Multi-category classification is an important topic in statistical learning and data mining. It has many applications, such as handwritten zip code digit recognition and cancer type DNA microarray classification. Multi-classification in many practical problems is even more challenging by the presence of a large number of candidate predictors. Especially in biological or medical applications such as microarrays, the overwhelming number of variables far exceeds the size of training samples even though and the underlying model may be sparse. In these applications it is essential to identify important variables for achieving classifiers with higher prediction accuracy and better model interpretability.

However, variable selection in multi-classification is much complicated than in binary classification or regression problems. We need to estimate the complex multiple discriminant functions for different classes, and also consider which variables should be included in each function. In this dissertation, we address the multi-classification variable selection problem by introducing a new penalty, the supSCAD penalty. Especially designed for multiclass problems, this new penalty groups all the coefficients by their associated covariates and imposes a SCAD penalty only on the supnorm of each group. This is where its name, supSCAD, came from. We apply the new variable selection method to both soft and hard classification through supSCAD multinomial logistic regression and supSCAD multicategory support vector machine. We show that, with a proper choice of the tuning parameter, the supSCAD multinomial logistic regression can identify the underlying sparse model consistently and has the desired oracle properties. Using local linear and quadratic approximation to deal with the non-concave SCAD and nonlinear multinomial log-likelihood function, both procedures can be reformulated into a series of linear or quadratic programming problems. The performance of the procedures is illustrated by both simulations and real data applications.
Sparse Learning in Multiclass Problems

by
Nan Li

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

Statistics

Raleigh, North Carolina

2011

APPROVED BY:

Dr. Howard Bondell  Dr. Dennis Boos

Dr. Yichao Wu  Dr. Hao Helen Zhang
Chair of Advisory Committee
DEDICATION

To my parents Dezhi Li and Zhaohua Fu.
ACKNOWLEDGEMENTS

I would like to express sincerest gratitude to my advisor Dr. Hao Helen Zhang for her constant guidance, encouragement and support, both intellectually and financially, in my PhD study. Without her, this would never have come true. During these years, it has been such a great pleasure and honor to have the opportunity to study with her and learn from her. I will always be very thankful for her wisdom, knowledge, and patience.

I also thank Dr. Dennis Boos, Dr. Howard Bondell and Dr. Yichao Wu for being my committee members, reviewing my dissertation, providing constructive comments and offering great graduate lectures. And I greatly appreciate Dr. Jacqueline Hughes-Oliver recruiting me to the Statistics Department and offering me the invaluable opportunity to continue on my Ph.D study. I would like to thank her for her guidance, support and consideration.

Special thanks to my good friends, Dongxia, Yin zheng, Chao jingyi and Liping, for seeing me laughers and tears and sharing the ups and downs along this way.

Thank you to another Nan, my husband, for making me feel at home when I am far away from home. And my dearest parents, your unconditional love and support give me courage to pursue my dreams and make my every step meaningful.
# TABLE OF CONTENTS

List of Tables .......................................................... vi

List of Figures .......................................................... vii

Chapter 1 Review for Classification Problems ......................... 1
  1.1 The Bayes Classifier ............................................. 1
  1.2 Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) .................................................. 3
  1.3 Logistic Regression .............................................. 4
  1.4 Support Vector Machine ......................................... 5
    1.4.1 Binary SVM ............................................... 5
    1.4.2 Multicategory SVM ....................................... 14
    1.4.3 Multicategory PSVM ...................................... 16

Chapter 2 Review for Variable Selection in Classification Problems .... 18
  2.1 $L_1$ Logistic Regression ...................................... 18
  2.2 $L_1$ SVM and $L_1$ MSVM ...................................... 20
  2.3 SCAD SVM .................................................. 20
  2.4 Supnorm MSVM .............................................. 21

Chapter 3 New Variable Selection Method for Multi-class Probability Estimation ....................................................... 25
  3.1 supSCAD Logistic Regression .................................. 26
  3.2 Computation Algorithm ....................................... 29
    3.2.1 Quadratic Approximation for Log-likelihood .............. 29
    3.2.2 Optimization Schemes for supSCAD Logistic Regression .. 30
  3.3 Extensions to Nonlinear Classification ........................ 34
  3.4 Asymptotic Properties ........................................ 35
  3.5 Simulation Studies ............................................ 38
    3.5.1 Four-Class Linear Example 1 ............................ 40
    3.5.2 Four-Class Linear Example 2 ............................ 42
    3.5.3 Five-Class Linear Example ................................ 43
    3.5.4 Four-Class Linear Example 3 ............................ 44
    3.5.5 Four-Class Linear Example 4 ............................ 45
    3.5.6 Three-Class Nonlinear Example .......................... 46
  3.6 Real Examples ................................................ 48
    3.6.1 Leukemia Study .......................................... 48
    3.6.2 Small Round Blue Cell Tumors Study .................... 50
Chapter 4 New Variable Selection Method for Multi-class Support Vector Machines ......................................................... 52
  4.1 supSCAD MSVM and MPSVM .......................................................... 53
  4.2 Computational Algorithm for supSCAD MSVM and MPSVM .......... 54
  4.3 Simulation Studies ................................................................. 57
    4.3.1 Four-Class Linear Example 1 ................................................. 58
    4.3.2 Four-Class Linear Example 2 ............................................... 60
    4.3.3 Four-Class Linear Example 3 ............................................... 61
    4.3.4 Four-Class Linear Example 4 ............................................... 62
    4.3.5 Three-Class Nonlinear Example ............................................. 63
  4.4 Real Examples ................................................................. 65
    4.4.1 Leukemia Study ............................................................... 65
    4.4.2 Small Round Blue Cell Tumors Study ....................................... 66

Chapter 5 Conclusion and Discussion ............................................. 67

References ..................................................................................... 69

Appendix ......................................................................................... 72
  Appendix A Proofs ........................................................................... 73
    A.1 Regularity Conditions .............................................................. 73
    A.2 Proof of Theorem 1 ................................................................. 74
    A.3 Proof of Lemma 1 ................................................................. 75
    A.4 Proof of Theorem 2 ................................................................. 76
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table 3.1</th>
<th>Simulation results for the four-class linear example 1.</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.2</td>
<td>Simulation results for four-class linear example 2.</td>
<td>43</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Simulation results for the five-class linear example.</td>
<td>43</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>Simulation results for the four-class linear example 3.</td>
<td>44</td>
</tr>
<tr>
<td>Table 3.5</td>
<td>Simulation results for four-class linear example 4.</td>
<td>45</td>
</tr>
<tr>
<td>Table 3.6</td>
<td>Classification and variable selection results for the nonlinear example</td>
<td>46</td>
</tr>
<tr>
<td>Table 3.7</td>
<td>Class distribution of the Leukemia data</td>
<td>49</td>
</tr>
<tr>
<td>Table 3.8</td>
<td>Classification and variable selection results for the Leukemia data</td>
<td>50</td>
</tr>
<tr>
<td>Table 3.9</td>
<td>Class distribution of the SRBCT data</td>
<td>50</td>
</tr>
<tr>
<td>Table 3.10</td>
<td>Classification and variable selection results for the SRBCT data</td>
<td>51</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Simulation results for the four-class linear example 1</td>
<td>58</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Simulation results for four-class example 2</td>
<td>60</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Simulation results for the four-class example 3</td>
<td>61</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Simulation results for four-class linear example 4.</td>
<td>62</td>
</tr>
<tr>
<td>Table 4.5</td>
<td>Simulation results for nonlinear example</td>
<td>63</td>
</tr>
<tr>
<td>Table 4.6</td>
<td>Class distribution of the Leukemia data</td>
<td>65</td>
</tr>
<tr>
<td>Table 4.7</td>
<td>Classification and variable selection results for the Leukemia data</td>
<td>65</td>
</tr>
<tr>
<td>Table 4.8</td>
<td>Class distribution of the SRBCT data</td>
<td>66</td>
</tr>
<tr>
<td>Table 4.9</td>
<td>Classification and variable selection results for the SRBCT data</td>
<td>66</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

| Figure 1.1 | Optimum hyperplane for binary classification in separable case. Reprinted from Hastie et al. (2001) | 6 |
| Figure 1.2 | SVM for separate case | 7 |
| Figure 1.3 | SVM for Non-separate Case | 11 |
| Figure 2.1 | SCAD penalty with $a=3.7$ and $\lambda = 1$ | 22 |
| Figure 3.1 | The Bayes boundary for the four-class example | 41 |
| Figure 3.2 | The Bayes boundary for the nonlinear three-class example | 47 |
| Figure 4.1 | The Bayes boundary for the four-class example 1 | 59 |
| Figure 4.2 | The Bayes boundary for the nonlinear three-class example | 64 |
Chapter 1

Review for Classification Problems

1.1 The Bayes Classifier

In a typical classification problem, we are given a training data set containing \( n \) pairs of observations \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \), where \( x_i = (x_{i1}, \ldots, x_{id})^T \in \mathbb{R}^d \) is the input vector, and the output \( y_i \in \mathcal{A} = \{1, 2, \ldots, K\} \) indicates its class label. Our task is to learn a discriminating rule \( \phi : \mathbb{R}^d \rightarrow \{1, 2, \ldots, K\} \) so that when given a new input \( x \) we can accordingly assign it a class label from \( \mathcal{A} \).

Let us have a look at three classification examples in the real world. First is the email/spam data from Hastie, Tibshirani & Friedman (2001). Emails provide a fast and economical way of communication, as well as result in a dramatic increase of spam emails. In this 4601 email message dataset, we want to find an efficient spam/email filtering method which predicts whether a message is a true email or a spam. The input are the relative frequencies of 57 most commonly occurring words and punctuation marks in the email messages. The output is the email type (email/spam).

The second example is the handwritten zip code digit recognition data, which is also from Hastie et al. (2001). In this dataset, handwritten ZIP codes on envelopes consist of five digits from \( \{0, 1, \ldots, 9\} \). Each digit is scanned in image and converted into a \( 16 \times 16 \) pixel matrix with each pixel ranging in intensity from 0 to 255. The US postal service wishes to use these pixel mapping as an input, design a system to read handwritten zip codes on mail and output a digit prediction between 0 and 9. This problem has a 256 dimensional input space, \( x \in \mathbb{R}^{256} \), and the output is one of 10 possible outcomes, \( \mathcal{A} = \{0, 1, \ldots, 9\} \).
The last one is cancer classification using DNA microarray data. With the breakthrough of microarray techniques in biology, expression levels of thousands of genes can be measured simultaneously from a single cell sample. This generates a large amount of high-throughput data with “high dimensional low sample size” property. Given a set of labeled samples from different cancer types, our goal is to predict the diagnostic cancer category of a sample on the basis of its gene expression profile. Molecular diagnosis based on high-throughput genomic data present a major challenge due to the overwhelming number of variables and complex multi-class nature of cancer samples. In this so called “large p small n” case, most classical classification methodologies, like Fisher linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and logistic regression, cannot be applied since the smaller number of samples does not provide enough information to estimate the underlying high-dimensional covariance matrix. The key reason for this failure is that among p columns of microarray data many are highly correlated, thus carrying redundant information. Therefore the covariance matrix is singular and cannot be inverted. To confront this challenge, some modern solutions have been developed, including a family of regularization methods, K-nearest neighbor (K-NN), and SVM.

In general statistical framework, we assume \((x_i, y_i)\) are i.i.d. from an unknown probability distribution \(P(x, y)\). Given a classifier \(\phi\), its performance is usually measured by the misclassification error rate i.e. the probability that the classification is different from \(y\),

\[
\text{GE}(\phi) = P(y \neq \phi(x)) = E_{(x,y)}[I(y \neq \phi(x))].
\] (1.1)

The optimal classifier in terms of minimizing this misclassification error rate is known as Bayes classifier.

Let \(p_k(x) = P(y = k|x)\) be the conditional probability of class \(k\) given \(x\) for \(k = 1, \ldots, K\). The Bayes classifier which minimizes GE is then given by

\[
\phi_B(x) = \arg \min_{k=1,\ldots,K} [1 - p_k(x)] = \arg \max_{k=1,\ldots,K} p_k(x).
\] (1.2)

However, in practice such \(p_k(x)\) are rarely available to produce the Bayes classifier. So Bayes rule is just our golden standard to aim for. Usually, we estimate a decision function
vector $\hat{f} = \{\hat{f}_1, \ldots, \hat{f}_K\}$ and define the decision rule as

$$\hat{\phi}(x) = \arg \max_{1 \leq k \leq K} \hat{f}_k(x).$$  \hspace{1cm} (1.3)

Each $\hat{f}_k$ represents the strength of the evidence that an example with input $x$ belongs to the class $k$, $k = 1, 2, \ldots, K$. A classifier induced by $\hat{f}$ assigns an example with $x$ to the class with the largest $\hat{f}_k(x)$.

There are mainly two types of classification methods, soft classification and hard classification. Soft classification approximates the Bayes rule by estimating $\hat{p}_k(x)$ first and then classifying to the maximum estimated probability. Among them are logistic regression, LDA, and QDA. On the other hand, hard classification bypasses the posterior probability $\hat{p}_k(x)$ and directly estimating the argmax function in (1.3). A famous example is support vector machine (SVM) (Vapnik (1995)).

In the rest of this Chapter, we will give a brief review of these methods.

1.2 Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA)

LDA and QDA are developed under normal distribution assumptions for explanatory variables. Let $g_k(x)$ denote the density of class $k$, and $\pi_k$ denote the prior probability of class $k$, with $\sum_{k=1}^{K} \pi_k = 1$. Then the posterior distribution of class $k$ after observing $x$ is:

$$p_k(x) = \frac{g_k(x)\pi_k}{\sum_{k=1}^{K} g_k(x)\pi_k}, \hspace{1cm} k = 1, \ldots, K.$$  

Now assume the distribution for class $k$ is multivariate normal with mean $\mu_k$ and covariance matrix $\Sigma_k$,

$$g_k(x) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left[-\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)\right].$$
Then the Bayes classification is to the class with maximum

\[ D_k = \log [g_k(x) \pi_k] \]
\[ = \log g_k(x) + \log \pi_k \]
\[ = -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \log |\Sigma_k| + \log \pi_k. \]

In practise, the parameters of the multivariate normal distributions are not known. Using training data, we can estimate them by

- \( \hat{\pi}_k = n_k/n \), where \( n_k \) is the number of observations in class \( k \),
- \( \hat{\mu}_k = \sum_{y_i=k} x_i / n_k \),
- \( \hat{\Sigma}_k = \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (n_k - 1) \),
- \( \hat{\Sigma} = \sum_{k=1}^K \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (n - K) \).

Then we produce an estimate for \( D_k \) and classify to the class with highest estimated \( \hat{D}_k \). The boundary between two classes is a quadratic function of \( x \), so this method of classification is known as QDA.

Further suppose that all the classes have a common covariance matrix \( \Sigma \). Then all the classification boundaries among the \( K \) classes are linear functions of \( x \) and we can maximize

\[ D_k = x \Sigma^{-1} \mu_k^T - \mu_k \Sigma^{-1} \mu_k^T / 2 + \log \pi_k. \]

This procedure is known as LDA.

1.3 Logistic Regression

Logistic regression assumes the following forms of log-odds (McCullagh and Nelder 1989):

\[ \log \frac{P(y = k|x)}{P(y = K|x)} = \beta_{k0} + \beta_k^T x, \ k = 1, ..., K - 1. \]
Those are linear functions in the explanatory variables, so are the decision boundaries. Inverting the log-odds, we have the posterior probabilities as:

\[
P(y = k|\mathbf{x}) = \frac{\exp(\beta_{k0} + \beta_k^T \mathbf{x})}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T \mathbf{x})}, \quad k = 1, ..., K - 1, \\
P(y = K|\mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T \mathbf{x})}.
\]

They automatically sum to one and remain in (0, 1). Logistic regression models are usually fit by maximizing the conditional likelihood of \( y \) given \( \mathbf{x} \) using the training data, like binomial or multinomial log-likelihood.

In logistic regression, only the conditional distribution \( P(y = k|\mathbf{x}) \) are specified and the marginal density of \( \mathbf{x} \) are left arbitrarily as any density function \( P(\mathbf{x}) \). While in LDA and QDA, the joint distribution of \( \mathbf{x} \) and \( y \) with \( P(\mathbf{x}) \) is specified a mixture of Gaussian. Therefore logistic regression is more general and should be more robust. But if the Gaussian assumption made by LDA/QDA is appropriate, then LDA/QDA tend to estimate the parameters more efficiently by using more information from the data (Hastie et al. (2001)).

### 1.4 Support Vector Machine

#### 1.4.1 Binary SVM

The SVM (Vapnik, 1995) was intended for binary classification problems and originated from machine learning society. Recently it has also gained increasing attention from statistics community. The SVM has a nice geometrical interpretation of discriminating one class from the other by a hyperplane with maximum margin between the two classes. The margin is defined as the distance to the closest points from either class in training data (Vapnik, 1995). Thus SVM is also known as a large margin classifier. Indeed, there are infinite many hyperplanes which can separate the two classes perfectly if the training data are linearly separable in the original input space. But the optimal hyperplane is the one with the maximum margin (Figure 1.1). If the two classes overlap in the input space, then soft-margin SVM uses slack variables to control an upper bound of the misclassification error. If the data cannot be separated by a hyperplane in the original input space, then nonlinear SVM performs classification by mapping the input data
from original input space to a higher dimensional feature space and then constructing a higher dimensional hyperplane that optimally separates the data into two categories. Lin (2002) showed that under some general conditions, the SVM solution approaches Bayes rule when the sample size increases.

Figure 1.1: Optimum hyperplane for binary classification in separable case. Reprinted from Hastie et al. (2001)

We will start with the simplest case: linear machines trained on separable data. Recall that the training data consist of $n$ pairs of $\{x_i, y_i\}, i = 1, ..., n$, with $x \in \mathbb{R}^d$. In binary classification, we usually have $y_i \in \{-1, 1\}$. There exists a hyperplane completely separating the two classes. Define the hyperplane as $\{x : f(x) = x^T \beta + \beta_0 = 0\}$, where $\beta$ is the normal to this hyperplane, $|\beta_0|/||\beta||$ is the perpendicular distance from the hyperplane to the origin, and $||\beta||$ is the Euclidean norm of $\beta$. The SVM looks for the optimal hyperplane as:

$$\max_{\beta_0, \beta, ||\beta|| = 1} M$$

subject to $y_i(x_i^T \beta + \beta_0) \geq M$, for $i = 1, ..., n$. (1.5)
Rescaling $\beta$ and $\beta_0$, we can get rid of $||\beta|| = 1$ by replacing the constraint condition with

$$y_i(x_i^T \beta + \beta_0) \geq M||\beta||. \quad (1.6)$$

If arbitrarily set $||\beta||$ equal to $1/M$, then (1.6) can be written as

$$y_i(x_i^T \beta + \beta_0) \geq 1. \quad (1.7)$$

The points for which the inequalities (1.7) hold are shown in Figure 1.2.

Hyperplanes $H_1$ and $H_2$ are parallel and no training data fall between them. Points on the hyperplane $H_1$ satisfy $\{x : x^T \beta + \beta_0 = -1\}$ and points on the hyperplane $H_2$ satisfy $\{x : x^T \beta + \beta_0 = 1\}$. The perpendicular distances are $|1 - \beta_0|/||\beta||$ from origin.
to $H_1$ and $|1 - \beta_0||\beta||$ from origin to $H_2$, and the margin is $2/||\beta||$. Thus we can find the hyperplane which gives the maximum margin by minimizing $||\beta||^2$, subject to constraint (1.7). Then (1.5) is equivalent to

$$\min_{\beta_0, \beta} \frac{1}{2} ||\beta||^2$$

subject to $y_i(x_i^T \beta + \beta_0) \geq 1$, for $i = 1, \ldots, n$. 

This convex optimization problem can be solved by quadratic programming technique. The primal function is obtained by introducing Lagrange multipliers $\alpha_i \geq 0$ for $i = 1, \ldots, n$, one for each of the inequality constraints in (1.8),

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

$L_P$ has to be minimized with respect to the primal variables $(\beta_0, \beta)$ and maximized with respect to the dual variables $\alpha_i$'s. Setting the first derivatives to zero, we have:

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

$$\frac{\partial L_P}{\partial \beta} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i x_i = \beta.$$  

Substituting $\beta_0$ and $\beta$ in (1.9) with (1.10), we get the following Wolfe dual problem

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to $\alpha_i \geq 0$, for $i = 1, \ldots, n,$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$  

Minimizing the primal problem $L_P$ and maximizing the dual problem $L_D$ give the same solution of $\hat{\beta}_0$, $\hat{\beta}$ and $\hat{\alpha}$.

For a convex problem with linear constraints, such as SVM, KKT conditions are necessary and sufficient for $\beta_0$, $\beta$ and $\alpha$ to have a solution (Fletcher, 1987). Therefore, solving the SVM problem is equivalent to finding a solution to the KKT conditions. For
the primal problem \((1.9)\), the KKT conditions are:

\[
\frac{\partial L_P}{\partial \beta_0} = \sum_{i=1}^{n} \alpha_i y_i = 0, \tag{1.12}
\]

\[
\frac{\partial L_P}{\partial \beta} = \beta - \sum_{i=1}^{n} \alpha_i y_i x_i = 0, \tag{1.13}
\]

\[
\alpha_i \geq 0, \text{for } i = 1, \ldots, n, \tag{1.14}
\]

\[
\alpha_i [y_i (x_i^T \beta + \beta_0) - 1] = 0, \text{for } i = 1, \ldots, n, \tag{1.15}
\]

\[
y_i (x_i^T \beta + \beta_0) \geq 1, \text{for } i = 1, \ldots, n. \tag{1.16}
\]

According to the constraint \((1.15)\), each observation satisfies either \(y_i (x_i^T \beta + \beta_0) - 1 = 0\) or \(\alpha_i = 0\). If \(y_i (x_i^T \beta + \beta_0) - 1 \neq 0\), which means the observation is outside of the margin, then \(\alpha_i\) will be equal to 0. If \(y_i (x_i^T \beta + \beta_0) - 1 = 0\), which means the observation lies on one of the two parallel hyperplanes \((H_1 \text{ or } H_2)\), then \(\alpha_i\) may or 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 \((1.13)\), \(\beta = \sum_{i=1}^{n} \alpha_i y_i x_i = \sum_{i \in \text{SVs}} \alpha_i y_i x_i\), we see that \(\beta\) is only defined by support vectors. Thus, the support vectors play a big role in shaping the decision boundary

\[
\hat{f}(x) = \beta_0 + \sum_{i \in \text{SVs}} \hat{\alpha}_i y_i x_i^T x. \tag{1.17}
\]

If all the other training samples were removed (or moved around, but so as not to cross \(H_1 \text{ and } H_2\)), then the separating hyperplane would not change. But to identify the SV points, all the data play a role. While \(\hat{\beta}\) can be explicitly determined by the training procedure, \(\beta_0\) cannot. By using the KKT conditions \((1.15)\), \(\beta_0\) can be solved with any of the support vectors or with an average of all solutions from all support vectors.

Once we have trained a Support Vector Machine and got the solutions \(\hat{\beta}_0\) and \(\hat{\beta}\), for a given test point \(x\), we take the class of \(x\) to be \(\text{sign}(x^T \beta + \hat{\beta}_0)\).

Now suppose the two classes of training data are overlap, and we cannot find a hyperplane satisfies the constraint \(y_i (x_i^T \beta + \beta_0) \geq 1\) for every observations. To relax the constraint, we introduce positive slack variables \(\vartheta = (\vartheta_1, \ldots, \vartheta_n)\) for each observation:

\[
x_i^T \beta + \beta_0 \geq 1 - \vartheta_i \text{ for } y_i = 1,
\]

\[
x_i^T \beta + \beta_0 \leq -1 + \vartheta_i \text{ for } y_i = -1.
\]
Since $\sum_{i=1}^n \vartheta_i$ is an upper bound on the number of training errors, we would like to impose a further cost on it. The corresponding optimization problem is:

$$
\min_{\beta, \beta_0, \vartheta} \frac{1}{2} \| \beta \|^2 + \gamma \sum_{i=1}^n \vartheta_i
$$
subject to $y_i(x_i^T \beta + \beta_0) \geq 1 - \vartheta_i$, for $i = 1, \ldots, n,$

$$\vartheta_i \geq 0, \text{ for } i = 1, \ldots, n,$$

where $\gamma > 0$ is tuning parameter. A larger $\gamma$ means assigning a higher penalty to errors while a smaller $\gamma$ means putting more penalty on the margin width. Too large $\gamma$ would make the model over fit the training sample but have a poor generalization in testing data, and too small $\gamma$ would result in an underfitting model with both large training and testing error. A separate tuning data set or cross-validation can be used to identify an appropriate $\gamma$ value. Figure 1.3 is a summarization of this non-separable case.

Introducing Lagrange multipliers $\alpha_i \geq 0$ and $\nu_i \geq 0$ for each constraint, the Lagrange formulation for (1.18) is:

$$L_P = \frac{1}{2} \| \beta \|^2 + \gamma \sum_{i=1}^n \vartheta_i - \sum_{i=1}^n \nu_i \vartheta_i - \sum_{i=1}^n \alpha_i [y_i(x_i^T \beta + \beta_0) - 1 + \vartheta_i].$$

The optimization problem is to minimize $L_p$ with respect to primal variables $\beta_0$, $\beta$, $\vartheta$ or to maximize $L_p$ with respect to dual variables $\alpha$, $\nu$. The KKT conditions for the primal problem (1.19) are therefore

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

$$\frac{\partial L_P}{\partial \beta} = \beta - \sum_{i=1}^n \alpha_i y_i x_i = 0,$$

$$\frac{\partial L_P}{\partial \vartheta_i} = \gamma - \nu_i - \alpha_i = 0,$$

$$\alpha_i [y_i(x_i^T \beta + \beta_0) - 1 + \vartheta_i] = 0, \text{ for } i = 1, \ldots, n,$$

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

$$\alpha_i \geq 0,$$
Figure 1.3: SVM for Non-separate Case
\[ \vartheta_i \geq 0, \quad \nu_i \geq 0, \quad \nu_i \vartheta_i = 0. \tag{1.26} \]

Substituting (1.21), (1.22), and (1.23) into \( L_P \), we get the Wolfe dual problem:

\[
\max_{\alpha} L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{subject to } 0 \geq \alpha_i \geq \gamma, \text{ for } i = 1, \ldots, n, \\
\sum_{i=1}^{n} \alpha_i y_i = 0. \tag{1.29} \]

The solution is again given by \( \hat{\beta} = \sum_{i \in SVs} \alpha_i y_i x_i \).

There are four possible situations in this non-separable case. In the first situation, a point \( x_i \) is correctly classified and lies outside the hyperplanes of \( H_1 \) and \( H_2 \). It satisfies \( y_i (x_i^T \beta + \beta_0) > 1 \), and \( \vartheta_i = 0, \alpha_i = 0 \) according to constraint (1.23). In the second situation, a point \( x_i \) is correctly classified and just lies on \( H_1 \) or \( H_2 \). It satisfies \( y_i (x_i^T \beta + \beta_0) = 1 \), and \( \vartheta_i = 0, \) but \( \alpha_i \) may or may not be zero. If \( \alpha_i \) is not zero, then the point \( x_i \) is a support vector and \( 0 < \alpha_i \leq \gamma \). The third situation is \( 0 < y_i (x_i^T \beta + \beta_0) < 1 \), where a point \( x_i \) is correctly classified but lies between \( H_1 \) and \( H_2 \). In this situation, we have \( 0 < \vartheta_i < 1 \) and \( \alpha_i = \gamma \). The fourth situation is \( y_i (x_i^T \beta + \beta_0) \leq 0 \), where a point \( x_i \) is misclassified and lies on the separate hyperplane or the other side of separate hyperplane. In this situation, \( \vartheta_i \geq 1 \) 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 hyperplanes \( (\alpha_i = \gamma) \) and all points at the other side of the hyperplane (i.e. misclassified points, \( \alpha_i = \gamma \)).

Sometimes, it is impossible to linearly separate the training data in the original input space. We then map the data from the original input space to a higher dimensional feature space and find an optimal hyperplane to discriminate the data in the enlarged space. Suppose the dictionary of basis functions of the enlarged feature space is \( \mathcal{D} = \{ h_1(x), h_2(x), \ldots, h_q(x) \} \), where \( q \) is the dimension of the enlarged feature space. The classification boundary in the enlarged feature space is given by \( \{ x : f(x) = \beta_0 + h(x)^T \beta \} \). Note that in the linear SVM, the solution (1.17) depends on input variables only through...
their inner products. Hence, when using the enlarged basis functions the new solution
should has the form of

\[ f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i y_i < h(x_i), h(x) >. \quad (1.30) \]

This implies that we need not specify the transformation \( h(x) \) at all. All we need to
know is the kernel function

\[ K(x, x') = < h(x), h(x') >. \quad (1.31) \]

This kernel \( K \) should be a symmetric positive (semi-) definite function. And the kernel
trick allows the enlarged feature space to be even infinite dimensional, i.e. \( q = \infty \),
without increasing the computational cost. Two popular kernels in the SVM literature are:

**dth - Degree polynomial** : \( K(x, x') = (1 + < x, x' >)^d \),

**Radial basis** : \( K(x, x') = \exp(-\gamma ||x - x'||^2) \).

Wahba (1998) and Evgenios et al. (2000) showed that SVM paradigm could comfort-
ably fit into the regularization framework, where one component loss is for data fitting
and the other component penalty is for controlling the model complexity. Then SVM
can be equivalently expressed as:

\[
\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i f(x_i)]_+ + \lambda \| \beta \|^2, \quad (1.32)
\]

with \( f(x) = \beta_0 + x^T \beta \) for linear SVM and \( f(x) = \beta_0 + h(x)^T \beta \) for nonlinear SVM. And the
classification rule is \( \text{sign}[f(x)] \). The loss function in (1.32), \( \sum_{i=1}^{n} [1 - y_i f(x_i)]_+ \), is refereed
as the hinge loss, where the subscript “+” indicates a positive part. The parameter \( \lambda \)
controls the trade-off between training error and model complexity. It minimizes an
average loss as well as avoids overfitting by penalizing model complexity. Further, if
\( h(x) \) are appropriate basis functions of a reproducing kernel Hilbert space (RKHS), then
(1.32) can be written as:

\[
\min_{f \in \mathcal{H}_K} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i f(x_i)]_+ + \lambda \|h\|_{\mathcal{H}_K}^2,
\]

(1.33)

where \( \mathcal{H}_K \) is the RKHS.

Some loss function examples used in margin-based classifiers are:

- Cross entropy for logistic regression: \( L(yf) = \log(1 + \exp(-yf(x))) \),
- Square error loss: \( L(yf) = (1 - yf(x))^2 \),
- Hinge loss: \( L(yf) = [1 - yf(x)]^+ \).

The above loss functions all have the function margin term \( yf \). They encourage large positive values of \( yf \) by penalizing negative margins more heavily than positive ones.

Lin (2002) showed the minimizer of SVM has asymptotic target function as:

\[
\arg \min_f E[(1 - yf(x))_+] = \text{sign}(P(y = 1|x) - \frac{1}{2}),
\]

(1.34)

which is the same as the Bayes rule \( \phi_B(x) \) for binary classification. Thus SVM asymptotically estimate the Bayes rule in a more efficient and direct way without estimating the actual conditional probability \( P(y = 1|x) \).

1.4.2 Multicategory SVM

To extend SVM from binary classification to multi-class classification, many researchers have proposed various procedures. There are mainly two approaches. One is decomposing a multi-class problem into multiple binary problems. The other is considering all classes at the same time and directly constructing a multi-classification decision function.

Allwein et al. (2000) presented a unified approach to study the solution of multi-class classification problems by decomposing a multi-class problem into multiple binary problems and then solving the multiple binary problems using a margin-based binary learning algorithm. The two most popular decomposition strategies are “one-vs-one” and “one-vs-rest”. In the “one-vs-one” strategy, we compute all \( \binom{K}{2} \) pairwise classifiers. For each test point, the predicted class is the one that wins the most pairwise contests. In the “one-vs-rest” strategy, a \( K \)-class problem is divided into \( K \) “one-vs-rest” problems,
and each “one-vs-rest” problem is addressed by a different class-specific binary classifier (e.g., “class 1” vs ”not class 1”). Then a new sample takes the class of classifier with the largest real valued output \( y_k = \arg \max_{k=1,\ldots,K} f_k \) where \( f_k \) is the real valued output of the \( k \)th classifier.

These two approaches indirectly solve a multi-class problem using binary classifiers, like SVMs, and achieve a reasonably good prediction accuracy. However, there are a lot of disadvantages for such decomposition strategies as showed in Lee et al. (2004). First, the “one-vs-rest” approach may fail when there is no class dominates the union of the others for some \( x \). The Bayes rule for a “one-vs-rest” binary problem is \( \phi_B(x) = \text{sign}[p_k(x) > 0.5] \). If a class \( k \) satisfies \( p_k(x) > 0.5 \) for a given \( x \), then the algorithm can determine the correct class label. However, if no class has conditional probability greater than 0.5, then it is hard for the algorithm to determine the right class label. Second, the “one-vs-rest” approach has a very unbalanced sample size in their binary problems if the sample size for one class is much smaller than the union of all the other classes. Third, the “one-vs-one” approach has a much smaller samples size, which potentially increases the variance of the learned classifier. Fourth, these two decomposing strategies cannot capture the correlation between different classes, since they break a multicategory problem into multiple independent binary problems (Crammer and Singer, 2001). Therefore, a better way to inherit and extend the optimal property of binary SVM to the multicategory case is highly desired, which can classify multiple classes simultaneously.

The multicategory SVM (MSVM) (Lee et al., 2004) is proposed to treat multiple classes simultaneously. In MSVM, multiple discriminant functions, \( f = (f_1, f_2, \ldots, f_K) \), are estimated instead of only one decision function in a binary SVM. Each \( f_k(x) \) is associated with a class \( k \) and represents the strength of a sample \((x, y)\) belonging to the class \( k \). That is, \( f_k(x) \propto P(y = k|x) \). Therefore, the classification rule in multiclass SVM is \( \phi(x) = \arg \max_{k=1,\ldots,K} f_k(x) \) while the classification rule in binary SVM is \( \phi(x) = \text{sign}(f(x)) \). And the decision boundary between class \( k \) and class \( l \) is \( \{x : f_k(x) = f_l(x)\} \). A reasonable decision vector \( f \) should encourage a large value for \( f_y(x) \) and generate small values for \( f_l(x), l \neq y \). Lee et al. (2004) showed that their loss function has good theoretical properties and ensures the solution directly target the Bayes rule in the same fashion as in the binary case.

In Lee et al. (2004), the class label for a multi-class problem is specially designed. They defined the class output \( y \) as a K-dimensional vector. When indicating class \( k \), it
has 1 in the $k$th coordinate and $\frac{-1}{K-1}$ elsewhere,

$$y_i = \begin{cases} 
(1, \frac{-1}{K-1}, \ldots, \frac{-1}{K-1}) & \text{if sample } i \text{ belongs to class 1,} \\
(\frac{-1}{K-1}, 1, \ldots, \frac{-1}{K-1}) & \text{if sample } i \text{ belongs to class 2,} \\
\ldots \\
(\frac{-1}{K-1}, \frac{-1}{K-1}, \ldots, 1) & \text{if sample } i \text{ belongs to class } K.
\end{cases}$$

Let a $K \times K$ cost matrix having 0 on the diagonal and 1 elsewhere. They define $R(\cdot)$ as a function which maps a class label $y_i$ to the $k$th row of the cost matrix if $y_i$ indicates class $k$. So, if $y_i$ represents class $k$, then $R(y_i)$ is a $k$ dimensional vector with 0 in the $k$th coordinate, and 1 elsewhere. Then they proposed the loss function as

$$\frac{1}{n} \sum_{i=1}^{n} R(y_i) \cdot (f(x_i) - y_i) + \lambda \sum_{k=1}^{K} \sum_{j=1}^{d} \beta_{kj}^2$$

subject to

$$\sum_{k=1}^{K} \beta_{k0} = 0,$$

$$\sum_{k=1}^{K} \beta_{kj} = 0, j = 1, \ldots, d.$$ (1.35)

The sum-to-zero constraints, $\sum_{k=1}^{K} \beta_{k0} = 0$ and $\sum_{k=0}^{K} \beta_{kj} = 0, j = 1, \ldots, d$, are enforced to eliminate redundancy and to assure identifiability of the solution.

### 1.4.3 Multicategory PSVM

Beside hinge loss, squared error loss can also be incorporated with SVM. Least square support vector machine (LSSVM) and proximal support vector machine (PSVM) were proposed by Suykens and Vandewalle (1999) and Fung and Mangasarian (2001). In theory, LSSVM, PSVM and SVM all target the optimal Bayes rule asymptotically. While in computation, LSSVM and PSVM are much efficient as they only need to solve a system of linear equations. Tang and Zhang (2006) extended PSVM to multiclass PSVM (MPSVM) which directly targets the boundaries among $K$ classes simultaneously by estimating some
functions of the conditional probabilities. They used the same class output code as Lee et al. (2004) and proposed the loss function as

\[
\frac{1}{n} \sum_{i=1}^{n} [y_i - f((x)_i)]^T Z(y_i)[y_i - f(x_i)],
\]

where \(Z(y_i) = \text{diag}(R(y_i))\) is a diagonal matrix.

If simply using the natural class label as output \(y_i \in \{1, \ldots, K\}\), then the linear MPSVM solves

\[
\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k) [\beta_{k0} + \beta_k^T x_i + 1]^2 + \lambda \sum_{k=1}^{K} \left( \sum_{j=0}^{d} \beta_{kj}^2 \right)
\]

subject to \(\sum_{k=1}^{K} \beta_{k0} = 0\), \(\sum_{k=1}^{K} \beta_{kj} = 0, j = 1, \ldots, d\). \hspace{1cm} (1.36)

Furthermore, they showed that the conditional class probabilities can be estimated as:

\[
\hat{p}_k(x) = 1 - (K - 1) \frac{1/[1 - \hat{f}_k(x)]}{\sum_{l=1}^{K} 1/[1 - \hat{f}_l(x)]}, \text{ for } k = 1, \ldots, K,
\]

where \(\hat{f}_k = \hat{\beta}_{k0} + \hat{\beta}_k^T x\).
Chapter 2

Review for Variable Selection in Classification Problems

Nowadays, many practical multi-classification problems are even more challenging by the presence of a large number of, but not necessarily all relevant, candidate predictors. Especially in biological or medical applications such as microarrays, the overwhelming number of variables far exceeds the size of training samples, and the underlying model is naturally sparse. Including all the candidate predictors at the initial stage of modeling could attenuate potential modeling biases (Fan and Li, 2001). However, if keeping all the variables without any discrimination, we would end up with a classifier having poor generalization performance and hardly providing any useful information on the underlying model structure. Therefore, in these applications it is essential to identify important variables for achieving classifiers with higher prediction accuracy and better model interpretability.

General methods for variable selection in regression settings are studied intensively. In this chapter, we will review some variable selection methods in multi-category classification.

2.1 $L_1$ Logistic Regression

LASSO is the short term for “Least Absolute Shrinkage and Selection Operator”, which was proposed by Tibshirani (1996) for variable selection in linear regression. It is defined similar to ridge regression, but replacing the $L_2$ penalty $\sum_{j=1}^{d} \beta_j^2$ by the $L_1$ penalty
\[ \sum_{j=1}^{d} |\beta_j|. \] Unlike ridge regression, LASSO shrinks some variables to exactly zero and produces a more sparse and interpretable model. It also inherits the characteristic of continuity and stability from ridge regression. In summary, LASSO combines the pros of model interpretability from subset selection and solution continuity from ridge regression.

Using lasso penalty to regularize binary logistic regression, Hastie et al. (2001) gave the form of \( L_1 \) penalized logistic regression as:

\[
\min_{\beta_0, \beta} -\frac{1}{n} \sum_{i=1}^{n} [I(y_i = 1)(\beta_0 + \beta^T x_i) - \log(1 + \exp(\beta_0 + \beta^T x_i))] + \lambda \sum_{j=1}^{d} |\beta_j|. \tag{2.1}
\]

When using multinomial logit model, it can be extended to the multiclass case naturally. Instead of using baseline multinomial logit model

\[
\log \frac{P(y = k|x)}{P(y = K|x)} = \beta_{k0} + \beta_k^T x, \quad k = 1, \ldots, K - 1,
\]

Hatie et al. (2001) proposed multinomial \( L_1 \) logistic regression with a symmetric posterior probability formation:

\[
P(y = 1|X = x) = \frac{e^{f_1(x)}}{\sum_{k=1}^{K} e^{f_k(x)}},
\]

\[
P(y = 2|X = x) = \frac{e^{f_2(x)}}{\sum_{k=1}^{K} e^{f_k(x)}},
\]

\[
\vdots
\]

\[
P(y = K|X = x) = \frac{e^{f_K(x)}}{\sum_{k=1}^{K} e^{f_k(x)}},
\]

where

\[
f_k(x) = \beta_{k0} + \beta_k x, \quad k = 1, \ldots, K.
\]

Then multinomial \( L_1 \) logistic regression has the form as:

\[
\min_{\beta_0, \beta} -\frac{1}{n} \sum_{i=1}^{n} \sum_{l=1}^{K} y_{il}(\beta_{l0} + \beta_l^T x_i) - \log(\sum_{l=1}^{K} \exp(\beta_{l0} + \beta_l^T x_i))) + \lambda \sum_{k=1}^{K} \sum_{j=1}^{d} |\beta_{kj}|, \tag{2.3}
\]

where \( y_{i} \) is a binary K-vector with values all zero except a 1 in position \( k \) if the class is \( k \), and \( f(x_i) = (f_1(x_i), \ldots, f_K(x_i)) \). Friedman et al. (2010) provided very fast algorithms
for fitting generalized linear models with convex penalties, including the $L_1$-penalized binary and multinomial logistic regression models.

### 2.2 $L_1$ SVM and $L_1$ MSVM

Replacing the $L_2$ norm with $L_1$ norm penalty in standard SVM, Bradley and Mangasarian (1998) introduced $L_1$ SVM. Different from the standard SVM which requires quadratic programming techniques, $L_1$ SVM is solved by linear programming. Zhu et al. (2004) studied the solution property of the $L_1$ SVM and showed that there is an efficient algorithm to compute the whole solution path since it is a piecewise linear function in the tuning parameter.

In multi-category SVM, to select important variables from $K$ decision functions is not trivial and depends on which loss function to use. Wang and Shen (2007) extended $L_1$ SVM to $L_1$ MSVM based on the loss function proposed by Lee et al. (2004), which solves:

$$
\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k) \left[ \beta_{k0} + \beta_k^T x_i + 1 \right]_+ + \lambda \sum_{k=1}^{K} \sum_{j=1}^{d} |\beta_{kj}| \\
\text{subject to } \sum_{k=1}^{K} \beta_{kj} = 0, j = 0, 1, \ldots, d,
$$

(2.4)

for linear classification rules. They developed a statistical learning theory to quantify the generalization error of $L_1$ MSVM and derived the entire solution path for linear $L_1$ MSVM.

### 2.3 SCAD SVM

Fan and Li (2001) proposed a new form of penalty, called smoothly clipped absolute deviation (SCAD) penalty, for regression problems. Different from all the previous popular penalties, it is a non-concave function. This unique property makes it provide nearly unbiased estimates for large coefficients and shrink small estimates to exactly zero, resulting in a sparse model continuous in data. SCAD penalty combines the pros from both $L_1$ norm penalty and subset selection. Incorporating the SCAD regularization in
standard binary SVM, Zhang et al. (2006) proposed SCAD SVM and applied it to high-dimensional low sample size biological data.

One instance of SCAD penalty function is plotted in Figure 2.1. Similarly to $L_1$ penalty, SCAD penalty is symmetric, nonconvex and singular at the origin, which leads to its sparse property. Different from $L_1$ penalty, SCAD applies a constant penalty for large coefficients while $L_1$ penalty increases linearly as the coefficient increases. This distinctive feature ensures SCAD penalty produce unbiased estimates for large coefficients. Mathematically, SCAD penalty has the expression of

$$J_\lambda(|\theta|) = \begin{cases} 
\lambda|\theta| & \text{if } |\theta| \leq \lambda, \\
-(|\theta|^2 - 2a\lambda|\theta| + \lambda^2) & \text{if } \lambda < |\theta| \leq a\lambda, \\
\frac{(a+1)\lambda^2}{2(a-1)} & \text{if } |\theta| > a\lambda,
\end{cases}$$

where $a > 2$ and $\lambda > 0$ are tuning parameters. Except being singular at the origin, the function $J_\lambda(|\theta|)$ has a continuous first-order derivative.

$$J_\lambda'(\theta) = \lambda I(\theta \leq \lambda) + \frac{(a\lambda - \theta)_+}{(a - 1)\lambda}I(\theta > \lambda). \quad (2.5)$$

Replacing the $L_2$ penalty in standard SVM, the binary linear SCAD SVM is

$$\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(\beta_0 + \beta x_i)]_+ + \sum_{j=1}^{d} J_\lambda(|\beta_j|).$$

### 2.4 Supnorm MSVM

The $L_1$ penalty treats all coefficients equally without any distinction. While in reality, some of the coefficients are associated with the same covariant and can be treated as a group. Taking this into account, Zhang et al. (2008) introduced a new group penalty, called supnorm penalty. It imposes a $L_1$ penalty to the maximum absolute value of the $K$ coefficients associated with the same covariant. Thus, if a variable is not important, then all the $K$ coefficients associated with it should be set to zero. Applied it to MSVM, the supnorm MSVM leads to a more sparse model than $L_1$ MSVM and is especially suitable for multi-classification. Furthermore, they proposed two forms of adaptive supnorm penalties treating variables differently according to their relative importance. Adaptive
Figure 2.1: SCAD penalty with $a=3.7$ and $\lambda = 1$
supnorm MSVMs make sure to retain important variables and drop unimportant variables more quickly. This lead to a more sparse and accurate model and hence better prediction capability.

In the $L_1$ MSVM formulation (2.4), all $\beta_{kj}$s are treated equally without any distinction. In contrast to this, the supnorm penalty groups coefficients by their associated variables. In Zhang et al. (2008), they defined $\beta$ as a weight matrix of size $K \times d$ such that its $(k, j)$ entry is $\beta_{kj}$. The $k$th row and $j$th column vector of $\beta$ are represented by $\beta_k = (\beta_{k1}, \ldots, \beta_{kd})^T$, and $\beta_{(j)} = (\beta_{1j}, \ldots, \beta_{Kj})^T$ respectively.

According to Crammer and Singer (2001), the value $\beta_{k0} + \beta_k^T x$ defines the similarity score of the class $k$. So the predicted label is the index of the row attaining the highest similarity score with $x$. The supnorm for the coefficient vector $\beta_{(j)}$ is defined as

$$\| \beta_{(j)} \|_\infty = \max_{k=1, \ldots, K} |\beta_{kj}|.$$ 

Applying the supnorm regulation to linear MSVM, we have linear supnorm MSVM as

$$\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k)[\beta_{k0} + \beta_k^T x_i + 1]_+ + \lambda \sum_{j=1}^{d} \| \beta_{(j)} \|_\infty,$$

subject to $\sum_{k=1}^{K} \beta_{kj} = 0, j = 0, 1, \ldots, d.$

The $L_1$ penalty applies equal penalty to all coefficients. It selects variables in a continuous fashion and provides a stable solution. However it may also be too restrictive, since a smaller penalty should be more desired for those variables which are so important that we want to retain them in the model. This led to the emergency of adaptive penalties. Adaptive $L_1$ penalty is also called Adaptive-LASSO, which was proposed by Zou (2006) in linear regression, Wang et al. (2007) in quantile regression, and Zhang and Lu (2007)
in Cox’s proportional hazards model. Adaptive $L_1$ penalty gives small penalty to large coefficients and large penalty to small coefficients. In this way, important variables with large coefficients are protectively retained in the model selection procedure whereas unimportant variables with small coefficients are dropped out more quickly. Motivated by this, Zhang et al. (2008) considered the following adaptive $L_1$ MSVM:

$$\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k)[\beta_{k0} + \beta_k^T x_i + \chi_k]_+ + \lambda \sum_{k=1}^{K} \sum_{j=1}^{d} \tau_{kj} |\beta_{kj}|,$$

subject to $\sum_{k=1}^{K} \beta_{kj} = 0$, $j = 0, 1, \ldots, d$.

where $\tau_{kj} > 0$ represents the weight for coefficient $\beta_{kj}$.

Adaptive supnorm penalty shares the same motivation with adaptive $L_1$ penalty. Zhang et al. (2008) gave two ways to employ the adaptive supnorm penalty.

Adaptive supnorm [I]:

$$\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k)[\beta_{k0} + \beta_k^T x_i + \chi_k]_+ + \lambda \sum_{j=1}^{d} \|\tau \beta_{(j)}\|_{\infty},$$

subject to $\sum_{k=1}^{K} \beta_{kj} = 0$, $j = 0, 1, \ldots, d$.

Adaptive supnorm [II]:

$$\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k)[\beta_{k0} + \beta_k^T x_i + \chi_k]_+ + \lambda \sum_{j=1}^{d} \|\tau \beta_{(j)}\|_{\infty},$$

subject to $\sum_{k=1}^{K} \beta_{kj} = 0$, $j = 0, 1, \ldots, d$.

The weights are empirically chosen as $\tau_{kj} = \frac{1}{|\hat{\beta}_{kj}|}$ and $\tau_j = \frac{1}{\|\hat{\beta}_{(j)}\|_{\infty}}$, where $\hat{\beta}$ is the solution of standard MSVM.
Chapter 3

New Variable Selection Method for Multi-class Probability Estimation

As discussed in Chapter 2, there is not much work on variable selection in multi-class problems. While a number of thresholding functions have been implemented with binary SVM or binomial logistic regression, only two of them, $L_1$ and supnorm, have been studied in multiclass cases. $L_1$ penalty has been implemented in multinomial logistic regression and Multicategory SVM based on the loss function proposed by Lee et al. (2004). Supnorm penalty is only implemented in the MSVM based on the same loss function.

These two penalties, $L_1$ and supnorm, achieve great success, but still have some drawbacks. Using $L_1$, variable selection is stable but does not force the related coefficients to be zero altogether if their associated variable is unimportant. Adaptive supnorm performs excellent, but it requests good estimates to construct the weights. The proper choice of weights are usually estimated from ordinary least square (Zou, 2006) or log partial likelihood (Zhang and Lu, 2007). However, in a high dimensional but small data set, such as gene expression data, these weights may not be estimated well. We will introduce a new form of penalty called supSCAD, which does not require any initial weight estimate and enforce high sparsity for multi-class problems.

In multi-category classification, multinomial logit model is probably the most widely used model. It is also a famous member of the soft classification family. In addition to determining class boundaries, predicting class labels, it also produces conditional probability estimators for future samples collected from the same population.
In this chapter, a new variable selection method in multi-class probability estimation will be explored via penalized multinomial logistic regression. In Chapter 4 we will study it with multiclass support vector machines.

3.1 supSCAD Logistic Regression

In the regression context, SCAD penalty was studied in Fan and Li (2001) and shown to have better theoretical properties than the $L_1$ penalty. Zhang et al. (2006) introduced SCAD to binary classification. They proposed SCAD SVM to give a compact classifier with high accuracy and interpretability. And their results in high-dimensional low sample size biological data show great promise to study SCAD in the multi-classification context.

However, in multiclass problems, variable selection is complicated by the increasing number of decision functions and the corresponding coefficients characterizing them. Although $L_1$ penalty can shrink some small coefficients to exact zero, it does not distinguish the sources among different coefficients. Often times, coefficients associated with the same covariate are more likely to be related with each other than those associated with different covariates. Therefore, a more efficient shrinkage method could be achieved by treating coefficients differently according to their corresponding covariates.

Consider a $K$-class problem with input variable $x$. Assume $x \in \mathbb{R}^{d+1}$, with the first column being 1’s to include the intercept term. There are $(d+1)$ different coefficients in each of the $K$ decision functions. Thus, we define a $K \times (d+1)$ decision function matrix $\beta$ as follows:

<table>
<thead>
<tr>
<th>Class 1</th>
<th>$\beta_{10}$</th>
<th>$\beta_{11}$</th>
<th>...</th>
<th>$\beta_{1j}$</th>
<th>...</th>
<th>$\beta_{1d}$</th>
<th>$\beta_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Class $k$</td>
<td>$\beta_{k0}$</td>
<td>$\beta_{k1}$</td>
<td>...</td>
<td>$\beta_{kj}$</td>
<td>...</td>
<td>$\beta_{kd}$</td>
<td>$\beta_k$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Class $K$</td>
<td>$\beta_{K0}$</td>
<td>$\beta_{K1}$</td>
<td>...</td>
<td>$\beta_{Kj}$</td>
<td>...</td>
<td>$\beta_{Kd}$</td>
<td>$\beta_K$</td>
</tr>
</tbody>
</table>

Each column of $\beta_{(j)} = (\beta_{1j}, \ldots, \beta_{Kj})^T$ are the $K$ coefficients associated with the same covariate $x_j$, $j = 0, 1, \ldots, d$. We use $x_0$ to denote the first element of $x$ corresponding to the intercept term. And each row of $\beta_{k} = (\beta_{10}, \beta_{11}, \ldots, \beta_{1d})^T$ are the $(d+1)$ coefficients characterizing the decision function $f_k$, $k = 1, \ldots, K$. A natural idea is grouping coefficients by their associated covariates, which makes the $K \times (d+1)$ coefficients into
(d + 1) groups of $\beta_{(0)}, \beta_{(1)}, \ldots, \beta_{(d)}$. For each group of $\beta_{(j)}$, its supnorm is defined as:

$$\|\beta_{(j)}\|_\infty = \max_{k=1,\ldots,K} |\beta_{kj}|. \tag{3.1}$$

Usually the intercept term $\beta_{(0)}$ is not penalized. If imposing a penalty of (3.1) on $\beta_{(j)}, j = 1, \ldots, d$, then the importance of each covariate $x_j$ is directly controlled by its largest absolute coefficient. Specifically, if $x_j$ is not important, then all the $K$ coefficients associated with it should be set to zero. On the other hand, if $x_j$ is important with a positive sup-norm, then all the associated $K$ coefficients will remain in the model as no further penalty is put on the remaining coefficients in terms of variable selection. Motivated by this, Zhang et al. (2008) introduced the supnorm penalty which is a good fit for multi-class classification and leads to a more sparse and accurate model than $L_1$ MSVM.

To retain the merits from both SCAD and supnorm, we introduce a new penalty and study it in multi-class context.

Our proposed new penalty, supSCAD, has the form of

$$J_\lambda(\|\zeta\|_\infty) = \begin{cases} 
\lambda\|\zeta\|_\infty & \text{if } \|\zeta\|_\infty \leq \lambda, \\
\frac{\|\zeta\|_\infty^2 - 2a\lambda\|\zeta\|_\infty + \lambda^2}{2(a - 1)} & \text{if } \lambda < \|\zeta\|_\infty \leq a\lambda, \\
\frac{(a + 1)\lambda^2}{2} & \text{if } \|\zeta\|_\infty > a\lambda,
\end{cases} \tag{3.2}$$

where $\zeta = (\zeta_1, \ldots, \zeta_d)^T$ and $\|\zeta\|_\infty = \max_{i=1,\ldots,d} |\zeta_i|$.

Logistic regression is an essential tool for the analysis in binary classification. Sometimes, it is criticized due to the difficulties of estimating a large number of parameters with a relatively small number samples. However, logistic regression has its own strength in estimating underlying conditional class probability. Moreover, it is natural to extend binary logistic regression to multinomial logistic regression.

In binary classification, the logit model is

$$\log \frac{P(y = 1|X = x)}{P(y = 2|X = x)} = x^T \beta.$$ 

The category of $\{y = 2\}$ is chosen as the baseline and its corresponding coefficients are set to be zero. Therefore, supSCAD is the same as SCAD penalty when $K = 2$. 

27
When the categorical response variable $y \in \{1, \ldots, K\}, K > 2$, we generalize the regularized multinomial logistic regression by choosing a symmetric formation in Zhu and Hastie (2004), which models the posterior probabilities as:

$$P(y = l | X = x) = \frac{e^{\beta_l^T x}}{\sum_{k=1}^K e^{\beta_k^T x}}, \quad l = 1, \ldots, K. \quad (3.3)$$

Since the parameters $\{\beta_l\}_{l=1}^K$ give identical probabilities (3.3) when adding any constant value across the $K$ classes, a sum-to-zero constraint is imposed to make it identifiable,

$$\sum_{k=1}^K \beta_{k0} = 0, \quad \text{and} \quad \sum_{k=1}^K \beta_{jk} = 0, \quad j = 1, \ldots, d. \quad (3.4)$$

And the log odds are essentially

$$\log \frac{P(y = l | x)}{P^*(x)} = \beta_l^T x$$

where

$$P^*(x) = \sqrt{\frac{1}{K} \prod_{k=1}^K P(y = k | x)} = \frac{\sqrt{e^{\sum_{k=1}^K \beta_k^T x}}}{\sum_{k=1}^K e^{\beta_k^T x}}.$$

Thus, we have our supSCAD regularized multinomial logistic regression as:

$$\min_{\beta} -\{\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K I(y_i = k) \cdot \beta_k^T x_i - \log(\sum_{k=1}^K \exp(\beta_k^T x_i))\} + \sum_{j=1}^d J_\lambda(\|\beta_{(j)}\|_\infty)$$

subject to $\sum_{k=1}^K \beta_{kj} = 0, \quad j = 0, 1, \ldots, d. \quad (3.5)$
3.2 Computation Algorithm

3.2.1 Quadratic Approximation for Log-likelihood

Since the negative log-likelihood in (3.5)

$$n^{-1} \mathcal{L}_n(\beta) = -\left\{ \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i = k) \cdot \beta_k^T x_i - \log \left( \sum_{l=1}^{K} \exp(\beta_l^T x_i) \right) \right\}$$

is nonlinear in the explanatory variable $x$, direct LP/QP technique cannot apply. Minimization can be approached by obtaining a quadratic approximation to the loss function based on the second order Taylor expansion.

Baseline logistic regression is an unconstrained convex optimization problem with a continuously differentiable objective function. It can be solved by Newton’s method through iteratively reweighed least squares (IRLS). In every iteration, Newton’s method finds a step direction by approximating the score function of negative log-likelihood with the first order Taylor expansion at the current point, and optimizing it in closed-form.

We propose an algorithm based on the IRLS. If the current estimates of the parameters is $\tilde{\beta}$, we use a quadratic approximation to the negative log-likelihood (Taylor expansion about current estimates) as:

$$n^{-1} \mathcal{L}_n(\beta) \approx n^{-1} \mathcal{L}_n(\tilde{\beta}) + n^{-1} \dot{\mathcal{L}}_n(\tilde{\beta})^T (\beta - \tilde{\beta}) + \frac{1}{2} (\beta - \tilde{\beta})^T \left\{ \frac{1}{n} \ddot{\mathcal{L}}_n(\tilde{\beta}) \right\} (\beta - \tilde{\beta})$$

(3.6)

where $\dot{\mathcal{L}}_n(\cdot)$ and $\ddot{\mathcal{L}}_n(\cdot)$ are the first and second order derivatives of the negative log-likelihood function. Then update is obtained by minimizing (3.6) subject to the constraint.

Our computation for supSCAD logistic regression involves two loops. First, an outer loop computes the quadratic approximation $Q$ about the current parameters $\tilde{\beta}$. Then, an inner loop is to execute optimization iterations for the supSCAD penalized quadratic problem.

1. **OUTER LOOP**: Compute the quadratic approximation $Q$ using Taylor expansion around the current parameters.

2. **INNER LOOP**: Solve the supSCAD problem with updated $Q$. 

29
3.2.2 Optimization Schemes for supSCAD Logistic Regression

Since supSCAD penalty is a non-convex and non-differentiable function, it is hard to solve exactly. To tackle this kind of difficulty, various methods have been proposed in different contexts. A unified least quadratic approximation (LQA) algorithm was proposed in Fan and Li (2001) to solve the SCAD penalized likelihood optimization problem. Zhang et al. (2006) proposed the successive quadratic algorithm (SQA) to convert SCAD SVM into a series of easily solved linear equation systems. These two algorithms both share the drawback of backward stepwise variable selection: if a covariate is deleted at any step in the LQA/SQA algorithm, it will necessarily be excluded from the final selected model. In addition, they require small coefficients to be deleted for the reason of numerical stability. Noticing that the SCAD penalty function can be decomposed as the difference of two convex functions, Wu and Liu (2009) proposed to solve the corresponding optimization using the Difference Convex Algorithm (DCA) in quantile regression. Addressing the drawbacks of LQA, Zou and Li (2008) developed a local linear approximation algorithm (LLA) to solve the non-concave penalized likelihood models. DCA and LLA naturally produce a sparse estimate via continuous penalization and turn out to be two instances of MM algorithm.

In this section we show that our supSCAD penalized logistic regression can be converted to a series of quadratic programming (QP) problems by using DCA and LLA, and therefore can be solved using standard QP techniques in polynomial time. This great computational advantage is very important in real applications, especially for large data sets.

DCA optimization

Consider a nonconvex minimization problem with the form of

$$\min C(\theta) = C_{\text{vex}, 1}(\theta) - C_{\text{vex}, 2}(\theta)$$

where $C_{\text{vex}, 1}(\theta)$ and $C_{\text{vex}, 2}(\theta)$ are both convex functions. An and Tao (1997) proposed DC programming to iteratively solve

$$\min C_{\text{vex}, 1}(\theta) - C'_{\text{vex}, 2}(\tilde{\theta})(\theta - \tilde{\theta})$$
with a current solution $\tilde{\theta}$. They proved that DC programming has the finite convergence to a local minimum.

Wu and Liu (2009) noticed that the first order derivative of the SCAD penalty function on $(0, +\infty)$ is the sum of two components: the first is a constant and the second is a decreasing function on the range $(0, +\infty)$. Thus, the SCAD penalty function can be decomposed as the difference of two convex functions. Explicitly, it has the form of $J_\lambda(\theta) = J_{\lambda,1}(\theta) - J_{\lambda,2}(\theta)$ where both $J_{\lambda,1}(\theta)$ and $J_{\lambda,2}(\theta)$ are convex functions and their first derivatives for $\theta > 0$ are given by

$$
\begin{align*}
J'_{\lambda,1}(\theta) &= \lambda \\
J'_{\lambda,2}(\theta) &= \lambda(1 - \frac{(a\lambda - \theta_+)}{(a-1)\lambda})I(\theta > \lambda).
\end{align*}
$$

Using the DC programming, the SCAD penalty is approximated by a difference of convex functions and leads to an efficient DC Algorithm (DCA).

DCA minimizes a non-convex objective function by solving a sequence of convex minimization problems. At each iteration, it approximates the second convex function by a linear function. As a result, the objective function at each step is convex and much easier to optimize than the original non-convex problem. DCA is a local and descent method with different decompositions in each iteration. It leads to a successive linear programming algorithm with finite convergence.

Applying DCA to supSCAD penalty, the non-convex function can be decomposed as:

$$
\sum_{j=1}^{d} J_\lambda(\|\beta_{(j)}\|_\infty) = \sum_{j=1}^{d} J_{\lambda,1}(\|\beta_{(j)}\|_\infty) - \sum_{j=1}^{d} J_{\lambda,2}(\|\beta_{(j)}\|_\infty)
$$

$$
= \lambda \sum_{j=1}^{d} \|\beta_{(j)}\|_\infty - \sum_{j=1}^{d} J'_{\lambda,2}(\|\beta_{(j)}^{(0)}\|_\infty) \cdot (\|\beta_{(j)}\|_\infty - \|\beta_{(j)}^{(0)}\|_\infty)
$$

for $\beta_{(j)} \approx \beta_{(j)}^{(0)}$, $j = 1, \ldots, d$.

Denote the solution at step $t$ by $\beta^{(t)} = (\beta_1^{(t)}, \ldots, \beta_K^{(t)})^T$. Repeatedly solving a series of decomposed convex functions, we have the following algorithm:

1. **Initialization:** $\beta^{(0)}$, set $t = 0$.  

31
2. Repeat:

\[
\beta^{(t+1)} = \arg \min_{\beta} (Q(\beta) + \sum_{j=1}^{d} \lambda \| \beta_{(j)} \|_\infty - \sum_{j=1}^{d} J'_{\lambda,2}(\| \beta_{(j)}^{(t)} \|_\infty)(\| \beta \|_\infty - \| \beta^{(t)} \|_\infty))
\]

\[
= \arg \min_{\beta} (Q(\beta) + \lambda \sum_{j=1}^{d} \| \beta_{(j)} \|_\infty - \sum_{j=1}^{d} J'_{\lambda,2}(\| \beta_{(j)}^{(t)} \|_\infty) \cdot \| \beta \|_\infty),
\]

s.t. \[ \sum_{k=1}^{K} \beta_{kj} = 0, \quad j = 0, 1, \ldots, d, \]

\[ t = t + 1. \]

3. **Stop:** \( \beta^{(t+1)} \) satisfies certain stopping criterion.

**LLA optimization**

Zou and Li (2008) proposed a new unified algorithm based on local linear approximation (LLA) to concave penalty functions,

\[
J_\lambda(|\theta_j|) \approx J_\lambda(|\theta_j^{(0)}|) + J'_\lambda(|\theta_j^{(0)}|)(|\theta_j| - |\theta_j^{(0)}|) \quad \text{for} \quad \theta_j \approx \theta_j^{(0)}. \tag{3.7}
\]

They demonstrated that LLA is the best convex MM algorithm which has the descent property for minimization and convergency (Lange et al. (2000)). Similar to DCA, LLA iteratively solves supSCAD penalty as:

1. **Initialization:** \( \beta^{(0)} \), set \( t = 0 \).

2. **Repeat:**

\[
\beta^{(t+1)} = \arg \min_{\beta} (Q(\beta) + \sum_{j=1}^{d} J'_\lambda(\| \beta_{(j)}^{(t)} \|_\infty) \cdot \| \beta_{(j)} \|_\infty)
\]

s.t. \[ \sum_{k=1}^{K} \beta_{kj} = 0, \quad j = 0, 1, \ldots, d, \]

\[ t = t + 1. \]

3. **Stop:** \( \beta^{(t+1)} \) satisfies certain stopping criterion.
Now we give the detailed iterative procedure for supSCAD logistic regression in Algorithm 1.

**Algorithm 1**

1. Initialize $\beta^{(0)}$.
2. for (t = 0 to MaxIteration)
   1. Compute $Q(\beta^{(t)})$ in (3.6).
   2. Use DCA or LLA to solve the supSCAD penalized problem.
   3. Let the solution be $\beta^{(t+1)}$.
   4. Evaluate the objective function given in (3.5) at $\beta^{(t+1)}$.
   5. if (the stopping criterion is satisfied)
      1. Break;
   end
end

**Discussion**

1. **Choice of Initial Points**

Since the supSCAD penalty is non-convex and both DCA and LLA are local algorithms, neither of them are guaranteed to a global minimum in general. Thus choosing the initial point is critical. Denote the sample size in each class as $n_k$, $k = 1, \ldots, K$, and the input variable has dimension $d$. From our experience, if $n_k \gg d$ or the unregularized maximum likelihood converges well, then the solution from unregularized multinomial logistic regression is appropriate for initialization. If $n_k \approx d$ or the unregularized maximum likelihood does not converge, then the origin is always a choice for initialization.

2. **Stopping Rule**

As is the case for traditional iteratively reweighed least squares algorithm, our iterative quadratic approximation method is not guaranteed to converge for logistic regression. However, we did not encounter any divergent problems so far. Most of time, the quadratic approximation converges very quickly in less than 10 iterations. Our code uses MaxIteration= 50. The stopping criterion we used to determine when to stop the iteration is how close the current estimation (denote as $\beta^{(t)} = (\beta_1^{(t)}, \ldots, \beta_K^{(t)})^T$) is to
the estimation one step back (denote as $\beta^{(t-1)} = (\beta_1^{(t-1)}, \ldots, \beta_K^{(t-1)})^T$). We define the stopping criterion as:

$$\sum_{k=1}^{K} \|\beta_k^{(t)} - \beta_k^{(t-1)}\|_2 < \tau.$$ 

In our algorithm, $\tau$ is $10^{-4}$.

### 3. DCA and LLA Comparison

Basically, DCA and LLA are two instances of local algorithms using linear approximation. They share similar convergence result and computation cost. Based on our empirical experience, DCA and LLA perform comparably in both computation accuracy and efficiency. In all the numerical studies presented in next section, over a total of 100 simulation runs, DCA and LLA gave almost identical simulation results and running time.

#### 3.3 Extensions to Nonlinear Classification

When the training data are impossible to be linearly separated in the original input space and a nonlinear classifier is more plausible, we map the data from the original input space $R^d$ to a higher dimensional feature space, $\mathcal{H}$, and find a linear classifier in the enlarged space.

One way for such mapping is using some basis functions $\{h_m(x), m = 1, \ldots, M\}$. The new features after mapping, $h(x_i) = (h_1(x_i), \ldots, h_M(x_i))^T$, are used to fit logistic regression. Then the decision function for class $k$ in the feature space is $\hat{f}_k(x_i) = h(x)^T\hat{\beta}_k + \hat{\beta}_{k0}$, which is nonlinear for $x$, but linear for $h(x)$. And $h(x)$ in the feature space can be treated as $x$ in the original input space. Therefore, one can replace $x$ by $h(x)$ in all the formulation that is derived for linear logistic regression and the fitting proceeds the same as in the original space.

Alternatively nonlinear classifiers can be achieved by applying the kernel trick. However, the kernel function is the dot product of $< h(x_i), h(x_j) >$, which is the summation of the data from all dimensions. Thus, the resulting classifier is like a “black box” which cannot provide a clear explicit interpretation of the importance of each variable to the final classifier.
Therefore we will only use the basis expansion to construct nonlinear classifiers in this paper.

3.4 Asymptotic Properties

We will study the asymptotic properties of the new estimator in this section. For any variable selection procedure, oracle properties (Fan and Li, 2001) contain two parts: selection consistency and asymptotic normality with the variance as if the true model were known. Adaptive LASSO and SCAD regression are all oracle procedures. Here we will show that our new supSCAD logistic regression is also an oracle procedure. In the following, Theorem 1 shows that the new estimator is root-n consistent. Theorem 2 contains two parts: first, it is showed that such a root-n consistent estimator correctly selects predictors with nonzero coefficients and excludes those with zero coefficients with probability approaching to one; second it estimates those nonzero coefficients with the asymptotic distribution they would have if all the zero coefficients were known in advance.

Before establishing the oracle properties of supSCAD penalized multinomial logistic regression, we need to lay out some basic assumption on our data. We assume that the data \(((x_1, y_1), \ldots, (x_n, y_n))\) consist of \(n\) independent and identically distributed observations from the symmetric multinomial logistic regression model (3.3)

\[
p_k(x_i) = P(y_{ik} = 1 | x_i) = \frac{\exp(x_i^T \beta_{ik})}{\sum_{j=1}^K \exp(x_i^T \beta_{ij})}
\]

\[s.t. \sum_{i=1}^K \beta_{ij} = 0, j = 0, 1, \ldots, d,
\]

with true parameters

\[
\beta^* = \begin{pmatrix}
\beta_{10}^* & \beta_{11}^* & \cdots & \beta_{1d}^* \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{K,0}^* & \beta_{K,1}^* & \cdots & \beta_{K,d}^*
\end{pmatrix}
\]
The log-likelihood and corresponding penalized log-likelihood are:

\[
L(\{\beta_k\}^K_1) = \log \prod_{i=1}^{n} \prod_{k=1}^{K} y_{ik} p_k(x_i) \\
= \sum_{i=1}^{n} \left( \sum_{k=1}^{K} y_{ik} (x_i^T \beta_k) - \log \left( \sum_{l=1}^{K} e^{x_i^T \beta_l} \right) \right)
\]  
(3.8)

\[
R(\{\beta_k\}^K_1) = -L(\{\beta_k\}^K_1) + n \sum_{j=1}^{d} J_\lambda(\|\beta_{(j)}\|_\infty) \\
s.t. \sum_{k=1}^{K} \beta_{kj} = 0, j = 0, 1, \ldots, d.
\]  
(3.9)

Assume that there are \(s + 1\) non-zero columns and \(d - s\) zero columns in \(\beta^*\). Without losing generality, we arrange the non-zero columns before the zero columns in \(\beta^*\), and denote them by \(\beta^*_+\) and \(\beta^*_z\). Thus, the true parameters are

\[
\beta^* = \begin{pmatrix} 
\beta^*_{10} & \beta^*_{11} & \cdots & \beta^*_{1,s+1} & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\beta^*_{K,0} & \beta^*_{K,1} & \cdots & \beta^*_{K,s+1} & 0 & \cdots & 0 
\end{pmatrix} = (\beta^*_+, \beta^*_z),
\]

where \(\beta^*_z = 0_{K \times (d-s)}\) and \(\beta^*_+\) is a \(K \times (s + 1)\) non-zero matrix.

Due to the identifiability constraints \(\sum_{k=1}^{K} \beta_{kj} = 0, j = 0, 1, \ldots, d\), if we know the first \(K-1\) coefficients \(\beta_{1j}, \beta_{2j}, \ldots, \beta_{K-1,j}\) for a certain covariant \(x_j\), then the last coefficient \(\beta_{Kj}\) is determined by \(\beta_{Kj} = -\sum_{k=1}^{K-1} \beta_{kj}\). Thus, we define a \(K-1\) element vector \(\beta_{(j)}_{-K} = (\beta_{1j}, \ldots, \beta_{K-1,j})^T\), which can completely characterize the \(\beta_{(j)}\). And the \(K \times (d + 1)\) matrix \(\beta\) can be reduced to a \((K - 1) \times (d + 1)\) matrix:

\[
\begin{pmatrix} 
\beta_{10} & \beta_{11} & \cdots & \beta_{1d} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{K-1,0} & \beta_{K-1,1} & \cdots & \beta_{K-1,d} 
\end{pmatrix}.
\]

Furthermore, the log-likelihood \(L(\{\beta_k\}^K_1)\) in (3.8) and its penalized form in (3.9) can
be reparameterized as $\tilde{\mathcal{L}}(\{\beta_k\}_1^{K-1})$ and

$$\tilde{R}(\{\beta_k\}_1^{K-1}) = -\tilde{\mathcal{L}}(\{\beta_k\}_1^{K-1}) + n \sum_{j=1}^d J_\lambda(\max\{|\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{(K-1),j}|, |\sum_{l=1}^{K-1} \beta_{lj}|\}).$$  \hfill (3.10)

Note that (3.9) and (3.10) are equivalent and they basically solve the same problem. For the true parameter matrix $\beta^*$, its counterpart for $\tilde{\mathcal{L}}(\{\beta_k\}_1^{K-1})$ is

$$\begin{pmatrix}
\beta_{10}^* & \beta_{11}^* & \cdots & \beta_{1,s+1}^* & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\beta_{K-1,0}^* & \beta_{K-1,1}^* & \cdots & \beta_{K-1,s+1}^* & 0 & \cdots & 0
\end{pmatrix} = (\beta_+^*, \beta_z^*),$$

where $\beta_+^*$ is reduced to a $(K - 1) \times (s + 1)$ nonzero matrix and $\beta_z^*$ is reduced to a $(K - 1) \times (d - s)$ zero matrix.

In the remaining of this section, we will show that our proposed supSCAD estimator possess oracle properties with a proper choice of tuning parameter, using the reparameterized penalized multinomial logit model (3.10). With a slight abuse of notation, we reformulate the matrix $\beta$ and $\beta^*$ as vectors

$$\beta = \begin{pmatrix}
(\beta_{(0),-K}^*) \\
(\beta_{(1),-K}^*) \\
\vdots \\
(\beta_{(d),-K}^*)
\end{pmatrix}, \quad \beta^* = \begin{pmatrix}
(\beta_{(0),-K})^* \\
(\beta_{(1),-K})^* \\
\vdots \\
(\beta_{(d),-K})^*
\end{pmatrix} = \begin{pmatrix}
\beta_{10}^* \\
\vdots \\
\beta_{K-1,0}^* \\
\beta_{1d}^* \\
\vdots \\
\beta_{K-1,d}^*
\end{pmatrix} = (\beta_+^*, \beta_z^*). \hfill (3.11)$$

Denote $I(\beta^*)$ as the Fisher information matrix of $\tilde{\mathcal{L}}$ at $\beta^*$, and $I((\beta_+^*, \beta_z^*))$ as the Fisher information matrix knowing $\beta_z^* = 0$. Let $\hat{\beta} = \begin{pmatrix} \hat{\beta}_+ \\ \hat{\beta}_z \end{pmatrix}$ be a local minimizer to (3.10), where $\hat{\beta}_+$ consists of the first $(K - 1) \times (s + 1)$ element of $\hat{\beta}$ and $\hat{\beta}_z$ consists the last $(K - 1) \times (d - s)$ elements of $\hat{\beta}$.

**Theorem 1** ($\sqrt{n}$-Consistency): Consider a sample $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ from multinomial logit model (3.3) satisfying regularity conditions in A.1. If $\lambda_n \to 0$, then there exists a local minimizer $\hat{\beta}$ such that $\|\hat{\beta} - \beta^*\| = O_p(n^{-1/2})$.  

37
In Theorem 1, it is stated that the supSCAD penalized likelihood estimator is root-n consistent. And such root-n consistent estimator possess the sparsity property $\hat{\beta}_z = 0$ under some further conditions as shown in Lemma 1.

**Lemma 1 (Sparsity):** Consider a sample $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ from multinomial logit model (3.3) satisfying regularity conditions in A.1. If $\lambda_n \to 0$ and $\sqrt{n}\lambda_n \to \infty$ as $n \to \infty$, then with probability tending to 1, for any given $\beta_+$ satisfying $||\beta_+ - \beta_+^*|| = O_p(n^{-1/2})$ and any constant $C$,

$$\hat{R}(\beta_+^*) = \min_{||\beta_+|| \leq Cn^{-1/2}} \hat{R}(\beta_+)$$.

**Theorem 2 (Oracle):** Consider a sample $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ from multinomial logit model (3.3) satisfying regularity conditions in A.1. If $\lambda_n \to 0$ and $\sqrt{n}\lambda_n \to \infty$ as $n \to \infty$ then with probability tending to 1, the root-n consistent local minimizer $(\hat{\beta}_+, \hat{\beta}_z)$ in Theorem 1 satisfy:

1. (Sparsity): $\hat{\beta}_z = 0$.
2. (Asymptotic normality): $\sqrt{n}(\hat{\beta}_+ - \beta_+^*) \to N(0, I^{-1}(\beta_+^*))$.

### 3.5 Simulation Studies

In this section, we examine the finite sample performance of the proposed supSCAD estimator in terms of variable selection, class prediction and probability estimation. Six simulation examples are studied. The five simulation experiments are: (1) a four-class linear example with the same two important variables for all the classes; (2) another four-class linear example with same two important variables and two additional important yet weak variables for all the classes; (3) a five-class linear example with the same two important variables for all the classes; (4) a four-class linear example with different important variables from one class to another; (5) another four-class linear example with different important variables from one class to another; (6) a three-class non-linear example with same important variables across the classes. For each experiment, a training data set of size $n$ and a test data set of size $n'$ are randomly generated in the same way. The training data are used to train classifiers and select the best tuning parameter from a series of $\lambda$. The test data are used to estimate the prediction performance of the final classifier.
Similar to other types of penalized procedure, the tuning parameter $\lambda$ is very important in supSCAD regularization. It controls the trade off between training error and generalization, and determines the number of variables used in the classifier. To choose an appropriate $\lambda$, we use a BIC-type selection criterion as,

$$BIC = -\frac{2}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i = k) \log \hat{p}_k(x_i) + \frac{1}{n} \log(n) \times df,$$

where $df$ is the number of nonzero coefficients in $\hat{\beta}$. Searched over a grid: $\log_2(\lambda) = -10, -9, \ldots, 10$, the optimum $\lambda$ is identified with the least BIC score from (3.12). When a tie occurs, the larger value of $\lambda$ would be chosen. Test data are used to examine the prediction accuracy of the final classifier.

According to Zou et al. (2007), the number of nonzero coefficients is an unbiased estimate for the degrees of freedom of the lasso. Zhao et al. (2009) extended the result and derived an unbiased estimate of the degrees of freedom of the regularized estimates using the composite absolute penalties (CAP) and the $L_2$ loss. The CAP families they introduced allow grouped and hierarchical variable selection and combine different norms including $L_1$. Especially, a subfamily of CAP estimates involving only the $L_1$ and $L_1$ norms, called iCAP, has piecewise linear regularization path and is computationally convenient. They provided an algorithm to trace the entire regularization path for the grouped selection problem. And using the unbiased estimate of the degrees of freedom with information theory criteria, one can pick an optimal fit without the use of cross validation. However, to estimate degrees of freedom in broader settings other than $L_2$ loss and $L_1$ or $L_\infty$ norms, still remains for future research. We adopt the result from Zou et al. (2007) here as an approximated solution.

In each simulation, the true conditional probability $p_k(x_i), i = 1, \ldots, n', k = 1, \ldots, K$, are known. To measure the estimation accuracy of the conditional probabilities, we use three criteria evaluated on the testing set as in Wu et al (2010):

- **P1 error** $\frac{1}{n'} \sum_{i=1}^{n'} \sum_{k=1}^{K} |\hat{p}_k(x_{i, test}) - p_k(x_{i, test})|$.  
- **P2 error** $\frac{1}{n'} \sum_{i=1}^{n'} \sum_{k=1}^{K} (\hat{p}_k(x_{i, test}) - p_k(x_{i, test}))^2$.  
- **Empirical generalized Kullback – Leibler (EGKL) loss** $\frac{1}{n'} \sum_{i=1}^{n'} \sum_{k=1}^{K} p_k(x_{i, test}) \log \frac{p_k(x_{i, test})}{\hat{p}_k(x_{i, test})}$.  

39
Every classifier consists of $K$ classification functions, each of which associates with $d$ covariates. Therefore a total of $Kd$ coefficients are estimated in each model. To compare the variable selection performance, we use the following criteria.

- **Correct Zero** is the number of zero estimates which are truly zero.
- **Incorrect Zero** is the number of zero estimates which are truly nonzero.
- **Model Size** is the number of covariates which are selected in the final model.
- **Correct Model** is the frequency of selecting the correct model.

A total of 100 simulations are conducted for each procedure under all settings. Each fitted classifiers is then evaluated in terms of probability estimation and classification accuracy, as well as variable selection performance. For each method, its average testing error, $P1$, $P2$, EGKL, the number of correct and incorrect zeros, the model size, and the number of times that the true model is correctly identified are summarized in tables.

Five different procedures are considered. They are logistic regression (LR), $L_1$ logistic regression ($L_1$ LR), supSCAD logistic regression (supSCAD LR), $L_1$ LR followed by the LR ($L_1$ LR2) and supSCAD LR followed by the LR (supSCAD LR2). In addition, we include the Bayes rule as a reference in comparison.

### 3.5.1 Four-Class Linear Example 1

Consider a four-class example with 20-dimensional input vector $x$. For each class $k = 1, 2, 3, 4$, the first two components of $x$ are generated from $N(\mu_k, 2I_2)$, where $\mu_1 = (\sqrt{2}, \sqrt{2})^T$, $\mu_2 = (-\sqrt{2}, \sqrt{2})^T$, $\mu_3 = (-\sqrt{2}, -\sqrt{2})^T$, $\mu_4 = (\sqrt{2}, -\sqrt{2})^T$ and $I_2$ is a $2 \times 2$ identity matrix. The remaining eighteen components are i.i.d. generated from $N(0, 1)$. Here the sample size is $n = 200$ for training data, and $n' = 40,000$ for testing data. Evidently, only the first 2 components of $x$ are relevant to classification, whereas the remaining 18 components are redundant. Figure 3.1 is a plot of a randomly chosen training set. The solid lines are the Bayes boundaries.

Table 3.1 presents the results for five logistic regression variants. Entries in the last four columns have standard errors in the range of 0.001 to 0.01. The performance from Bayes method is listed on the last line as a reference to the best we can expect.

In terms of model selection criteria such as Correct Zero, Incorrect Zero, Model Size, and Correct Model, supSCAD LR achieves the best results. In each run of the simulation
Figure 3.1: The Bayes boundary for the four-class example.
supSCAD penalty is able to select the important variables correctly while exclude the noise variables as often as possible. Over the 100 simulation runs, supSCAD LR identifies the exact true model 96 times and get an average model size of 2.04 which is slightly larger than the oracle model.

As for the classification accuracy, supSCAD LR(2), L1LR all get the smallest average testing error of 0.30 which is very close to the 0.29 from Bayes method. This excellent result is not surprising, since under this example setting logistic regression is the oracle model.

With regard to the probability estimation, supSCAD LR gets smaller values of P1, P2 and EGKL than L1LR. And a two stage method can improve it even more in both situations of L1LR and supSCAD LR. As a result, supSCAD LR2 obtains the lowest P1, P2 and EGKL.

Overall, supSCAD LR2 gives the most satisfactory performance.

### 3.5.2 Four-Class Linear Example 2

Consider a similar four-class example with the last one. This time, the 20-dimensional input vector $x$ contains two important variables and another two important but weak variables. For each class $k = 1, 2, 3, 4$, the first two components of $x$ are generated from $N(\mu_k, 3I_2)$, where $\mu_1$, $\mu_2$, $\mu_3$, and $\mu_4$ are the same as in the last example. And the next two components are generated from $N(\nu_k, I_2)$, where $\nu_1 = (0.5, 0.5)^T$, $\nu_2 = (-0.5, 0.5)^T$, $\nu_3 = (0.5, -0.5)^T$, $\nu_4 = (-0.5, -0.5)^T$. The remaining sixteen components are i.i.d. generated from $N(0, 1)$. Training sample size and testing sample size are still $n = 200$, and $n' = 40,000$. In other words, the first 2 components of $x$ are strong signals to classification, and the second 2 components are weak signals, whereas the remaining 16 components are redundant noises.
Table 3.2: Simulation results for four-class linear example 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
<th>P1</th>
<th>P2</th>
<th>EGKL</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0.60</td>
<td>0.22</td>
<td>0.68</td>
<td>0.36</td>
</tr>
<tr>
<td>L1 LR</td>
<td>59.61</td>
<td>0.45</td>
<td>9.89</td>
<td>10</td>
<td>0.36</td>
<td>0.06</td>
<td>0.12</td>
<td>0.32</td>
</tr>
<tr>
<td>L1 LR2</td>
<td>40.41</td>
<td>0.04</td>
<td>9.89</td>
<td>10</td>
<td>0.31</td>
<td>0.07</td>
<td>0.16</td>
<td>0.34</td>
</tr>
<tr>
<td>supSCAD LR</td>
<td>61.92</td>
<td>0.72</td>
<td>4.34</td>
<td>70</td>
<td>0.24</td>
<td>0.04</td>
<td>0.07</td>
<td>0.32</td>
</tr>
<tr>
<td>supSCAD LR2</td>
<td>61.92</td>
<td>0.72</td>
<td>4.34</td>
<td>70</td>
<td>0.21</td>
<td>0.03</td>
<td>0.07</td>
<td>0.32</td>
</tr>
<tr>
<td>Bayes rule</td>
<td>64</td>
<td>0</td>
<td>4</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Results shown in table 3.2 are not as good as in table 3.1 due to the increasing difficulty of this problem. Entries in the table are based on 100 replications and the values in the last four columns have standard errors in the range of 0.001 to 0.02. From the nonzero values in the column of “Incorrect Zeros”, all the penalized methods miss the two weak signals sometimes. However supSCAD LR still identifies the true model most frequently, 70 out of 100, and achieves the smallest model size of 4.34. In classification, L1 LR, supSCAD LR and supSCAD LR2 get the same lowest error rate of 0.30. In probability estimation, supSCAD LR2 is still the best.

3.5.3 Five-Class Linear Example

The setting of this five-class example is similar to the first one, except that the five centers are

$$\mu_k = 2(\cos([2k-1]\pi/5), \sin([2k-1]\pi/5))^T, k = 1, 2, 3, 4, 5.$$  

And the input $x$ is 10-dimensional, with $n = 250$, $n' = 50,000$. Table 3.2 presents the results.

Table 3.3: Simulation results for the five-class linear example.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
<th>P1</th>
<th>P2</th>
<th>EGKL</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0.30</td>
<td>0.05</td>
<td>0.11</td>
<td>0.42</td>
</tr>
<tr>
<td>L1 LR</td>
<td>35.94</td>
<td>2.82</td>
<td>5.11</td>
<td>4</td>
<td>0.29</td>
<td>0.03</td>
<td>0.08</td>
<td>0.40</td>
</tr>
<tr>
<td>L1 LR2</td>
<td>24.85</td>
<td>2.82</td>
<td>5.11</td>
<td>4</td>
<td>0.23</td>
<td>0.03</td>
<td>0.07</td>
<td>0.41</td>
</tr>
<tr>
<td>supSCAD LR</td>
<td>40</td>
<td>0</td>
<td>2</td>
<td>100</td>
<td>0.13</td>
<td>0.01</td>
<td>0.02</td>
<td>0.39</td>
</tr>
<tr>
<td>supSCAD LR2</td>
<td>40</td>
<td>0</td>
<td>2</td>
<td>100</td>
<td>0.12</td>
<td>0.01</td>
<td>0.02</td>
<td>0.39</td>
</tr>
<tr>
<td>Bayes rule</td>
<td>40</td>
<td>0</td>
<td>2</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.39</td>
</tr>
</tbody>
</table>
Entries in the last four columns have standard errors in the range of 0.0004 to 0.005. In this example, supSCAD LR and supSCAD LR2 performed identically well like the oracle model. For L1 LR, its model selection and probability estimation performance were much worse and obviously beaten by supSCAD LR.

### 3.5.4 Four-Class Linear Example 3

In the examples above, the important variables are the same for all the classes. However in reality, there may be some cases where the important variables are different from class to class. Consider a 10-dimension input $x$, the first four elements $x_1, x_2, x_3, x_4$ are generated i.i.d from Unif$[-1, 1]$ and the last six elements $x_5, \ldots, x_{10}$ are generated i.i.d from $N(0, 8^2)$. Define the functions

\[
\begin{align*}
    f_1 &= -5x_1 + 5x_4, \\
    f_2 &= 5x_1 + 5x_2, \\
    f_3 &= -5x_2 + 5x_3, \\
    f_4 &= -5x_3 - 5x_4,
\end{align*}
\]

and set $p_k(x) = P(Y = k|X = x) \propto exp(f_k(x))$, $k = 1, 2, 3, 4$. In this example, we have $n = 200$ and $n' = 40,000$. Although the first 4 variables are important variables, $x_1$ is not important for distinguishing class 3 and class 4, $x_2$ is noninformative for distinguishing class 1 and class 4, $x_3$ is noninformative for distinguishing class 1 and class 2, and $x_4$ is noninformative for distinguishing class 2 and class 3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
<th>P1</th>
<th>P2</th>
<th>EGKL</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0.23</td>
<td>0.07</td>
<td>0.19</td>
<td>0.18</td>
</tr>
<tr>
<td>L1 LR</td>
<td>26.72</td>
<td>0</td>
<td>7.01</td>
<td>9</td>
<td>0.28</td>
<td>0.05</td>
<td>0.10</td>
<td>0.15</td>
</tr>
<tr>
<td>L1 LR2</td>
<td>11.98</td>
<td>0</td>
<td>7.01</td>
<td>9</td>
<td>0.20</td>
<td>0.05</td>
<td>0.13</td>
<td>0.17</td>
</tr>
<tr>
<td>supSCAD LR</td>
<td>23.84</td>
<td>0</td>
<td>4.04</td>
<td>98</td>
<td>0.14</td>
<td>0.03</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>supSCAD LR2</td>
<td>23.84</td>
<td>0</td>
<td>4.04</td>
<td>98</td>
<td>0.14</td>
<td>0.03</td>
<td>0.06</td>
<td>0.15</td>
</tr>
<tr>
<td>Bayes rule</td>
<td>32</td>
<td>0</td>
<td>4</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Our proposed supSCAD penalty treats all coefficients of one variable corresponding
to different classes as a group and removes the variable if it is non-informative across all class labels. By design of this example, important variables have zero coefficients for certain classes. As a result, $L_1$ penalty is more appropriate and should give better result. However, supSCAD LR still gave satisfactory results especially in variable selection. Entries in the last four columns of table 3.4 have standard error in the range of 0.001 to 0.008.

### 3.5.5 Four-Class Linear Example 4

Here we consider a similar example to the last one, where the important variables are different from class to class. But this time, we have a 20-dimension input $x$, the first four elements $x_1, x_2, x_3, x_4$ are generated uniformly from a 4-dimension ball with radius of 3, i.e. $\{(x_1, x_2, x_3, x_4)^T \in R^4 : \sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2} \leq 3\}$. The remaining $x_5, \ldots, x_{20}$ are generated i.i.d from $N(0, 1)$. Define the functions

\[
\begin{align*}
    f_1 &= -3x_1 + 3x_4, \\
    f_2 &= 3x_1 + 3x_2, \\
    f_3 &= -3x_2 + 3x_3, \\
    f_4 &= -3x_3 - 3x_4,
\end{align*}
\]

and set $p_k(x) = P(Y = k|X = x) \propto exp(f_k(x)), k = 1, 2, 3, 4$. We have $n = 200$ and $n' = 40,000$. The first 4 variables are important variables but not to all the classes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
<th>P1</th>
<th>P2</th>
<th>EGKL</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0.34</td>
<td>0.17</td>
<td>0.54</td>
<td>0.21</td>
</tr>
<tr>
<td>L1 LR</td>
<td>67.75</td>
<td>0</td>
<td>7.44</td>
<td>5</td>
<td>0.38</td>
<td>0.07</td>
<td>0.17</td>
<td>0.13</td>
</tr>
<tr>
<td>L1 LR2</td>
<td>50.25</td>
<td>0</td>
<td>7.44</td>
<td>5</td>
<td>0.20</td>
<td>0.07</td>
<td>0.18</td>
<td>0.15</td>
</tr>
<tr>
<td>supSCAD LR</td>
<td>63.36</td>
<td>0</td>
<td>4.16</td>
<td>96</td>
<td>0.16</td>
<td>0.04</td>
<td>0.08</td>
<td>0.13</td>
</tr>
<tr>
<td>supSCAD LR2</td>
<td>63.36</td>
<td>0</td>
<td>4.16</td>
<td>96</td>
<td>0.14</td>
<td>0.03</td>
<td>0.08</td>
<td>0.13</td>
</tr>
<tr>
<td>Bayes rule</td>
<td>72</td>
<td>0</td>
<td>4</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.11</td>
</tr>
</tbody>
</table>

As in the last example, supSCAD LR still gives satisfactory results. It achieves best results in model selection and class prediction. Entries in the last four columns of table
3.5 have standard error in the range of 0.001 to 0.01.

3.5.6 Three-Class Nonlinear Example

Now let us consider a 3-class example which has nonlinear decision boundaries in its input variables. First generate \( x_1 \sim \text{Unif}[-3, 3] \) and \( x_2 \sim \text{Unif}[-6, 6] \). Define the functions

\[
\begin{align*}
    f_1 &= -2x_1 + 0.2x_1^2 - 0.1x_2^2 + 0.2, \\
    f_2 &= -0.4x_1^2 + 0.2x_2^2 - 0.4, \\
    f_3 &= 2x_1 + 0.2x_1^2 - 0.1x_2^2 + 0.2,
\end{align*}
\]

and set \( p_k(x) = P(Y = k|X = x) \sim \exp(f_k(x)), k = 1, 2, 3 \). Evidently the classification is defined by \( x_1, x_1^2, \) and \( x_2^2 \). Then generate five noise variables \( x_i \sim \text{N}(0, \sigma^2), \sigma = 2, i = 3, 4, 5, 6, 7 \). To achieve nonlinear classification, we use a kind of basis expansion. We fit the nonlinear LR by including the seven main effects, their square terms, and their cross products as the basis functions. This leads to a total of 35 variable terms in the model. In each simulation run, 200 samples were generated for training, and 40000 samples for testing. Figure 3.2 is a plot of a randomly chosen training set. The solid lines are the Bayes boundaries.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
<th>P1</th>
<th>P2</th>
<th>EGKL</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>0.38</td>
<td>0.21</td>
<td>0.93</td>
<td>0.22</td>
</tr>
<tr>
<td>L1 LR</td>
<td>92.74</td>
<td>3.99</td>
<td>6.91</td>
<td>20</td>
<td>0.27</td>
<td>0.04</td>
<td>0.10</td>
<td>0.13</td>
</tr>
<tr>
<td>L1 LR2</td>
<td>84.27</td>
<td>0</td>
<td>6.91</td>
<td>20</td>
<td>0.17</td>
<td>0.05</td>
<td>0.12</td>
<td>0.15</td>
</tr>
<tr>
<td>supSCAD LR</td>
<td>95.91</td>
<td>0</td>
<td>3.05</td>
<td>95</td>
<td>0.35</td>
<td>0.06</td>
<td>0.15</td>
<td>0.13</td>
</tr>
<tr>
<td>supSCAD LR2</td>
<td>95.85</td>
<td>0</td>
<td>3.05</td>
<td>95</td>
<td>0.09</td>
<td>0.01</td>
<td>0.03</td>
<td>0.13</td>
</tr>
<tr>
<td>Bayes rule</td>
<td>97</td>
<td>0</td>
<td>3</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.12</td>
</tr>
</tbody>
</table>

In this basis expansion example, we intended to use the LR solution as our initial point for supSCAD LR as we did in the previous linear examples. However, the \texttt{mnrfit} function in MATLAB kept giving “Maximum likelihood estimation did not converge” warnings and the output estimates were not reliable. So we used the origin as our initial point. Therefore, supSCAD LR did not estimate the conditional probabilities and
Figure 3.2: The Bayes boundary for the nonlinear three-class example.
decision boundaries good enough as seen from the large values of P1, P2, and EGKL. But a two procedure can fix this problem due to the high accuracy model selection of supSCAD LR. From the origin point, it still selected the true model 95 times out of 100. Overall supSCAD LR2 again gave the best results. Entries in the last four columns of table 3.6 have standard errors in the range of 0.001 to 0.01.

3.6 Real Examples

To further evaluate the performance of this newly developed approach, two real data sets are used for investigation. One is leukemia data set with 3 class levels (Guyon et al., 2002). The other is Small Round Blue Cell Tumor data with 4 class levels from Khan et al. (2001).

Oftentimes, the microarray data have extremely high dimensional gene features, a popular way for genes selection is univariate ranking. The weakness for such ranking method is that: first, the classification and variable selection are performed in two steps and second, the correlation and interaction among genes are not considered when selecting genes for building classifier. The new supSCAD method can achieve the goals of classification of patients and selection of genes simultaneously.

3.6.1 Leukemia Study

Leukemia study from Guyon et al. (2002) analyzed human bone marrow samples using oligonucleotide microarrays produced by Affymetrix. Leukemia data consist of 7129 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. The whole data is divided into a training set consisting of 38 samples, and a test set consisting of 34 samples. Table 3.7 summarizes the class distribution in each set.

Following the procedures used by Lee and Lee (2003), we preprocess the data as: (1) thresholding (floor of 100 and ceiling of 16, 000); (2) filtering (exclusion of genes with max/min ≤ 5 and max – min ≤ 500 across the samples); and (3) base-10 logarithmic transformation. The filtering results in 3,571 genes. Then we rank all genes using their marginal relevance in class separation defined in Dudoit et al. (2002). Specifically, the
relevance measure for gene \( g \) is defined to be the ratio of between classes sum of squares to within class sum of squares as follows:

\[
R(g) = \frac{\sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i = k)(\bar{x}_{(k)}^g - \bar{x}_g)^2}{\sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i = k)(x_{ig} - \bar{x}_{(k)}^g)^2},
\]

where \( n \) is the size of the training set, \( \bar{x}_{(k)}^g \) denotes the average expression level of gene \( g \) for class \( k \) observations, and \( \bar{x}_g \) is the overall mean expression level of gene \( g \) in the training set. To examine the performance of variable selection of all different methods, we select the top 100 and bottom 100 genes as covariates according to the relevance measure \( R \). Our main goal here is to get a small set of “important” genes which may provide a clue to the mechanisms related to different acute leukemia and a guideline for biologist to conduct additional confirmatory experiments.

After these preprocess, we standardize the data set by applying a simple linear transformation based on training data. Specifically, we standardize the expression \( \tilde{x}_{gi} \) of the \( g \)-th gene of subject \( i \) to obtain \( x_{gi} \) by the following formula:

\[
x_{gi} = \frac{\tilde{x}_{gi} - \frac{1}{n} \sum_{j=1}^{n} \tilde{x}_{gj}}{sd(\tilde{x}_{g1}, \ldots, \tilde{x}_{gn})}.
\]

Then various penalized logistic regression are applied to the data. We use leave-one-out cross validation on the standardized training data to select an optimal tuning parameter. All the results are summarized in table 3.8.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ALL B</th>
<th>ALL T</th>
<th>AML</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>19</td>
<td>8</td>
<td>11</td>
<td>38</td>
</tr>
<tr>
<td>Test</td>
<td>19</td>
<td>1</td>
<td>14</td>
<td>34</td>
</tr>
</tbody>
</table>

In this real example, the training sample size 38 is less than the total of genes 200. Thus logistic regression cannot be applied. And in two step procedures of penalized logistic regression, the second step is intended for improving probability estimation. However, in this real data, the true conditional probabilities are not available to justify this improvement. Thus table 3.8 does not include logistic regression, and two step L1/supSCAD
Table 3.8: Classification and variable selection results for the Leukemia data

<table>
<thead>
<tr>
<th>Method</th>
<th>L1 LR</th>
<th>supSCAD LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model size</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Test error</td>
<td>3/34</td>
<td>2/34</td>
</tr>
</tbody>
</table>

logistic regression.

Both methods select variables only from the top 100 genes. Our supSCAD LR identifies less number of important genes, which are good targets for further examination. The smaller number of selected genes can greatly save the time and costs for biologists to explore and conduct further experimental confirmation.

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 with 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 four tumor class distributions in training and test set are given in table 3.9.

Table 3.9: Class distribution of the SRBCT data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NB</th>
<th>RMS</th>
<th>BL</th>
<th>EWS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>12</td>
<td>20</td>
<td>8</td>
<td>23</td>
<td>63</td>
</tr>
<tr>
<td>Test</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>20</td>
</tr>
</tbody>
</table>

Before analyzing this example, we standardize the data set and rank all the genes in the same way we did in the last Leukemia study. The only difference here is that we do not need the preprocessing steps as in Leukemia study. Two penalized logistic regression approaches were applied to the training data of 200 genes and 63 training samples. The 200 genes include the top significant 100 genes and bottom non-significant 200 genes. Leave-one-out cross validation is used to select the appropriate tuning parameter. The learned models were used to predict the unseen 20 test samples. Results are compared in table 3.10. Both methods achieve perfect classification and exclude the non-significant
Table 3.10: Classification and variable selection results for the SRBCT data

<table>
<thead>
<tr>
<th>Method</th>
<th>L1 LR</th>
<th>supSCAD LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model size</td>
<td>31</td>
<td>35</td>
</tr>
<tr>
<td>Test error</td>
<td>0/20</td>
<td>0/20</td>
</tr>
</tbody>
</table>

100 genes from the model.
Chapter 4

New Variable Selection Method for Multi-class Support Vector Machines

Multinomial logistic regression models the log odds of $K$-class posterior probabilities via linear functions in $\mathbf{x}$, while at the same time ensuring that the posterior probabilities sum to one and remain in $(0, 1)$. On the other hand, support vector machines bypass the conditional probability and construct linear decision boundaries that explicitly separate the data into different classes as well as possible.

Binary SVM has good theoretic properties and superior practice performance, especially in classifying high-dimensional and low sample size data. To extend the binary SVM to multi-class SVM, traditional methods are to decompose a single multi-class problem into multiple independent binary problems using one-versus-one or one-versus-all approaches. But these decompositions do not consider the correlation between different classes. Further, these approaches make the variable selection difficult to be accommodated. New methods like MSVM and MPSVM are developed to consider all classes simultaneously and have better theoretical properties and empirical performance. It is now possible to learn an optimum SVM classifier by consider all classes and variables at the same time.

In this Chapter, we will implement supSCAD regularization via multicategory support vector machines and proximal support vector machines.
4.1 supSCAD MSVM and MPSVM

As reviewed in Chapter 2, although variable selection is very important and challenging in multicategory SVM, the existing work only include $L_1$ MSVM and supnorm MSVM. Our result in Chapter 3 demonstrates the promising performance of supSCAD penalty in sparse learning of multi-classification. It will be interesting to extend this penalty to other contexts of multi-classification methods.

Replacing supSCAD penalty with $L_2$ penalty in MSVM, we have supSCAD MSVM as:

$$
\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k)[\beta_{k0} + \beta_k^T x_i + 1]_+ + \sum_{j=1}^{d} J_\lambda(\|\beta_{(j)}\|_\infty)
$$

subject to $\sum_{k=1}^{K} \beta_{k0} = 0,$ $j = 1, \ldots, d.$

Tang and Zhang (2006) extended PSVM to multiclass PSVM (MPSVM) which also implements the Bayes rule asymptotically by estimating some functions of the conditional probabilities instead. When using MPSVM, we have supSCAD MPSVM,

$$
\min_{\beta_0, \beta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} I(y_i \neq k)[\beta_{k0} + \beta_k^T x_i + 1]^2 + \sum_{j=1}^{d} J_\lambda(\|\beta_{(j)}\|_\infty)
$$

subject to $\sum_{k=1}^{K} \beta_{k0} = 0,$ $j = 1, \ldots, d.$

And the K-class conditional probabilities can be estimated by (1.37).

The hinge loss in MSVM and the squared error loss in MPSVM can be handled by standard LP/QP techniques. Applying the DCA/LLA introduced in section 3.2.2, we can solve the supSCAD MSVM (4.1) and supSCAD MPSVM (4.2) through a series of LP/QP subproblems with no extra difficulty.
4.2 Computational Algorithm for supSCAD MSVM and MPSVM

Let $\Delta$ be an $n \times K$ matching matrix with its entry $\delta_{ik} = I(y_i \neq k)$ for $i = 1, \ldots, n$, $k = 1, \ldots, K$. And define a set of slack variables $\xi_{ik} = [\beta_k0 + \beta_k^T x_i + 1]_+$ for $i = 1, \ldots, n$, $k = 1, \ldots, K$.

Then supSCAD MSVM (4.1) can be written as

$$
\min_{\beta_0, \beta, \xi} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \delta_{ik} \xi_{ik} + \sum_{j=1}^{d} J_\lambda(\|\beta_{(j)}\|_\infty)
$$

subject to

$$
\sum_{k=1}^{K} \beta_k0 = 0,
$$

$$
\sum_{k=1}^{K} \beta_{kj} = 0, j = 1, \ldots, d,
$$

$$
\xi_{ik} \geq \beta_k0 + \beta_k^T x_i + 1, \xi_{ik} \geq 0, i = 1, \ldots, n \text{ and } k = 1, \ldots, K.
$$

Introduce a second set of slack variables $\eta = (\eta_1, \ldots, \eta_d)^T$ with $\eta_j = \|\beta_{(j)}\|_\infty = \max_{k=1, \ldots, K} |\beta_{kj}|$, also add some new constraints to the problem: $|\beta_{kj}| \leq \eta_j$, for $k = 1, \ldots, K$, $j = 1, \ldots, d$.

Then

$$
\min_{\beta_0, \beta, \xi, \eta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \delta_{ik} \xi_{ik} + \sum_{j=1}^{d} J_\lambda(\eta_j),
$$

subject to

$$
\sum_{k=1}^{K} \beta_k0 = 0,
$$

$$
\sum_{k=1}^{K} \beta_{kj} = 0, j = 1, \ldots, d,
$$

$$
\xi_{ik} \geq \beta_k0 + \beta_k^T x_i + 1, \xi_{ik} \geq 0, i = 1, \ldots, n \text{ and } k = 1, \ldots, K,
$$

$$
\beta_{(j)} \leq \eta_j 1_K, -\beta_{(j)} \leq \eta_j 1_K, j = 1, \ldots, d.
$$

Therefore applying DCA, the above non-convex minimization problem can be solved via
a sequence of LP subproblems.

\[
\min_{\beta_0, \beta, \xi, \eta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \delta_{ik} \xi_{ik} + \lambda \sum_{j=1}^{d} \eta_j - \sum_{j=1}^{d} J'_{X,2}(\eta_j^{(t)}) (\eta_j - \eta_j^{(t)})
\]

subject to \( \sum_{k=1}^{K} \beta_{k0} = 0, \)
\[\sum_{k=1}^{K} \beta_{kj} = 0, j = 1, \ldots, d,\]
\[\xi_{ik} \geq \beta_{k0} + \beta_k^T x_i + 1, \xi_{ik} \geq 0, i = 1, \ldots, n \text{ and } k = 1, \ldots, K\]
\[\beta_{(j)} \leq \eta_{j1K}, -\beta_{(j)} \leq \eta_{j1K}, j = 1, \ldots, d.\]  

(4.3)

This procedure of applying DCA in supSCAD MSVM is summarized in Algorithm 2:

1. **Initialization**: \( \beta_0^{(0)} = 0, \beta^{(0)} = 0, \xi^{(0)} = 0, \eta^{(0)} = 0, t = 0. \)

2. **Repeat**: solve \( \beta_0^{(t+1)}, \beta^{(t+1)}, \xi^{(t+1)} \) and \( \eta^{(t+1)} \) from (4.3), \( t = t + 1. \)

3. **Stop**: \( \beta_0^{(t+1)} \) and \( \beta^{(t+1)} \) are convergent.

Similarly, applying DCA to supSCAD MPSVM, we get a series of quadratic programming problems. Specifically, supSCAD MPSVM (4.2) is decomposed to the following QP problems:

\[
\min_{\beta_0, \beta, \xi, \eta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \delta_{ik} [\beta_{k0} + \beta_k^T x_i + 1]^2 + \lambda \sum_{j=1}^{d} \eta_j - \sum_{j=1}^{d} J'_{X,2}(\eta_j^{(t)}) (\eta_j - \eta_j^{(t)})
\]

subject to \( \sum_{k=1}^{K} \beta_{k0} = 0, \)
\[\sum_{k=1}^{K} \beta_{kj} = 0, j = 1, \ldots, d,\]
\[\beta_{(j)} \leq \eta_{j1K}, -\beta_{(j)} \leq \eta_{j1K}, j = 1, \ldots, d.\]  

(4.4)

To iteratively solve supSCAD MPSVM by DCA, we have algorithm 3:

1. **Initialization**: \( \beta_0^{(0)} = 0, \beta^{(0)} = 0, \eta^{(0)} = 0, t = 0. \)

2. **Repeat**: solve \( \beta_0^{(t+1)}, \beta^{(t+1)}, \) and \( \eta^{(t+1)} \) from (4.4), \( t = t + 1. \)
3. **Stop:** $\beta_0^{(t+1)}$ and $\beta^{(t+1)}$ are convergent.

Alternatively, using LLA, supSCAD MSVM (4.1) can also be solved via the following different set of LP subproblems:

$$
\min_{\beta_0, \beta, \xi, \eta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \delta_{ik} \xi_{ik} + \sum_{j=1}^{d} J'_\lambda(\eta_{j}^{(t)}) \cdot \eta_j
$$

subject to

$$
\sum_{k=1}^{K} \beta_{k0} = 0,
$$

$$
\sum_{k=1}^{K} \beta_{kj} = 0, j = 1, \ldots, d,
$$

$$
\xi_{ik} \geq \beta_{k0} + \beta_k^T x_i + 1, \xi_{ik} \geq 0, i = 1, \ldots, n \text{ and } k = 1, \ldots, K
$$

$$
\beta_{(j)} \leq \eta_j 1_K, -\beta_{(j)} \leq \eta_j 1_K, j = 1, \ldots, d
$$

And the supSCAD MPSVM (4.2) can be solved via a different series of quadratic programming subproblems using LLA.

$$
\min_{\beta_0, \beta, \xi, \eta} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \delta_{ik} [\beta_{k0} + \beta_k^T x_i + 1]^2 + \sum_{j=1}^{d} J'_\lambda(\eta_{j}^{(t)}) \cdot \eta_j
$$

subject to

$$
\sum_{k=1}^{K} \beta_{k0} = 0,
$$

$$
\sum_{k=1}^{K} \beta_{kj} = 0, j = 1, \ldots, d,
$$

$$
\beta_{(j)} \leq \eta_j 1_K, -\beta_{(j)} \leq \eta_j 1_K, j = 1, \ldots, d
$$

The complete procedure for solving supSCAD MSVM by LLA is described in algorithm 4:

1. **Initialization:** $\beta_0^{(0)} = 0, \beta^{(0)} = 0, \xi^{(0)} = 0, \eta^{(0)} = 0, t = 0$.

2. **Repeat:** Solve $\beta_0^{(t+1)}, \beta^{(t+1)}, \xi^{(t+1)},$ and $\eta^{(t+1)}$ from (4.5), $t = t + 1$.

3. **Stop:** $\beta_0^{(t+1)}$ and $\beta^{(t+1)}$ are convergent.

Algorithm 5 is for solving supSCAD MPSVM by LLA:
1. **Initialization**: $\beta_0^{(0)} = 0, \beta^{(0)} = 0, \eta^{(0)} = 0, t = 0$.

2. **Repeat**: Solve $\beta_0^{(t+1)}, \beta^{(t+1)}$, and $\eta^{(t+1)}$ from (4.6), $t = t + 1$.

3. **Stop**: $\beta_0^{(t+1)}$ and $\beta^{(t+1)}$ are convergent.

Basically, DCA and LLA are two instances of the MM algorithm. They share the similar convergence result and computation cost. Therefore, Algorithm 2 and Algorithm 4, Algorithm 3 and Algorithm 5, should have comparable performance.

### 4.3 Simulation Studies

In this section, we use similar simulation examples as used in section 3.5 to compare variable selection and classification performance of supSCAD MSVM/MPSVM with six different regularized MSVMs. The six MSVMs are standard MSVM (L2 MSVM), $L_1$ MSVM (L1 MSVM), Adaptive $L_1$ MSVM (Adapt-L1 MSVM), supnorm MSVM, Adaptive supnorm MSVM I (Adapt-sup MSVM I), and Adaptive supnorm MSVM II (Adapt-sup MSVM II).

All the experiments are repeated 100 times. In each simulation run, we randomly generate a training data set of size $n$ and a testing data set of size $n'$. The training data are used to select the optimum $\lambda$ and obtain the final classifier. To choose the best $\lambda$ we use five-fold cross-validation. The complete procedure is as follows: divide the full training set $T$ into the cross-validation training and test set as $T_{CV_v}$ and $CV_v$, $v = 1, \ldots, 5$. For each $\lambda$ and $v$, we find a classifier using training set $T_{CV_v}$, and calculate the cross-validation criterion as:

$$ CV(\lambda) = \frac{1}{5} \sum_{v=1}^{5} \sum_{(x_i, y_i) \in CV_v} I(y_i \neq \hat{y}_i). \quad (4.7) $$

The tuning parameter $\lambda$ is searched over a grid: $\log_2(\lambda) = -14, -13, \ldots, 15$. When a tie occurs, the larger value of $\lambda$ would be chosen. The test data are used to examine the prediction accuracy of the final classifier in terms of classification accuracy and variable selection criteria as defines in last Chapter.

For each method, its average testing error, number of correct and incorrect zeros, model size, and number of times that the true model is correctly identified are summarized in tables. The numbers in the parentheses are standard errors. We also report the
theoretical performance from the Bayes Classifier on the bottom of each table to provide a reference in evaluation.

### 4.3.1 Four-Class Linear Example 1

Consider the same four-class example in Section 3.5.1. Input vector $x \in \mathbb{R}^{20}$, only the first two elements are relevant. For each class $k = 1, 2, 3, 4$, the first two components of $x$ are generated from $N(\mu_k, 2I_2)$, where $\mu_1 = (\sqrt{2}, \sqrt{2}), \mu_2 = (-\sqrt{2}, \sqrt{2}), \mu_3 = (-\sqrt{2}, -\sqrt{2}), \mu_4 = (\sqrt{2}, -\sqrt{2})$, and $I_2$ is a $2 \times 2$ identity matrix. The remaining eighteen components are i.i.d. generated from $N(0, 1)$. Here the sample size of training data is $n = 200$, and the sample size of testing data is $n' = 40,000$. Figure 4.1 is a plot of a randomly chosen training set. The solid lines are the Bayes boundaries.

<table>
<thead>
<tr>
<th>Method</th>
<th>Testing Error</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2 MSVM</td>
<td>0.40</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>L1 MSVM</td>
<td>0.42</td>
<td>18.99</td>
<td>0.16</td>
<td>17.41</td>
<td>2</td>
</tr>
<tr>
<td>Adapt-L1 MSVM</td>
<td>0.42</td>
<td>38.37</td>
<td>0.29</td>
<td>13.73</td>
<td>12</td>
</tr>
<tr>
<td>Supnorm MSVM</td>
<td>0.30</td>
<td>67.04</td>
<td>0</td>
<td>3.24</td>
<td>73</td>
</tr>
<tr>
<td>Adapt-supI MSVM</td>
<td>0.30</td>
<td>70.08</td>
<td>0</td>
<td>2.48</td>
<td>82</td>
</tr>
<tr>
<td>Adapt-supII MSVM</td>
<td>0.32</td>
<td>70.80</td>
<td>0</td>
<td>2.30</td>
<td>84</td>
</tr>
<tr>
<td>supSCAD MSVM</td>
<td>0.30</td>
<td>71.60</td>
<td>0</td>
<td>2.10</td>
<td>91</td>
</tr>
<tr>
<td>supSCAD MPSVM</td>
<td>0.30</td>
<td>65.00</td>
<td>0</td>
<td>3.75</td>
<td>69</td>
</tr>
<tr>
<td>Bayes</td>
<td>0.29</td>
<td>72</td>
<td>0</td>
<td>2</td>
<td>100</td>
</tr>
</tbody>
</table>

From table 4.1, we see that, similar to the logit model, supSCAD penalty is better than other penalties in MSVM framework. In term of classification accuracy, supnorm MSVM, Adapt-supI MSVM, supSCAD MSVM/MPSVM all achieve an average testing error of 0.30, just a slightly larger than the Bayes error. As of variable selection performance, sup-SCAD MSVM achieves the largest frequency of getting the correct model. In the total of 100 runs, sup-SCAD MSVM select the correct model 91 times and have an average model size of 2.1 which is just slightly greater than the oracle model size. Entries in the first column of table 4.1 have standard errors in the range of 0.001 to 0.003.
Figure 4.1: The Bayes boundary for the four-class example 1.
4.3.2 Four-Class Linear Example 2

Consider a similar four-class example with the last one. This time, the 20-dimensional input vector \( \mathbf{x} \) contains two important variables and another two important but weak variables. For each class \( k = 1, 2, 3, 4 \), the first two components of \( \mathbf{x} \) are generated from \( N(\mu_k, 3I_2) \), where \( \mu_1, \mu_2, \mu_3, \) and \( \mu_4 \) are the same as in the last example. And the next two components are generated from \( N(\nu_k, I_2) \), where \( \nu_1 = (0.5, 0.5)^T, \nu_2 = (-0.5, 0.5)^T, \nu_3 = (0.5, -0.5)^T, \nu_4 = (-0.5, -0.5)^T \). The remaining sixteen components are i.i.d. generated from \( N(0, 1) \). Training sample size and testing sample size are still \( n = 200 \), and \( n' = 40,000 \). In other words, the first 2 components of \( \mathbf{x} \) are strong signals to classification, and the second 2 components are weak signals, whereas the remaining 16 components are redundant noises.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2 MSVM</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0.37</td>
</tr>
<tr>
<td>L1 MSVM</td>
<td>10.05</td>
<td>0.08</td>
<td>19.05</td>
<td>0</td>
<td>0.37</td>
</tr>
<tr>
<td>Adapt-L1 MSVM</td>
<td>28.34</td>
<td>0.14</td>
<td>16.55</td>
<td>0</td>
<td>0.36</td>
</tr>
<tr>
<td>Supnorm MSVM</td>
<td>53.16</td>
<td>0.08</td>
<td>6.69</td>
<td>50</td>
<td>0.31</td>
</tr>
<tr>
<td>Adapt-supI MSVM</td>
<td>61.20</td>
<td>1.04</td>
<td>4.44</td>
<td>58</td>
<td>0.32</td>
</tr>
<tr>
<td>Adapt-supII MSVM</td>
<td>54.94</td>
<td>0.72</td>
<td>6.09</td>
<td>48</td>
<td>0.34</td>
</tr>
<tr>
<td>supSCAD MSVM</td>
<td>50.28</td>
<td>0.32</td>
<td>7.35</td>
<td>60</td>
<td>0.33</td>
</tr>
<tr>
<td>supSCAD MPSVM</td>
<td>53.10</td>
<td>0</td>
<td>6.73</td>
<td>45</td>
<td>0.30</td>
</tr>
<tr>
<td>Bayes rule</td>
<td>64</td>
<td>0</td>
<td>4</td>
<td>100</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Results shown in table 4.2 are not as good as in table 4.1 due to the increasing difficulty of this problem. From the nonzero values in the column of “Incorrect Zeros”, all the penalized methods, except for supSCAD MPSVM, miss the two weak signals sometimes. The comparisons among supnorm MSVM, Adaptive supnormI/II MSVM, supSCAD MSVM/MPSVM are obscure. On one hand, supSCAD MSVM identifies the true model 60 times but has a average model size of 7.35. On the other hand, supSCAD MPSVM does not miss any important variables even a single time, it only selects the true model 45 times. A more balanced result is achieved by Adaptive supnormI MSVM. It selects the true model 58 times with an average model size of 4.44.

In classification accuracy, supSCAD MPSVM gets the lowest error rate of 0.30. Entries in the last column of this table have standard errors in the range of 0.001 to 0.003.


### 4.3.3 Four-Class Linear Example 3

The same example as in section 3.5.4. In the last two examples, the important variables are all the same for all the classes. However in reality, there may be some cases where the important variables are different from class to class. Consider a 10-dimension input \( x \), the first four elements \( x_1, x_2, x_3, x_4 \) are generated i.i.d from \( \text{Unif}[-1,1] \) and the last six elements \( x_5, \ldots, x_{10} \) are generated i.i.d from \( \mathcal{N}(0,8^2) \). Define the functions

\[
\begin{align*}
  f_1 &= -5x_1 + 5x_4, \\
  f_2 &= 5x_1 + 5x_2, \\
  f_3 &= -5x_2 + 5x_3, \\
  f_4 &= -5x_3 - 5x_4,
\end{align*}
\]

and set \( p_k(x) = P(Y = k|X = x) \propto \exp(f_k(x)), k = 1, 2, 3, 4 \). In this example, we have \( n = 200 \) and \( n' = 40,000 \). Although the first 4 variables are important variables, \( x_1 \) is not important for distinguishing class 3 and class 4, \( x_2 \) is noninformative for distinguishing class 1 and class 4, \( x_3 \) is noninformative for distinguishing class 1 and class 2, and \( x_4 \) is noninformative for distinguishing class 2 and class 3.

Table 4.3: Simulation results for the four-class example 3

<table>
<thead>
<tr>
<th>Method</th>
<th>Testing Error</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2 MSVM</td>
<td>0.34</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>L1 MSVM</td>
<td>0.34</td>
<td>9</td>
<td>0.28</td>
<td>8.90</td>
<td>9</td>
</tr>
<tr>
<td>Adapt-L1 MSVM</td>
<td>0.33</td>
<td>18.02</td>
<td>0.30</td>
<td>7.30</td>
<td>16</td>
</tr>
<tr>
<td>Supnorm MSVM</td>
<td>0.33</td>
<td>3.65</td>
<td>0.12</td>
<td>9.24</td>
<td>3</td>
</tr>
<tr>
<td>Adapt-sup MSVMI</td>
<td>0.34</td>
<td>10.08</td>
<td>0.22</td>
<td>7.64</td>
<td>15</td>
</tr>
<tr>
<td>Adapt-sup MSVMII</td>
<td>0.33</td>
<td>10.08</td>
<td>0.08</td>
<td>7.55</td>
<td>11</td>
</tr>
<tr>
<td>supSCAD MSVM</td>
<td>0.34</td>
<td>10.32</td>
<td>0.26</td>
<td>7.61</td>
<td>7</td>
</tr>
<tr>
<td>supSCAD MPSVM</td>
<td>0.21</td>
<td>10.85</td>
<td>0</td>
<td>7.29</td>
<td>11</td>
</tr>
<tr>
<td>Bayes</td>
<td>0.14</td>
<td>32</td>
<td>0</td>
<td>4</td>
<td>100</td>
</tr>
</tbody>
</table>

Our proposed supSCAD penalty treats all coefficients of one variable corresponding to different classes as a group and removes the variable if it is non-informative across all classes. By design of this example, important variables have zero coefficients for certain classes. As a result, \( L_1 \) penalty is more appropriate and should give better result.

From Table 4.3 we see all the methods don’t perform well compared to the penal-
ized logistic regression methods in last chapter. Adapt-L1 MSVM dominates the other MSVMs in identifying the true model, while supSCAD MPSVM gets the smallest average model size and testing error. And entries in the first column of table 4.3 have standard errors in the range of 0.003 to 0.008.

4.3.4 Four-Class Linear Example 4

Here we consider a similar example to the last one, where the important variables are different from class to class. But this time, we have a 20-dimension input $\mathbf{x}$, the first four elements $x_1, x_2, x_3, x_4$ are generated uniformly from a 4-dimension ball with radius of 3, i.e. $\{(x_1, x_2, x_3, x_4)^T \in \mathbb{R}^4 : \sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2} \leq 3\}$. The remaining $x_5, \ldots, x_{20}$ are generated i.i.d from $\mathcal{N}(0, 1)$. Define the functions

$$f_1 = -3x_1 + 3x_4,$$
$$f_2 = 3x_1 + 3x_2,$$
$$f_3 = -3x_2 + 3x_3,$$
$$f_4 = -3x_3 - 3x_4,$$

and set $p_k(x) = P(Y = k|X = x) \propto exp(f_k(x)), k = 1, 2, 3, 4$. We have $n = 200$ and $n' = 40,000$. The first 4 variables are important variables but not to all the classes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2 MSVM</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0.36</td>
</tr>
<tr>
<td>L1 MSVM</td>
<td>22.82</td>
<td>0.16</td>
<td>17.12</td>
<td>4</td>
<td>0.32</td>
</tr>
<tr>
<td>Adapt-L1 MSVM</td>
<td>47.75</td>
<td>0.2</td>
<td>11.74</td>
<td>13</td>
<td>0.30</td>
</tr>
<tr>
<td>Supnorm MSVM</td>
<td>10.86</td>
<td>0.08</td>
<td>17.49</td>
<td>0</td>
<td>0.31</td>
</tr>
<tr>
<td>Adapt-supI MSVM</td>
<td>26.6</td>
<td>0.16</td>
<td>13.57</td>
<td>10</td>
<td>0.32</td>
</tr>
<tr>
<td>Adapt-supII MSVM</td>
<td>28.62</td>
<td>0.06</td>
<td>12.88</td>
<td>8</td>
<td>0.31</td>
</tr>
<tr>
<td>supSCAD MSVM</td>
<td>27.17</td>
<td>0.2</td>
<td>13.43</td>
<td>4</td>
<td>0.33</td>
</tr>
<tr>
<td>supSCAD MPSVM</td>
<td>41.49</td>
<td>0</td>
<td>9.63</td>
<td>2</td>
<td>0.19</td>
</tr>
<tr>
<td>Bayes rule</td>
<td>72</td>
<td>0</td>
<td>4</td>
<td>100</td>
<td>0.11</td>
</tr>
</tbody>
</table>

As in the last example, Adapt-L1 MSVM still gives the most satisfactory results. It achieves best results in model selection and class prediction. However, supSCAD
MPSVM obtains a significant low misclassification error. Entries in the last column of table 4.4 have standard error in the range of 0.004 to 0.008.

### 4.3.5 Three-Class Nonlinear Example

Consider the same example as in section 3.5.5. In this section, we consider a 3-class example which has nonlinear decision boundaries in its input variables. First generate $x_1 \sim Unif[-3, 3]$ and $x_2 \sim Unif[-6, 6]$. Define the functions

$$
\begin{align*}
    f_1 &= -2x_1 + 0.2x_1^2 - 0.1x_2^2 + 0.2, \\
    f_2 &= -0.4x_1^2 + 0.2x_2^2 - 0.4, \\
    f_3 &= 2x_1 + 0.2x_1^2 - 0.1x_2^2 + 0.2,
\end{align*}
$$

and set $p_k(x) = P(Y = k|X = x) \sim \exp(f_k(x)), k = 1, 2, 3$ and $\sigma = 2, n = 200, n' = 40,000$. Then generate three noise variables $x_i \sim N(0, \sigma^2), i = 3, 4, 5, 6, 7$. To achieve nonlinear classification, we use basis expansion. We fit the nonlinear MSVM and MPSVM by including the seven main effects, their square terms, and their cross products as the basis functions. Figure 4.2 is a plot of a randomly chosen training set. The solid lines are the Bayes boundaries.

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct Zeros</th>
<th>Incorrect Zeros</th>
<th>Model Size</th>
<th>Correct Model</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2 MSVM</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>0.19</td>
</tr>
<tr>
<td>L1 MSVM</td>
<td>51.42</td>
<td>1.16</td>
<td>21.02</td>
<td>1</td>
<td>0.16</td>
</tr>
<tr>
<td>Adapt-L1 MSVM</td>
<td>67.71</td>
<td>1.39</td>
<td>15.31</td>
<td>8</td>
<td>0.15</td>
</tr>
<tr>
<td>Supnorm MSVM</td>
<td>51.42</td>
<td>1.15</td>
<td>21.02</td>
<td>1</td>
<td>0.16</td>
</tr>
<tr>
<td>Adapt-supI MSVM</td>
<td>71.06</td>
<td>1.51</td>
<td>12.60</td>
<td>10</td>
<td>0.15</td>
</tr>
<tr>
<td>Adapt-supII MSVM</td>
<td>56.66</td>
<td>1.35</td>
<td>15.68</td>
<td>9</td>
<td>0.15</td>
</tr>
<tr>
<td>supSCAD MSVM</td>
<td>91.03</td>
<td>1.96</td>
<td>4.22</td>
<td>67</td>
<td>0.14</td>
</tr>
<tr>
<td>supSCAD MPSVM</td>
<td>72.82</td>
<td>1.48</td>
<td>12.69</td>
<td>15</td>
<td>0.14</td>
</tr>
<tr>
<td>Bayes rule</td>
<td>96</td>
<td>0</td>
<td>3</td>
<td>100</td>
<td>0.12</td>
</tr>
</tbody>
</table>

From Table 4.5, supSCAD MSVM achieves the least classification error and highest model selection accuracy. And supSCAD MPSVM has the same smallest test error of 0.14 but a lower frequency of selecting correct model as of 15 out of 100. Entries in the last column of “Testing Error” have standard errors in the range of 0.001 to 0.002.
Figure 4.2: The Bayes boundary for the nonlinear three-class example.
4.4 Real Examples

We use the same real data as in the last chapter, the Leukemia data and the Small Round Blue Cell Tumors data.

4.4.1 Leukemia Study

Leukemia data from Guyon et al. (2002) consist of 7129 genes and 72 patient bone marrow samples, which can be categorized into three classes. The three classes are 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 downloaded from http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi. The whole data is divided into a training set consisting of 38 samples, and a test set consisting of 34 samples. Table 4.6 summarizes the class distribution in each set.

After using the same preprocessing, standardizing and ranking steps as in last chapter, then eight different MSVMs are applied to the data. Leave-one-out cross validation is used for tuning parameter selection. All the results are summarized in table 4.7.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ALL B</th>
<th>ALL T</th>
<th>AML</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>19</td>
<td>8</td>
<td>11</td>
<td>38</td>
</tr>
<tr>
<td>Test</td>
<td>19</td>
<td>1</td>
<td>14</td>
<td>34</td>
</tr>
</tbody>
</table>

All the methods, except L2 MSVM, select variables only from top 100 genes. As indicated in Zhang et al. (2008), the L1 MSVM and supnorm MSVM are essentially the same in three-class problems. In this case, these two procedures achieve the least testing error of 0. While supSCAD MSVM achieves the least model size of 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>L2 MSVM</th>
<th>L1 MSVM</th>
<th>Adapt-L1 MSVM</th>
<th>supnorm MSVM</th>
<th>Adapt-sup MSVM</th>
<th>Adapt-sup MSVM</th>
<th>supSCAD MSVM</th>
<th>supSCAD MPSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Size</td>
<td>200</td>
<td>16</td>
<td>12</td>
<td>16</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Test Error</td>
<td>2/34</td>
<td>0/34</td>
<td>3/34</td>
<td>0/34</td>
<td>2/34</td>
<td>2/34</td>
<td>4/34</td>
<td>2/34</td>
</tr>
</tbody>
</table>
4.4.2 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 with 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 four tumor class distributions in training and test set are given in table 4.8.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NB</th>
<th>RMS</th>
<th>BL</th>
<th>EWS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>12</td>
<td>20</td>
<td>8</td>
<td>23</td>
<td>63</td>
</tr>
<tr>
<td>Test</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>20</td>
</tr>
</tbody>
</table>

Before analyzing this example, we standardize the data set and rank all the genes in the same way we did in the last Leukemia study. The only difference here is that we do not need the preprocessing steps as in Leukemia study. After the same data standardization and gene ranking as in last chapter, eight penalized MSVM/MPSVM approaches were applied to the training data of 200 genes and 63 training samples. The 200 genes include the top significant 100 genes and bottom non-significant 200 genes. Leave-one-out cross validation is used to select the appropriate tuning parameter. The learned models were used to predict the unseen 20 test samples. Results are compared in table 4.9.

<table>
<thead>
<tr>
<th>Method</th>
<th>L2 MSVM</th>
<th>L1 MSVM</th>
<th>Adapt-L1 MSVM</th>
<th>supnorm MSVM</th>
<th>Adapt-sup MSVMI</th>
<th>Adapt-sup MSVMII</th>
<th>supSCAD MSVM</th>
<th>supSCAD MPSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Size</td>
<td>200</td>
<td>51</td>
<td>44</td>
<td>53</td>
<td>38</td>
<td>47</td>
<td>8</td>
<td>39</td>
</tr>
<tr>
<td>Test Error</td>
<td>0/20</td>
<td>1/20</td>
<td>0/20</td>
<td>0/20</td>
<td>0/20</td>
<td>1/20</td>
<td>4/20</td>
<td>1/20</td>
</tr>
</tbody>
</table>
Chapter 5

Conclusion and Discussion

In this dissertation, a discussion about a new regularization method is presented in multi-class problems. This new proposed penalty, the supSCAD penalty, enhances sparse learning in multi-classification by retaining the merits from both SCAD and supnorm penalties. It incorporates the natural group effects of the coefficients associated with the same covariate to deliver more parsimonious classifiers with desired oracle properties.

To tackle the numerical challenge due to the inherited non-differentiability and non-convexity, two methods of iteratively linear approximation, DCA and LLA, are proposed. For multiclass probability estimation, supSCAD is applied to multinomial logistic regression, which achieves variable selection and conditional probability estimation simultaneously. An optimization procedure involving quadratic approximation to the multinomial log-likelihood function and nested DCA/LLA for the supSCAD penalty is developed and evaluated by numeric experiments. Extending to multi-category SVM framework, supSCAD MSVM/MPSVM demonstrated competitive performance compared to other regularized MSVM using simulated and real data.

For the supSCAD logistic regression, a good initial point is essential to learn the final classifier. We suggest using the MLE estimates when possible. Otherwise, origin point is always available. In this case, a two-step procedure, supSCAD logistic regression followed by logistic regression, works very well and improve the classification and conditional probability estimation of the corresponding one-step procedure.

The underlying assumption of supSCAD penalty is that the coefficients across different classes can be naturally grouped by their associated covariates. We notice that when this assumption is not meet, the outstanding performance of supSCAD classifier
could be badly undermined, especially for supSCAD MSVM. While in some problems, it is possible to form groups among covariates and varying from class to class, as discussed in Yuan and Lin (2006) and Zou and Yuan (2006). A future research direction is to form flexible groups and construct a more robust multi-classification method.
REFERENCES


Appendix A

Proofs

Before we present the proofs of the theorems, we first state some regularity conditions. The reparameterized multinomial log-likelihood \( \tilde{L} \) and its associated true parameter vector \( \beta^* \) are defined in (3.10) and (3.11).

A.1 Regularity Conditions

1. The observations \((x_i, y_i), i = 1, \ldots, n,\) are i.i.d. with multinomial distribution \((\pi_1, \ldots, \pi_K), 1 > \pi_k > 0, \sum_{k=1}^{K} \pi_k = 1.\)

2. The Fisher information matrix \( I(\beta) = E\{(\frac{\partial \tilde{L}}{\partial \beta})(\frac{\partial \tilde{L}}{\partial \beta})^T\} \) is finite and positive definite at \( \beta = \beta^*. \)

3. There is a sufficient large enough open set \( w \) that contains \( \beta^* \) such that for all \( \beta \in w \)

\[ \left| \frac{\partial^3 \tilde{L}}{\partial \beta_{k_j} \partial \beta_{k_{j1}} \partial \beta_{k_{j2}}} \right| \leq M(x) < \infty \]

and

\[ E_{\beta^*}[M(x)] < \infty. \]
A.2 Proof of Theorem 1

To prove Theorem 1, it is enough to show that for any given \( \varepsilon > 0 \), there exists a large enough constant \( C \) such that

\[
P\{ \inf_{\|u\| = C} \tilde{R}(\beta^* + u/\sqrt{n}) > \tilde{R}(\beta^*) \} \geq 1 - \varepsilon, \tag{A.1}
\]

which implies that with probability at least \( 1 - \varepsilon \) there exists a local minimum in the ball \( \{ \beta^* + u/\sqrt{n} : \|u\| \leq C \} \). This in turn implies that there exists a local minimizer such that \( \|\hat{\beta} - \beta^*\| = O_p(n^{-1/2}) \), which is exactly what we want to show.

Notice that

\[
\tilde{R}(\beta^* + \frac{u}{\sqrt{n}}) - \tilde{R}(\beta^*) = -(\tilde{\mathcal{L}}(\beta^* + \frac{u}{\sqrt{n}}) - \tilde{\mathcal{L}}(\beta^*))
\]

\[
+ n \sum_{j=1}^{d} \left[ J_{\lambda_n}(\max\{|\beta^*_1 + \frac{u_{1j}}{\sqrt{n}}|, |\beta^*_2 + \frac{u_{2j}}{\sqrt{n}}|, \ldots, |\beta^*_{(K-1),j} + \frac{u_{(K-1),j}}{\sqrt{n}}|, |\sum_{l=1}^{K-1} (\beta^*_{lj} + \frac{u_{lj}}{\sqrt{n}})|\}ight]
\]

\[
- J_{\lambda_n}(\max\{|\beta^*_1|, |\beta^*_2|, \ldots, |\beta^*_{(K-1),j}|, |\sum_{l=1}^{K-1} \beta^*_{lj}|\})
\]

\[
\geq -(\tilde{\mathcal{L}}(\beta^*) + \frac{u}{\sqrt{n}}) - \tilde{\mathcal{L}}(\beta^*)
\]

\[
+ n \sum_{j=1}^{d} \left[ J_{\lambda_n}(\max\{|\beta^*_1 + \frac{u_{1j}}{\sqrt{n}}|, |\beta^*_2 + \frac{u_{2j}}{\sqrt{n}}|, \ldots, |\beta^*_{(K-1),j} + \frac{u_{(K-1),j}}{\sqrt{n}}|, |\sum_{l=1}^{K-1} (\beta^*_{lj} + \frac{u_{lj}}{\sqrt{n}})|\}ight]
\]

\[
- J_{\lambda_n}(\max\{|\beta^*_1|, |\beta^*_2|, \ldots, |\beta^*_{(K-1),j}|, |\sum_{l=1}^{K-1} \beta^*_{lj}|\})
\]

\[
= -\tilde{\mathcal{L}}'(\beta^*)^T \frac{u}{\sqrt{n}} + \frac{1}{2} u^T I(\beta^*) u \{1 + o_p(1)\}
\]

\[
+ n \sum_{j=1}^{d} \left[ J_{\lambda_n}(\max\{|\beta^*_1 + \frac{u_{1j}}{\sqrt{n}}|, |\beta^*_2 + \frac{u_{2j}}{\sqrt{n}}|, \ldots, |\beta^*_{(K-1),j} + \frac{u_{(K-1),j}}{\sqrt{n}}|, |\sum_{l=1}^{K-1} (\beta^*_{lj} + \frac{u_{lj}}{\sqrt{n}})|\}ight]
\]

\[
- J_{\lambda_n}(\max\{|\beta^*_1|, |\beta^*_2|, \ldots, |\beta^*_{(K-1),j}|, |\sum_{l=1}^{K-1} \beta^*_{lj}|\})
\]

\[
= D_1 + D_2 + D_3
\]

Note that \( \tilde{\mathcal{L}}'(\beta^*)^T / \sqrt{n} = O_p(1) \), thus \( D_2 \) is asymptotic positive and dominates \( D_1 \) by
choosing a sufficiently large $C$. And for large $n$, when $\lambda_n \to 0$,

$$D_3 = \sum_{j=1}^{s} |J_{\lambda_n}(\max\{|\beta_{1j}^*|, |\beta_{2j}^*|, \ldots, |\beta_{(K-1)j}^*|, |\sum_{m=1}^{K-1} \beta_{mj}^*|\})|$$

$$- J_{\lambda_n}(\max\{|\beta_{1j}^*|, |\beta_{2j}^*|, \ldots, |\beta_{(K-1)j}^*|, |\sum_{l=1}^{K-1} \beta_{lj}^*|\}) = 0,$$

since the supSCAD penalty is flat for coefficient of magnitude larger than $a\lambda_n$.

Based on the above, $\tilde{R}(\beta^* + u/\sqrt{n}) - \tilde{R}(\beta^*)$ is dominated by $D_2$. Hence, by choosing a sufficient large $C$ (A.1) holds.

### A.3 Proof of Lemma 1

As long as the max $\{\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\}$ is zero, then each component in $\{|\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\}$ is zero.

It is sufficient to show that with probability tending to 1 as $n \to \infty$, for any $\beta_+$ satisfying $||\beta_+ - \beta^*|| = O_p(n^{-1/2})$ and any constant $C$, for $j = s+1, \ldots, d$,

$$\frac{\partial \tilde{R}(\beta)}{\partial \beta_{kj}^*} > 0, \text{ for } 0 \leq \beta_{kj} \leq Cn^{-1/2} \text{ and } \beta_{kj} = \max\{|\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\}$$

$$\frac{\partial \tilde{R}(\beta)}{\partial \beta_{kj}^*} < 0, \text{ for } -Cn^{-1/2} \leq \beta_{kj} \leq 0 \text{ and } \beta_{kj} = -\max\{|\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\}$$

where $\frac{\partial \tilde{R}}{\partial \beta_{kj}^*}$ and $\frac{\partial \tilde{R}}{\partial \beta_{kj}^*}$ denote the right and left hand partial derivative respectively.

$$\frac{\partial \tilde{R}}{\partial \beta_{kj}} = -\frac{\partial \tilde{\mathcal{L}}}{\partial \beta_{kj}} + nJ_{\lambda_n}(|\beta_{kj}|)sgn(\beta_{kj}) \text{ when } |\beta_{kj}| = \max\{|\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\}$$

By Taylor expansion,

$$\frac{\partial \tilde{\mathcal{L}}(\beta)}{\beta_{kj}} = \frac{\partial \tilde{\mathcal{L}}(\beta^*)}{\beta_{kj}} + \sum_{j_1=1}^{d} \sum_{k_1=1}^{K-1} \frac{\partial^2 \tilde{\mathcal{L}}(\beta^*)}{\beta_{kj} \beta_{k_{1j_1}}} (\beta_{k_{1j_1}} - \beta_{k_{1j_1}}^*)$$

$$+ \sum_{j_1=1}^{d} \sum_{k_1=1}^{K-1} \sum_{j_2=1}^{d} \sum_{k_2=1}^{K-1} \frac{\partial^3 \tilde{\mathcal{L}}(\beta^*)}{\beta_{kj} \beta_{k_{1j_1}} \beta_{k_{2j_2}}} (\beta_{k_{1j_1}} - \beta_{k_{1j_1}}^*) (\beta_{k_{2j_2}} - \beta_{k_{2j_2}}^*)$$

75
where $\beta^*$ lies between $\beta$ and $\beta^*$. Note that $n^{-1}\partial^2 \tilde{a}(\beta^*) = O_p(n^{-1/2})$ and $\frac{1}{n}\partial^2 \tilde{a}(\beta^*) = E\{\partial^2 \tilde{a}(\beta^*)\} + o_p(1)$. We have

$$\frac{\partial \tilde{R}(\beta)}{\partial \beta_{kj}} = n\lambda_n \{O_p(n^{-1/2}/\lambda_n) + \frac{J_n'(|\beta_{kj}|)}{\lambda_n}\} \text{ if } \beta_{kj} > 0$$

$$\frac{\partial \tilde{R}(\beta)}{\partial \beta_{kj}} = n\lambda_n \{O_p(n^{-1/2}/\lambda_n) - \frac{J_n'(|\beta_{kj}|)}{\lambda_n}\} \text{ if } \beta_{kj} < 0$$

In both cases, the second term dominates the first term.

If $|\kappa_j| = |\sum_{m=1}^{K-1} \beta_{mj}| = \max\{||\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\}$, then

$$\frac{\partial \tilde{R}}{\partial \kappa_j} = -\frac{\partial \tilde{L}}{\partial \beta_{kj}} \frac{\partial \beta_{kj}}{\partial \kappa_j} + nJ_n'(|\kappa_j|)\text{sgn}(\kappa_j) = -\frac{\partial \tilde{L}}{\partial \beta_{kj}} + nJ_n'(|\kappa_j|)\text{sgn}(\kappa_j).$$

Therefore,

$$\frac{\partial \tilde{R}(\beta)}{\partial \kappa_j} = n\lambda_n \{O_p(n^{-1/2}/\lambda_n) + \frac{J_n'(|\kappa_j|)}{\lambda_n}\} \text{ if } \kappa_j > 0$$

$$\frac{\partial \tilde{R}(\beta)}{\partial \kappa_j} = n\lambda_n \{O_p(n^{-1/2}/\lambda_n) - \frac{J_n'(|\kappa_j|)}{\lambda_n}\} \text{ if } \kappa_j < 0$$

The second term still dominates the first term. Thus the result of Lemma 1 follows.

### A.4 Proof of Theorem 2

Proof: Part 1 holds by lemma 1. For Part 2

$$\lambda_n \leq a^{-1} \max_{1 \leq j \leq s} \max\{||\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\} \leq \lambda_n$$

$$\Rightarrow \max_{1 \leq j \leq s} \max\{||\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\} \geq a\lambda_n$$

$$\Rightarrow J_n(\max\{||\beta_{1j}|, |\beta_{2j}|, \ldots, |\beta_{K-1,j}|, |\sum_{m=1}^{K-1} \beta_{mj}|\}) = J_n(a\lambda_n) = \frac{(a + 1)^2\lambda_n^2}{2}.$$