. Stampfl (2003). Hepatotoxicity and mechanism of action of haloalkanes: carbon tetrachloride as a toxicological model. *Crit Rev Toxicol 33(2)*, 105–136.

Weston, J., A. Elisseeff, B. Scholkopf, and M. Tipping (2003). Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research 3*, 1439–1461.

Weston, J., S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik (2001). *Advances in kernel methods: Support vector learning*, Chapter Feature selectioni for SVMs, pp. 668–674. Cambridge, MA: MIT press.

Weston, J. and C. Watkins (1999). Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*.

Wu, B., T. Abbott, D. Fishman, W. McMurray, G. Mor, K. Stone, D. Ward, K. Williams, and H. Zhao (2003). Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics 19*, 1636–1643.

Yao, Z. and W. Ruzzo (2006). A regression-based k nearest neighbor algorithm for gene function prediction from heterogeneous data. *BMC Bioinformatics 7*, S11.

Yu, J., V. Smith, P. Wang, A. Hartemink, and E. Jarvis (2002). Using bayesian network inference algorithms to recover molecular genetic regulatory networks. In *International Conference on Systems Biology*.

Zamara, E., S. Galastri, S. Aleffi, I. Petrai, M. Aragno, R. Mastrocola, E. Novo, C. Bertolani, S. Milani, and V. F. et al (2007). Prevention of severe toxic liver injury and oxidative stress in mcp-1-deficient mice. *J Hepatol 46(2)*, 230–238.

Zeeberg, B., H. Qin, S. Narasimhan, M. Sunshine, H. Cao, D. Kane, M. Reimers, R. Stephens, D. Bryant, and S. e. a. Burt (2005). High-throughput gominer, an 'industrial-strength' integrative gene ontology tool for interpretation of multiple-microarray experi-

ments, with application to studies of common variable immune deficiency (cvid). *BMC Bioinformatics 6*, 168.

Zhang, H. and W. Lu (2007). Adaptive-lasso for cox's proportional hazard model. *Biometrika 94*, 691–703.

Zhang, H. H., J. Ahn, X. Lin, and C. Park (2006). Gene selection using support vector machines with non-convex penalty. *Bioinformatics 22*, 88–95.

Zhang, H. H., Y. Liu, Y. Wu, and J. Zhu (2006). Variable selection for multicategory svm via sup-norm regularization. Technical report, Institute of Statistics Mimeo Series 2596, NCSU.

Zhu, J., T. Hastie, S. Rosset, and R. Tibshirani (2003). 1-norm support vector machines. *Neural Information Processing Systems 16*, 49–56.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association 101*, 14181429.