THE SOLUTION PATHS OF MULTICATEGORY SUPPORT VECTOR MACHINES: ALGORITHM AND APPLICATIONS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for 
the Degree Doctor of Philosophy in the 
Graduate School of The Ohio State University

By

Zhenhuan Cui, M.S.

* * * * *

The Ohio State University

2007

Dissertation Committee:
Professor Yoonkyung Lee, Adviser
Professor Prem Goel
Professor Radu Herbei

Approved by

Adviser
Graduate Program in Statistics
ABSTRACT

The solution path of a regularization method means the entire set of solutions indexed by each value of the regularization parameter that controls the complexity of a fitted model. An algorithm for fitting the entire regularization path of the support vector machine (SVM) was recently proposed by Hastie et al. (2004). It allows effective computation of solutions and greatly facilitates the choice of the regularization parameter that balances a trade-off between complexity of a solution and its fit to data. Extending the idea to more general setting of the multiclass case, we characterize the coefficient path of the multiclass SVM via the complementarity conditions for optimality. The extended algorithm provides a computational shortcut to attain the entire spectrum of solutions from the most regularized to the completely overfitted ones.

In practice, large data sets and the choice of a flexible kernel may pose a computational challenge to the sequential updating algorithm. We extend the solution path algorithm to incorporate different data weights and apply it to a compressed data set with weights by subset sampling to alleviate the computational load for large data sets. A few approaches for approximate solution paths are proposed. In addition, some related computational issues are discussed and the effectiveness of the algorithm is demonstrated for some benchmark data sets.
ACKNOWLEDGMENTS

I owe a great amount of appreciation to Dr. Lee, my advisor throughout all these years of my PhD study. Words are too pale to show my gratitude to her consistent help, powerful support, inspiring encouragement and never failing patience. I am also grateful for the valuable suggestions and opinions from the committee members, including Professors Prem Goel, Joseph Verducci, Steven MacEachern and Radu Herbei, who have devoted their precious time to assist me in the completion of my degree.
VITA

October 30, 1973 ......................... Born - Beijing, China

1996 ........................................ B.S. Electrical Engineering, Beijing Information Technology Institute

2004 ........................................... M.S. Statistics, The Ohio State University

1996-2000 ................................. Central View Survey and Consulting Center, Beijing, China

2002 - 2007 ................................. Graduate Teaching or Research Associate, The Ohio State University

2007 ........................................... Summer Internship, SAS Institute, Cary, NC

PUBLICATIONS

Research Publications


FIELDS OF STUDY

Major Field: Statistics
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapters:</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iii</td>
</tr>
<tr>
<td>Vita</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>Chapters:</td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Multicategory SVM and Optimality Conditions</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Standard Case</td>
<td>5</td>
</tr>
<tr>
<td>2.2 General Case</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Optimality Conditions</td>
<td>8</td>
</tr>
<tr>
<td>3. Solution Path in the Standard Case</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Initialization</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Characterizing Coefficient Paths</td>
<td>15</td>
</tr>
<tr>
<td>3.3 Computation</td>
<td>20</td>
</tr>
<tr>
<td>3.4 An Illustrative Example</td>
<td>22</td>
</tr>
<tr>
<td>4. Solution Path for Weighted Data</td>
<td>28</td>
</tr>
<tr>
<td>4.1 Modified Initialization</td>
<td>28</td>
</tr>
<tr>
<td>4.2 Modified Path Finding Algorithm</td>
<td>30</td>
</tr>
</tbody>
</table>
5. Coping with Large Data Sets ........................................ 32
   5.1 Motivation ......................................................... 32
   5.2 Subset Sampling Schemes ........................................ 33
   5.3 Reformulation of Regularization Problems ...................... 34

6. Practical Issues in Implementation ................................. 37
   6.1 Stopping Rule .................................................... 37
   6.2 Choice of Kernel and Its Parameter ............................ 39

7. Numerical Studies .................................................. 43
   7.1 Simulation Study ................................................ 43
      7.1.1 Comparison of Subset Sampling Schemes ................. 43
      7.1.2 Bayes Error, Dimensionality and Data Reduction Factor . 45
   7.2 Benchmark Study ............................................... 47

8. Discussion .......................................................... 53

Appendices:

A. R Package, msvmpath manual ...................................... 56

Bibliography ............................................................ 66
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Optimal $\gamma$ values determined by the criteria of $BSS$ and $D$ and grid search</td>
<td>41</td>
</tr>
<tr>
<td>6.2 Test error rates or cross validated error rates at the optimal $\gamma$ chosen by grid search and $BSS$ and the efficiency of the $BSS$ criterion relative to grid search</td>
<td>42</td>
</tr>
<tr>
<td>7.1 Test error rates of the approximate SVM solutions with different data reduction schemes.</td>
<td>45</td>
</tr>
<tr>
<td>7.2 Statistics of solution paths, average elbow set sizes, path lengths and computing time.</td>
<td>45</td>
</tr>
<tr>
<td>7.3 Description of data sets</td>
<td>50</td>
</tr>
<tr>
<td>7.4 Error rates of the MSVM assisted with the solution path algorithm</td>
<td>50</td>
</tr>
<tr>
<td>7.5 Test error rates of the MSVM with different fractions of the training data for two subproblems of <code>zipcode</code> data</td>
<td>51</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>MSVM component loss $(f^j - y^j)_+$ where $y^j = -1/(k - 1)$.</td>
</tr>
<tr>
<td>3.1</td>
<td>Left: the boundaries of the Bayes classification rule in a scatter plot of the training data. Right: the classification boundaries determined by the MSVM at an appropriately chosen $\lambda$; Class 1: green, Class 2: magenta, and Class 3: blue.</td>
</tr>
<tr>
<td>3.2</td>
<td>Test error rate as a function of $\lambda$.</td>
</tr>
<tr>
<td>3.3</td>
<td>An estimated density function of the optimal parameters $\hat{\lambda}$ chosen by 100 tuning sets of sample size 300.</td>
</tr>
<tr>
<td>3.4</td>
<td>The entire paths of $\hat{f}^1_\lambda(x_i)$, $\hat{f}^2_\lambda(x_i)$, and $\hat{f}^3_\lambda(x_i)$ for an outlying instance $x_i$ from Class 3. The circles correspond to $\lambda$ with the minimum test error rate.</td>
</tr>
<tr>
<td>3.5</td>
<td>The size of elbow set $\mathcal{E}^j_\ell$ for three classes as a function of $\lambda$.</td>
</tr>
<tr>
<td>5.1</td>
<td>(a) Voronoi Tessellation of the centers of 40 clusters formed by $k$-means algorithm and (b) statistically equivalent blocks (40 = $10 \times 4$) obtained by using principal components as the ordering functions.</td>
</tr>
<tr>
<td>6.1</td>
<td>Heuristic criteria for selection of $\gamma$ in the radial kernel evaluated over the three-class simulated example; (a) $BSS$ and (b) $D$.</td>
</tr>
<tr>
<td>7.1</td>
<td>Test error rates as a function of the data fraction $(1/RF)$ for the two-class simulated example under three scenarios of (a) low (b) intermediate and (c) high Bayes error rate. The solid lines indicate the mean of ten replicates, and the dotted lines indicate the standard deviation around the mean.</td>
</tr>
</tbody>
</table>
7.2 Statistics of solution paths as a function of the data fraction (1/RF): (a) average elbow size and (b) path length for the intermediate Bayes error case. ................................................. 49

7.3 The hierarchical cluster analysis of the median vectors of the ten digits in zipcode. NOTE: Cluster 10 corresponds to the digit 0. ........... 52
CHAPTER 1

INTRODUCTION

Regularization methods are widely used in statistics and machine learning for data analysis. In general, the solution of a regularization method is defined as the minimizer \( f \) of an objective function of the form \( R_n(f) + \lambda J(f) \). The first term \( R_n(f) \) denotes the risk of \( f \) over the training data of size \( n \) with respect to a loss function and the second term includes the complexity of the function \( f \) measured by \( J(f) \) and the regularization parameter \( \lambda \). A few examples of regularization methods are smoothing splines (Wahba 1990), penalized logistic regression, and support vector machines (Vapnik 1998). The effectiveness of a regularization method often largely depends on the choice of the regularization parameter (or tuning parameter) which controls model elaboration in a continuous fashion. The LAR (least angle regression) of Efron et al. (2004) and the SVM path of Hastie et al. (2004) showcase recent developments of computational algorithms to characterize the entire regularization path in place of a user-dependent grid search for a good solution. These constructive algorithms not only enable efficient computation of solutions along the path, but also provide a bird’s-eye view of the spectrum of solutions from the least to the most complex fit to given data.
As illustrated in Hastie et al. (2004), capability of solving a system of linear equations is sufficient to find the complete solution path of the binary SVM as a function of its tuning parameter. In other words, the whole range of SVM solutions can be obtained without resorting to an external quadratic programming solver, except for one-time initialization if the two classes are unbalanced.

Different from this approach, there are quite a few widely used algorithms for the SVM such as SMO (Sequential Minimal Optimization) (Platt 1999), SVM light (Joachims 1999), and LIBSVM (Hsu and Lin 2002). However, these are only tailored for scalable computation of the SVM solution at a single value of the tuning parameter, proper specification of which would require a non-trivial inspection. Moreover, Hastie et al. (2004) empirically demonstrated that the computational cost of obtaining the entire regularization path could be almost the same as getting a single solution by other methods, while misspecification of the tuning parameter can be readily avoided through the SVM path.

Motivated by the idea of sequentially finding the SVM path for the binary case, we extend the algorithm to the multiclass case for general treatment of classification. This extension is for the Multicategory SVM (MSVM) in Lee et al. (2004b) that subsumes the binary SVM as a special case and retains the same problem structure. The Karush-Kuhn-Tucker optimality conditions (Mangasarian 1994) for the corresponding optimization problems play an important role in fully determining solution paths as a function of the regularization parameter $\lambda$. Hastie et al. (2004) cleverly utilized the conditions to show that the SVM coefficient path is piecewise linear in $1/\lambda$. In this thesis, we draw a parallel to this idea and necessary derivations for the multiclass case. It is established that the MSVM coefficient path is also piecewise
linear in $1/\lambda$ with an additional number of joints roughly proportional to the number of classes. The joints of the piecewise linear solution path are identified as the values of $\lambda$ at which any of data points on the margin of MSVM coordinates changes. The entire coefficient path is then constructed sequentially, and it provides a computational shortcut to simultaneous fitting and tuning. The extended algorithm of finding the MSVM coefficient path seamlessly encompasses that for the binary SVM, contributing further to our general understanding of the structure of the SVM formulation. When developing the analogous algorithm for the multiclass case, we closely follow the notation and terminology used in Hastie et al. (2004) for the binary case, in order to illuminate the connection between them.

Large data sets pose computational challenges to statistical learning methods in general. The path finding algorithm of the SVM is no exception in this matter though it requires solving a sequence of a system of linear equations only. As with other kernel methods, choice of a flexible nonlinear kernel for the SVM can significantly lower the computational efficiency that the algorithm enjoys in the linear setting. In dealing with massive data, one may seek a fast and inexpensive numerical approximation to the solution in lieu of exact computation. Keeping both computational and conceptual advantages of the entire solution path, we examine a few approaches to finding an approximate solution path of the SVM in this thesis. They rely mainly on subsetting of data-dependent basis functions from a function approximation point of view by reducing the data to a representative subset with appropriately defined weights. In fact, this idea is widely applicable to fitting models by various methods which can incorporate different weights. More generally, related data-analytic strategy for approximation has been commonly employed in statistics, machine learning and signal
processing for modeling and mining of large data sets and lossy data compression. A few examples are data squashing (DuMouchel et al. 2000), basis thinning (Xiang 1996), low-rank approximation to large Gram matrix (Williams and Seeger 2001) and vector quantization.

In Chapter 2, the framework of multicategory support vector machines is reviewed. The solution path for the standard case is developed in Chapter 3, while the solution path for weighted data is discussed in Chapter 4. Chapter 5 explores some approaches to dealing with large data sets. Practical issues in implementation are addressed in Chapter 6. Chapter 7 presents the results of numerical studies. And Chapter 8 discusses the directions in future research. The appendix includes the help manual for the R package msvmpath (Cui 2007).
CHAPTER 2

MULTICATEGORY SVM AND OPTIMALITY CONDITIONS

In the classification problem, we are given a training data set of \( n \) pairs of covariates and a known class label \((x_i, y_i)\) for \( i = 1, \ldots, n \). \( x_i \in \mathcal{X} \subset \mathbb{R}^p \) represents covariates and \( y_i \in \{1, \ldots, k\} \) denotes the class it falls into. Here \( k \) is the number of categories in the problem. A classification rule \( \phi(x) \) that maps \( x \) to the class label \( y \) is sought to learn. Each \((x_i, y_i)\) is assumed to be an independent random observation from a target population with probability distribution \( P(x, y) \). For the support vector machine in general, see Cristianini and Shawe-Taylor (2000) and Schölkopf and Smola (2002). We briefly review the SVM in the standard case of the equal misclassification costs and equal data weights, and then move on to the general case of either different costs or unequal data weights.

2.1 Standard Case

In the binary case with \( k = 2 \), the class labels \( y \) are coded either 1 or \( -1 \). Instead of finding a category-valued mapping \( \phi \) directly, the SVM looks for a real valued function \( f \) whose sign induces a classification rule via \( \phi(x) = \text{sign}\{f(x)\} \). As a regularization method in a reproducing kernel Hilbert space (RKHS), the SVM finds
a function \( f(x) = b + h(x) \) with \( h \in H_K \), an RKHS and \( b \), a constant that minimizes
\[
\frac{1}{n} \sum_{i=1}^{n} \{1 - y_i f(x_i)\}_+ + \lambda \|h\|_{H_K}^2. \tag{2.1}
\]
Here \((x)_+ = \max(x, 0)\), and \( \|h\|_{H_K}^2 \) denotes the square norm of the function \( h \) defined in the reproducing kernel Hilbert space with the reproducing kernel \( K \). \( \lambda \) is a tuning parameter that balances the data-fit and the complexity of \( f \). By the representer theorem, the minimizer \( \hat{f} \) is given as
\[
\hat{f}(x) = b + \sum_{i=1}^{n} c_i K(x_i, x). \tag{2.2}
\]
Hastie et al. (2004) showed the piecewise linearity of \( b \) and \( c_i \) in \( 1/\lambda \) and how to generate the entire regularized solution path.

The Multicategory SVM proposed by Lee et al. (2004b) is a general classification method that extends good theoretical properties of the binary SVM to the multiclass case such as the Bayes consistency. It is formulated via the general scheme of finding \( k \) functions \( (f^1, \ldots, f^k) \) whose maximum component determines the predicted class, that is, \( \phi(x) = \arg \max_j f^j(x) \). Each component \( f^j(x) \) is assumed to be of the form \( b^j + h^j(x) \) with \( h^j \in H_K \). In place of the nominal class label \( y_i \), a vector valued class code \( \mathbf{y}_i = (y^1_i, \ldots, y^k_i) \) is defined as the contrast vector with \( y^j_i = 1 \) and \(-1/(k-1)\) elsewhere if \( y_i = j \). Then the SVM solution is defined to be the minimizer \( \hat{f} = (\hat{f}^1, \ldots, \hat{f}^k) \) of
\[
\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1, j \neq y_i}^{k} \{f^j(x_i) - y^j_i\}_+ + \frac{\lambda}{2} \sum_{j=1}^{k} \|h^j\|_{H_K}^2 \tag{2.3}
\]
with the sum-to-zero constraint \( \sum_{j=1}^{k} f^j(x) = 0 \). By the representer theorem, each \( \hat{f}^j \) is of the form
\[
\hat{f}^j(x) = b^j + \sum_{i=1}^{n} c^j_i K(x_i, x) \text{ for } j = 1, \ldots, k. \tag{2.4}
\]
2.2 General Case

In the standard case, the costs for different types of misclassification are assumed to be the same, and the data pairs have equal weights. In many classification problems, different types of misclassification may result in different costs. For instance, in medical diagnosis, misclassifying a healthy individual as diseased would have vastly different consequence than the other type of error. This consideration calls for unequal misclassification costs. Another extension of the standard SVM formulation aside from differential costs is to allow different weights to the data pairs \((x_i, y_i)\). When we have unequal probabilities for the data points, such an extension can naturally incorporate them as weights. It also proves to be useful for coping with large data sets, which is to be discussed later in detail. In this section, we consider the setup with unequal misclassification costs and unequal data weights.

To generalize the so-called hinge loss in the standard case, let \(L_j^j\) be the cost of misclassifying \(j\) as \(j'\), and define the misclassification cost vector \(L(y_i) = (L_1 y_i, \ldots, L_k y_i)^t\). Let \(w_i\) be the weight of the \(i\)th data point. Then, the generalized loss function is

\[
L(y_i, f(x_i)) = w_i L(y_i)^t (f(x_i) - y_i)_+.
\]

It can be written explicitly as \(\sum_{j=1}^k w_i L_{y_i}(f^j(x_i) - y_i_j)_+\). For equal misclassification costs, \(L_j^j = I(j \neq j')\), it reduces to \(L(y_i, f(x_i)) = \sum_{j \neq y_i} (f^j(x_i) + 1/(k-1))_+\) in the previous section. In this general case, the MSVM solution \(\hat{f}_\lambda(x) = (\hat{f}_1^1(x), \ldots, \hat{f}_\lambda^k(x))\) given \(\lambda\) is defined as the minimizer of

\[
\frac{1}{n} \sum_{i=1}^n w_i L(y_i)^t (f(x_i) - y_i)_+ + \frac{\lambda}{2} \sum_{j=1}^k \|h^j\|^2, \tag{2.5}
\]

with the sum-to-zero constraint \(\sum_{j=1}^k f^j(x) = 0\) for any \(x \in \mathbb{R}^p\). Again, by the representer theorem, \(\hat{f}_\lambda = (\hat{f}_1^1, \ldots, \hat{f}_\lambda^k)\) is of the form

\[
\hat{f}_\lambda^j(x) = b^j + \sum_{i=1}^n c_i^j K(x_i, x) \text{ for } j = 1, \ldots, k, \tag{2.6}
\]
where $K(s, t)$ is the reproducing kernel of $\mathcal{H}$.

### 2.3 Optimality Conditions

The main focus of this thesis is how to explicitly characterize the coefficient paths of $b^j$ and $c^j$ of the solution as a function of $\lambda$. For expositions to follow, we briefly discuss the optimization problems associated with (2.5). Let the coefficient vector $c^j = (c^j_1, \ldots, c^j_n)^t$ for $j = 1, \ldots, k$, $b = (b^1, \ldots, b^k)^t$, and $C = (c^1, \ldots, c^k)$. With some abuse of notation, let bold-faced $K$ stand for an $n$ by $n$ matrix with the $lm$th entry $K(x_l, x_m)$. Consider a generalized cost matrix $L_w$ with the $ij$th entry $w_i L_j y_i$. Also, let $L'_w$ denote the $j$th coordinates of the $n$ generalized cost vectors, $(w_1 L'_j y_1, \ldots, w_n L'_j y_n)^t$ and $y^j = (y^j_1, \ldots, y^j_n)^t$. Then the MSVM in (2.5) can be rewritten as the problem of finding $(b, C)$ to minimize

$$L_P(b, C) = \frac{1}{n} \sum_{j=1}^{k} (L'_w)^t (b^j e + Kc^j - y^j) + \frac{\lambda}{2} \sum_{j=1}^{k} (c^j)^t Kc^j$$

(2.7)

subject to $\sum_{j=1}^{k} (b^j e + Kc^j) = 0$,

(2.8)

where $e$ is the vector of $n$ ones. To handle the truncate function $(x)_+$ in (2.7), we introduce nonnegative slack variables denoted by $\xi^j = (\xi^j_1, \ldots, \xi^j_n)^t$ for $j = 1, \ldots, k$. Let $\xi = (\xi^1, \ldots, \xi^k)$. By using the slack variables, (2.7) and (2.8) can be reformulated as finding $b, C,$ and $\xi$ that minimize

$$L_P(b, C, \xi) = \frac{1}{n} \sum_{j=1}^{k} (L'_w)^t \xi^j + \frac{\lambda}{2} \sum_{j=1}^{k} (c^j)^t Kc^j$$

(2.9)

subject to $b^j e + Kc^j - y^j \leq \xi^j$, for $j = 1, \ldots, k$,

(2.10)

$\xi^j \geq 0$, for $j = 1, \ldots, k$; and

(2.11)

$$\sum_{j=1}^{k} (b^j e + Kc^j) = 0.$$

(2.12)
For the Lagrangian dual formulation of the problem, we introduce nonnegative Lagrange multipliers $\alpha^j = (\alpha^j_1, \ldots, \alpha^j_n)^t \in \mathbb{R}^n$ for (2.10), $\gamma^j \in \mathbb{R}^n$ for (2.11), and unconstrained multipliers $\delta_f \in \mathbb{R}^n$ for (2.12). Then the dual problem becomes

$$
\max L_D = \sum_{j=1}^{k} (L^j_w)^t \xi^j + \frac{n\lambda}{2} \sum_{j=1}^{k} (c^j)^t K c^j + \sum_{j=1}^{k} (\alpha^j)^t (b^j e + K c^j - y^j - \xi^j)
$$

$$
- \sum_{j=1}^{k} (\gamma^j)^t \xi^j + \delta_f^t \sum_{j=1}^{k} (b^j e + K c^j)
$$

subject to, for $j = 1, \ldots, k$,

$$
\frac{\partial L_D}{\partial \xi^j} = L^j_w - \alpha^j - \gamma^j = 0, \quad (2.13)
$$

$$
\frac{\partial L_D}{\partial c^j} = n\lambda K c^j + K \alpha^j + K \delta_f = 0, \quad (2.14)
$$

$$
\frac{\partial L_D}{\partial b^j} = (\alpha^j + \delta_f)^t e = 0, \quad (2.15)
$$

$$
\alpha^j \geq 0 \text{ and } \gamma^j \geq 0.
$$

Letting $\bar{\alpha} = (\sum_{j=1}^{k} \alpha^j)/k$, we have $(\alpha^j - \bar{\alpha})^t e = 0$ by taking the unconstrained $\delta_f = -\bar{\alpha}$ in (2.15) and $c^j = -(\alpha^j - \bar{\alpha})/(n\lambda)$ from (2.14). Using these relations and (2.13), and denoting $(\alpha^1, \ldots, \alpha^k)$ by $\alpha$, we have the dual problem of

$$
\min L_D(\alpha) = \frac{1}{2} \sum_{j=1}^{k} (\alpha^j - \bar{\alpha})^t K (\alpha^j - \bar{\alpha}) + n\lambda \sum_{j=1}^{k} (\alpha^j)^t y^j
$$

subject to $0 \leq \alpha^j \leq L^j_w$ for $j = 1, \ldots, k$, (2.17)

$$
(\alpha^j - \bar{\alpha})^t e = 0 \quad \forall j = 1, \ldots, k.
$$

Note that the $\alpha^j_1$'s corresponding to zero $L^j_{y^j}$'s are trivially zero, so the above dual problem involves only $n(k - 1)$ Lagrange multipliers. Throughout this thesis, we consider only the $n(k - 1)$ non-trivial $\alpha^j_1$. By the Karush-Kuhn-Tucker (KKT) complementarity conditions, the solution satisfies

$$
\alpha^j \perp (b^j e + K c^j - y^j - \xi^j) \quad \forall j = 1, \ldots, k, \quad (2.19)
$$

$$
\gamma^j = (L^j_w - \alpha^j) \perp \xi^j \quad \forall j = 1, \ldots, k, \quad (2.20)
$$
Figure 2.1: MSVM component loss \((f^j - y^j)_+\) where \(y^j = -1/(k-1)\).

where \(\perp\) indicates that componentwise product of two vectors is zero. For instance, if \(0 < \alpha^j_i < w_i L^j_{y_i}\) for some \(i\), then \(\xi^j_i\) should be zero from (2.20), and this implies

\[ b^j + \sum_{l=1}^n c^j_i K(x_l, x_i) - y^j_i = 0 \] from (2.19). The KKT conditions categorize each component of \(\hat{f}_\lambda(x_i)\) as one of three types, defining three different sets. To refer to the three sets of indices, we borrow the names from Hastie et al. (2004) but slightly modify them as follows. Abbreviating \(\hat{f}_\lambda^j(x_i)\) as \(f^j_i\),

- \(\mathcal{E} = \{(i, j) \mid f^j_i - y^j_i = 0, \ \xi^j_i = 0, \ 0 \leq \alpha^j_i \leq w_i L^j_{y_i}\}\), an elbow set,
- \(\mathcal{U} = \{(i, j) \mid f^j_i - y^j_i > 0, \ \xi^j_i > 0, \ \alpha^j_i = w_i L^j_{y_i}\}\), an upper set of the elbow, and
- \(\mathcal{L} = \{(i, j) \mid f^j_i - y^j_i < 0, \ \xi^j_i = 0, \ \alpha^j_i = 0\}\), a lower set of the elbow.

Figure 2.1 depicts the \(j\)th component of the MSVM hinge loss \((f^j - y^j)_+\) as a function of \(f^j\) with \(y^j = -1/(k-1)\). The elbow set \(\mathcal{E}\) consists of indices of data points falling on the soft margin of the MSVM solution, while the lower set \(\mathcal{L}\) is associated with non-support vectors.
CHAPTER 3

SOLUTION PATH IN THE STANDARD CASE

To describe how MSVM solutions in (2.6) change as a function of the regularization parameter $\lambda$, we begin with a very large value of the parameter at which the initial set of dual variables $\alpha^j$ is easily determined. Then a constructive algorithm is laid out for successive update of the dual minimizers as $\lambda$ decreases. From the relation between the coefficients and the dual variables, $c^j = -(\alpha^j - \bar{\alpha})/(n\lambda)$, $c^j = 0$ as $\lambda$ goes to $\infty$. So, only $b^j$ and $\alpha^j$ need to be initialized for a sufficiently large $\lambda$. For brevity, equal misclassification costs are considered. Generalization of the following to unequal misclassification costs and unequal data weight will be discussed in Chapter 4.

3.1 Initialization

Let $\mathcal{I}_j$ be the index set of observations in class $j$ and $n_j = |\mathcal{I}_j|$, the number of instances in class $j$. Since initialization of $b^j$ and $\alpha^j$ depends on which class is the largest in terms of sample size, we define $\mathcal{M} = \arg\max n_j$ and $n_{\mathcal{M}} = \max n_j$ first. Lemma 1 below is concerned with initialization when $|\mathcal{M}| = 1$, that is, there is a unique class with the maximum sample size, while Lemma 2 is for $|\mathcal{M}| > 1$. The
results presented here subsume Lemma 1 and Lemma 2 in Hastie et al. (2004) for the binary case.

**Lemma 1.** Suppose there is only one class $j^*$ with maximum sample size. For a sufficiently large $\lambda$, $b_j = 1$ if $j = j^*$, and $-1/(k - 1)$ otherwise. $\alpha$ minimizes
\[
\sum_{j=1}^k (\alpha_j - \bar{\alpha})^t K(\alpha_j - \bar{\alpha})
\]
subject to
\[
0 \leq \alpha_i \leq 1 \quad \text{for} \quad j \neq j^* \quad \text{and} \quad i \notin I_j,
\]
\[
\alpha_{j^*} = 1 \quad \text{for} \quad i \notin I_{j^*},
\]
\[
\sum_i \alpha_{j} = n - n_{j^*} \quad \text{for} \quad j \neq j^*.
\]

**Proof.** For a sufficiently large $\lambda$, the minimizer of (2.7) is a constant vector $(b_1, \ldots, b_k)$, for which (2.7) is reduced to $\sum_{j=1}^k (n - n_j)(b_j + 1/(k - 1))$ up to a multiplicative constant. To minimize the objective function, it is sufficient to assume that all the $b_i$’s are at least $-1/(k - 1)$. Otherwise, we can always find $b_i$’s that make the objective function smaller. Thus, it amounts to finding $(b_1, \ldots, b_k)$ minimizing $\sum_{j=1}^k (n - n_j)(b_j + 1/(k - 1)) = \sum_{j \neq j^*} (n_{j^*} - n_j)b_j + n$. Since this is a non-negatively weighted sum of $b_i$’s for $j \neq j^*$, the sum becomes smallest when $b_j = -1/(k - 1)$ for $j \neq j^*$, and consequently $b_{j^*} = 1$ by the sum-to-zero constraint. The rest follows from (2.16), (2.17), and (2.18) by observing the following three facts. First, $\xi_{i}^{j^*} = (b_i^{j^*} - y_{i}^{j^*})_+ = k/(k - 1)$ for $i \notin I_{j^*}$, thus $\alpha_i^{j^*} = 1$ by the KKT conditions of (2.19) and (2.20). Second, (2.18) is then restated as $\sum_i \alpha_{i} = n - n_{j^*}$ for all $j$. Third, as a result, $\sum_{j=1}^k (\alpha_i^j)^t y_j = -1/(k - 1) \sum_{j=1}^k \sum_i \alpha_i^j$ is fixed at $-k/(k - 1)(n - n_{j^*})$. \[\Box\]

**Remark 1.** The value of the primal objective function (2.9) for the initial $b$ is $(n - n_{j^*})k/(k - 1)$ except for the multiplicative constant $1/n$. Lemma 1 is a generalized version of Lemma 2 for the unbalanced binary case in Hastie et al. (2004).
Lemma 2. Suppose that there is more than one class in $\mathcal{M} = \text{argmax} n_j$. For a sufficiently large $\lambda$, $b^j = -1/(k-1)$ for $j \notin \mathcal{M}$, and $b^j \geq -1/(k-1)$ for $j \in \mathcal{M}$ with $\sum_{j \in \mathcal{M}} b^j = (k - |\mathcal{M}|)/(k-1)$. $\alpha$ minimizes $\sum_{j=1}^{k} (\alpha^j - \bar{\alpha})^t \mathbf{K} (\alpha^j - \bar{\alpha})$ subject to

$$0 \leq \alpha^j_i \leq 1 \quad \text{for } j \notin \mathcal{M} \text{ and } i \notin \mathcal{I}_j,$$

$$\alpha^j_i = 1 \quad \text{for } j \in \mathcal{M} \text{ and } i \notin \mathcal{I}_j,$$

$$\sum_i \alpha^j_i = n - n_{\mathcal{M}} \quad \text{for } j \notin \mathcal{M}. \quad (3.4)$$

Proof. By the same arguments used in the proof of Lemma 1, finding the minimizer $(b^1, \ldots, b^k)$ of the primal objective function leads to searching $(b^1, \ldots, b^k)$ that minimizes $\sum_{j \notin \mathcal{M}} (n_{\mathcal{M}} - n_j) b^j + n$ with $b^j \geq -1/(k-1)$ for $j \notin \mathcal{M}$ and the remaining $b^j$’s are arbitrary except that they satisfy $\sum_{j \in \mathcal{M}} b^j = (k - |\mathcal{M}|)/(k-1)$ by the sum-to-zero constraint. From the equality constraint on $b^j$’s with $j \in \mathcal{M}$, we infer that there is at least one $j^* \in \mathcal{M}$ such that $b^{j^*} > -1/(k-1)$. Then the KKT conditions of (2.19) and (2.20) imply that $\alpha^j_{i^*} = 1$ for $i \notin \mathcal{I}_{j^*}$ since $\xi^j_i = (b^{j^*} - y_i^{j^*})_+ > 0$. However, by (2.18), $\sum_i \alpha^j_i$ should be the same for all $j$, which implies that $\alpha^j_i = 1$ for other $j \in \mathcal{M}$ and $i \notin \mathcal{I}_j$ to have the same sum of $n - n_{\mathcal{M}}$. This proves (3.4) and (4.5) in particular, and the rest follows immediately. \qed

Remark 2. The value of (2.9) for the initial $b$ in Lemma 2 is $(n - n_{\mathcal{M}})k/(k-1)$ except for the multiplicative constant $1/n$. In fact, Lemma 1 is a special case of Lemma 2. Each $b^j$ with $j \in \mathcal{M}$ can be chosen to be $(k/|\mathcal{M}| - 1)/(k-1)$ for computational ease. If $k$ classes are completely balanced, that is $\mathcal{M} = \{1, \ldots, k\}$, then $\alpha^j_i = 1$ for each $j$ and $i \notin \mathcal{I}_j$. Again, Lemma 2 is a generalized version of Lemma 1 for the balanced binary case in Hastie et al. (2004).
We start from $\lambda$ sufficiently large but indefinite, which determines the limit MSVM solution in the foregoing two lemmas, and decrease $\lambda$ until non-trivial solutions emerge. To find such a genuine starting value of $\lambda$ and the corresponding $b$, we consider two possible cases of the limit solution. First, $\alpha_i^j \in \{0, 1\}$ for all $j$ and $i \notin I_j$. Due to Lemma 2, the completely balanced situation leads to this case. Unbalanced class proportions may also lead to this case. As $\lambda$ decreases, $\alpha$ changes, but with the restriction that the equality constraint (2.18) is satisfied. (2.18) states that the sum of the Lagrange multipliers $\alpha_i^j$ is the same across all $j = 1, \ldots, k$. Any change in $\alpha$ is bound to reduce $\alpha_i^j$ with $j \in M$ from 1 because the values of such $\alpha_i^j$’s in the limit solution are at their maxima. Consequently, this change would reduce some $\alpha_i^j$ with $j \notin M$ from 1 as well, by (2.18). Hence, some $k$ indices of $\alpha_i^j$, one from each $j$, should enter the elbow set $E$ simultaneously. Note that the lower set of the limit solution is empty. Letting $B^j = \{i \mid \alpha_i^j = 1\}$ for each $j$, we choose the data index $i^j_* = \arg\min_{i \in B^j} H_i^j$, where $H_i^j = -\sum_{r=1}^n (\alpha_i^r - \bar{\alpha}_r) K(x_r, x_i)$. These $k$ indices are chosen to satisfy $f^j(x_{i^j_*}) = -1/(k-1)$, yielding $k$ equations that determine the initial $\lambda$ and $b$.

Second, there could be two or more $0 < \alpha_i^j < 1$ for some $j \notin M$, by (4.3) and (4.5). By the same logic as in the first case, any change in $\alpha$ would reduce $\alpha_i^j$ from 1 for other component(s) $j$ with $\alpha_i^j = 0$ or 1 only. By Lemmas 1 and 2, there is at least one component $j \in M$ for which no data index falls into the elbow set of the limit solution. In this case, $\alpha_i^j$ for $(i, j) \in E$ stays the same until other components $j$ without such index $i$ have a point reaching the margin $f_i^j = y_i^j$. A formal proof of this fact is given in the next section. As a result, $\alpha_i^j$ strictly between 0 and 1 will remain on the elbow set until each of the other component(s) $j$ has a data index in
the elbow set. So if there is \( l \) such that \( 0 < \alpha_l^j < 1 \) for \( j \), we define \( \hat{i}_l^j = l \), otherwise \( \hat{i}_l^j = \arg\min_{i \in B^j} H_i^j \). Again, \( f^j(x_{i_l^j}) = -1/(k - 1) \) gives a set of \( k \) equations as follows. For \( j = 1, \ldots, k \),

\[
\begin{align*}
b^j &= -\frac{1}{k - 1} + \frac{1}{n\lambda} \sum_{r=1}^{n} (\alpha_r^j - \bar{\alpha}_r) K(x_r, x_{i^*_l}) \\
\lambda &= \frac{k - 1}{kn} \sum_{j=1}^{k} \sum_{r=1}^{n} (\alpha_r^j - \bar{\alpha}_r) K(x_r, x_{i^*_l}) \quad \text{and} \quad (3.6) \\
b^j &= \frac{1}{k - 1} \sum_{j=1}^{k} b^j = 0. \quad (3.7)
\end{align*}
\]

Solving the equations, we have the initial \( \lambda \) and \( b \) in both scenarios:

\[
\begin{align*}
\lambda &= \frac{k - 1}{kn} \sum_{j=1}^{k} \sum_{r=1}^{n} (\alpha_r^j - \bar{\alpha}_r) K(x_r, x_{i^*_l}) \quad \text{and} \quad (3.6) \\
b^j &= \frac{1}{k - 1} \sum_{j=1}^{k} b^j = 0. \quad (3.7)
\end{align*}
\]

Since the above initial \( \lambda \) and \( b \) depend on \( \hat{i}_l^j \) only through the value of \( H_i^j \), they are uniquely determined regardless of the choice of \( \hat{i}_l^j \) with the minimum \( H_i^j \).

### 3.2 Characterizing Coefficient Paths

As seen in the previous initialization step, the elbow set permits explicit equations of \( f^j_i = y_i^j \) for \((i, j) \in E\). Accordingly, this allows us to find some of the Lagrange multipliers fully and determine the coefficients in (2.6). As a result, our strategy for constructing the coefficient paths is to keep track of changes in the elbow set. There are three types of events that can change the elbow set.

1. An index \((i, j)\) leaves from \( E \) to join either \( \mathcal{L} \) or \( \mathcal{U} \).

2. An index \((i, j)\) from \( \mathcal{L} \) enters \( E \).

3. An index \((i, j)\) from \( \mathcal{U} \) enters \( E \).

Continuity of the objective function (2.16) in \( \lambda \) implies that the minimizer \( \alpha \), as a function of \( \lambda \), changes continuously between consecutive values of \( \lambda \) at which
one of the above three events occurs, and so does the MSVM solution. Consider
\{\lambda_\ell, \ell = 0, 1, \ldots \}, a decreasing sequence of \lambda starting from the initial value in (4.6)
and indicating the values at which some change occurs in \mathcal{E}. In fact, the sequence
determines the break points of \lambda at which \alpha can be completely characterized. For \lambda_\ell,
denote the corresponding elbow, upper and lower sets by \mathcal{E}_\ell, \mathcal{U}_\ell, and \mathcal{L}_\ell respectively.
Letting \alpha_0^j = n\lambda b^j, we write
\[
\hat{f}_\lambda^j(x) = \frac{1}{n\lambda} \left( -\sum_{i=1}^n (\alpha_i^j - \bar{\alpha}_i)K(x_i, x) + \alpha_0^j \right).
\]
For \lambda_{\ell+1} < \lambda < \lambda_\ell, we can express \hat{f}_\lambda^j(x) in terms of an incremental change from
\hat{f}_\lambda^j(x). Denoting \alpha_i^j at \lambda_\ell by \alpha_i^{j(\ell)},
\[
\hat{f}_\lambda^j(x) = \left[ \hat{f}_\lambda^j(x) - \frac{\lambda_\ell}{\lambda} \hat{f}_\lambda^j(x) \right] + \frac{\lambda_\ell}{\lambda} \hat{f}_\lambda^j(x)
= \frac{1}{n\lambda} \left[ -\sum_{i=1}^n ((\alpha_i^j - \alpha_i^{j(\ell)}) - (\bar{\alpha}_i - \bar{\alpha}_{i(\ell)}))K(x_i, x) + (\alpha_0^j - \alpha_0^{j(\ell)}) + n\lambda_\ell \hat{f}_\lambda^j(x) \right]
= \frac{1}{n\lambda} \left[ -\sum_{i\in\cup_j \mathcal{E}_\ell^j} ((\alpha_i^j - \alpha_i^{j(\ell)}) - (\bar{\alpha}_i - \bar{\alpha}_{i(\ell)}))K(x_i, x) + (\alpha_0^j - \alpha_0^{j(\ell)}) + n\lambda_\ell \hat{f}_\lambda^j(x) \right],
\]
where \mathcal{E}_\ell^j = \{i | (i, j) \in \mathcal{E}_\ell\} for each \ell. The last equality holds because \alpha_i^j = 0 or 1
for all \ell without any change from \alpha_i^{j(\ell)} if \ell \notin \cup_j \mathcal{E}_\ell^j. For all \ell, \ell \in \mathcal{E}_\ell, \hat{f}_\lambda^j(x) = y_i^j.
Letting \delta_{0}^j = \alpha_0^j - \alpha_0^{j(\ell)}, \delta_i^j = -(\alpha_i^j - \alpha_i^{j(\ell)}), \bar{\delta}_i = -(\bar{\alpha}_i - \bar{\alpha}_{i(\ell)}) for \ell \geq 1, and \mathcal{E}_\ell^j = \cup_j \mathcal{E}_\ell^j,
we have
\[
\hat{f}_\lambda^j(x) = \frac{1}{n\lambda} \left[ \sum_{r\in \mathcal{E}_\ell^j} (\delta_r^j - \bar{\delta}_r)K(x_r, x) + \delta_0^j - \frac{n\lambda_\ell}{k - 1} \right] = -\frac{1}{k - 1},
\]
which gives
\[
\sum_{r\in \mathcal{E}_\ell^j} (\delta_r^j - \bar{\delta}_r)K(x_r, x) + \delta_0^j = \frac{n(\lambda_\ell - \lambda)}{k - 1}. \tag{3.8}
\]
Given any \lambda, \sum_{i=1}^n \alpha_i^j should be the same for all \ell and \sum_{j=1}^k b^j = 0 by the sum-tozero constraint. This yields \sum_{i\in \mathcal{E}_\ell^j} \delta_i^j = \cdots = \sum_{i\in \mathcal{E}_\ell^j} \delta_k^j and \sum_{j=1}^k \delta_0^j = 0. As a result,
these \( k \) constraints and \( (3.8) \) provide a set of \( |\mathcal{E}_\ell| + k \) equations to solve for \( |\mathcal{E}_\ell| + k \) unknowns if all the \( k \) elbow sets \( \mathcal{E}_\ell^j \) are non-empty. To re-express \( (3.8) \) conveniently in a vector notation, let \( m_j = |\mathcal{E}_\ell^j| \) and \( i_1^j, \ldots, i_{m_j}^j \) denote the \( m_j \) data indices in \( \mathcal{E}_\ell^j \).

Now we define \( \delta_0 = (\delta_0^1, \ldots, \delta_0^k) \) and \( \delta = (\delta^1, \ldots, \delta^k) \) with \( \delta^j = (\delta_{i_1}^j, \ldots, \delta_{i_{m_j}}^j) \). Note that \( (3.8) \) depends only on \( \delta_0 \) and \( \delta \) since \( \delta_i^j = 0 \) for all \((i, j) \notin \mathcal{E}_\ell \). \( K^*_\ell = [K^*_{ij}] \) is the square block matrix of \( |\mathcal{E}_\ell| = m_1 + \cdots + m_k \) rows and columns, whose \( lj \)th block \((l, j = 1, \ldots, k) \) is given by

\[
K^*_{ij} = \left( I(l = j) - \frac{1}{k} \right) \begin{pmatrix} K(x_{i_1^j}, x_{i_1^j}) & \cdots & K(x_{i_{m_j}^j}, x_{i_1^j}) \\ \vdots & \ddots & \vdots \\ K(x_{i_1^j}, x_{i_{m_j}^j}) & \cdots & K(x_{i_{m_j}^j}, x_{i_{m_j}^j}) \end{pmatrix}.
\]

Also, define

\[
1_\delta = \begin{pmatrix} e_{m_1}^t \\ -e_{m_2}^t & 0 & \cdots & 0 \\ 0 & e_{m_2}^t & -e_{m_3}^t & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & e_{m_{k-1}}^t & -e_{m_k}^t \end{pmatrix} \quad \text{and} \quad 1_0 = \begin{pmatrix} e_{m_1} \\ 0 & e_{m_2} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & e_{m_k} \end{pmatrix},
\]

where \( e_m \) is the vector of \( m \) ones and \( 0 \) indicates a vector of zeros of appropriate length. Then, \( (3.8) \) and the constraints are succinctly expressed as

\[
\begin{pmatrix} 0 \\ K^*_\ell \\ 1_\delta \end{pmatrix} \begin{pmatrix} e_k^t \\ \delta \\ \delta_0 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{n(\lambda - \lambda)}{k-1}e_{|\mathcal{E}_\ell|} \\ 0 \end{pmatrix}.
\]

(3.9)

Letting \( A_\ell \) be the square matrix on the left-hand side of (3.9), and \( v_\ell = (0, e_{|\mathcal{E}_\ell|}, 0) \), we solve for \( \delta_0 \) and \( \delta \). If \( A_\ell \) is invertible,

\[
\begin{pmatrix} \delta \\ \delta_0 \end{pmatrix} = \frac{n(\lambda - \lambda)}{k-1}A_\ell^{-1}v_\ell.
\]

Abbreviating \( A_\ell^{-1}v_\ell \) by \( w_\ell \), we have

\[
\alpha_0 - \alpha_{0(\ell)} = n(\lambda - \lambda)/(k - 1)w_0 \quad \text{and} \quad \alpha_i - \alpha_{i(\ell)} = n(\lambda - \lambda)/(k - 1)w_i \quad \text{for} \ (i, j) \in \mathcal{E}_\ell, \]

(3.10)
where $\mathbf{w}_0$ is the last $k$ elements of $\mathbf{w}_\ell$ and $w^j_i$ is the element of $\mathbf{w}_\ell$ corresponding to $\delta^j_i$. This shows that the scaled intercepts and the Lagrange multipliers in the elbow set change linearly in $\lambda$ on the interval $(\lambda_{\ell+1}, \lambda_\ell)$. Rescaling them properly by $n\lambda$ to obtain equations for the coefficients, we have

$$b = \frac{\lambda_\ell}{\lambda}(b_\ell + \frac{1}{k-1}\mathbf{w}_0) - \frac{1}{k-1}\mathbf{w}_0 \text{ and}$$

(3.11)

$$c^j_i = \frac{\lambda_\ell}{\lambda}(c^j_i(\ell) + \frac{w^j_i - \bar{w}_i}{k-1}) - \frac{w^j_i - \bar{w}_i}{k-1}. \quad (3.12)$$

Here $\bar{w}_i = (1/k) \sum_j w^j_i$ and the summation is only over $j$’s with $(i, j) \in E_\ell$. This proves the following theorem concerning piecewise linearity of the paths of the coefficients $b$ and $C$.

Theorem 1. If there is at least one data index in the elbow set $E_\ell$ at $\lambda_\ell$ for each $j$, then the coefficient path of the MSVM is linear in $1/\lambda$ on the interval $(\lambda_{\ell+1}, \lambda_\ell)$.

Likewise, the $j$th coordinate of the MSVM output has a path linear in $1/\lambda$:

$$\hat{f}^j_\lambda(x) = \frac{\lambda_\ell}{\lambda}\left(\hat{f}^j_\lambda(x) - \hat{g}^j_\lambda(x)\right) + \hat{g}^j_\lambda(x),$$

(3.13)

where $\hat{g}^j_\lambda(x) = -\left(\sum_{i\in E^j_\ell}(w^j_i - \bar{w}_i)K(x, x_i) + w^j_0\right)/(k-1)$. $\hat{g}^j_\lambda(x)$ is pivotal to $\hat{f}^j_\lambda(x)$ in the sense that $\hat{f}^j_\lambda(x)$ can be expressed as a scaled $\hat{f}^j_\lambda(x)$, once both are pivoted on $\hat{g}^j_\lambda(x)$.

So far, we have discussed how the MSVM solution path is explicitly characterized as a function of $\lambda$ when the elbow set $E^j_\ell$ for each $j$ is not empty. If there is at least one empty elbow set $E^j_\ell$ at $\lambda = \lambda_\ell$, then the constraints used in the previous characterization need to be modified. The constraint that the sum of $\alpha^j_i$ should stay the same for all $j$ now becomes $\sum_{i\in E^j_\ell} \delta^j_i = 0$ for each non-empty elbow set $E^j_\ell$. We eliminate the component(s) corresponding to the empty $E^j_\ell$ from $\delta_0$ and the
corresponding column(s) from $1_0$, and denote the resulting vector and matrix by $\delta^*_0$ and $1^*_0$, respectively. Then (3.9) is adjusted by taking into account the presence of some empty elbow set(s):

$$
\begin{pmatrix}
    K^*_\ell & 1^*_0 \\
    1^*_0 & 0
\end{pmatrix}
\begin{pmatrix}
    \delta \\
    \delta^*_0
\end{pmatrix} = 
\begin{pmatrix}
    \frac{n(\lambda - \lambda)}{k-1} e_{|E_\ell|} \\
    0
\end{pmatrix}.
\quad (3.14)
$$

Assuming that $K^*_\ell$ is of full rank, we solve for $\delta$ and $\delta^*_0$. Due to the simple structure of the block matrix on the left-hand side, its inverse can be easily written out to give the explicit solution of $\delta = 0$ and $\delta^*_0 = n(\lambda - \lambda)/(k-1)e_{k^*}$, where $k^*$ is the number of non-empty elbow sets $E^j_\ell$. This yields

$$
c^j_i = \frac{\lambda}{\lambda} c^j_{i(\ell)} \text{ for } (i, j) \in E_\ell \text{ and}
$$

$$
b^j = \frac{\lambda}{\lambda} (b^j_\ell + \frac{1}{k-1}) - \frac{1}{k-1} \text{ for non-empty } E^j_\ell.
$$

When there is more than one empty $E^j_\ell$, $\delta^j_0$’s for the empty sets are not uniquely determined other than that they are constrained to satisfy $\sum_j \delta^j_0 = 0$. In practice, a linear path can be chosen for such $b^j$ corresponding to the empty elbow sets, for convenience. Theorem 1 and the following result give the desired conclusion: the solution path of the MSVM is piecewise linear.

**Theorem 2.** If there is only one empty elbow set $E^j_\ell$ at $\lambda_\ell$, then the coefficient path of the MSVM is linear in $1/\lambda$ on the interval $(\lambda_\ell+1, \lambda_\ell)$. If there is more than one empty elbow set, then the coefficient path is still linear in $1/\lambda$ except that the path of $b^j$ for the empty $E^j_\ell$ can be arbitrary.
3.3 Computation

We show here how to generate the decreasing sequence of \( \{\lambda_\ell, \ell = 0, 1, \ldots\} \) that determines the joints of the piecewise linear MSVM solution path. Given \( \lambda_\ell \), we find \( \lambda_{\ell+1} \) by considering the following possible events.

1. An index \((i, j)\) in \( E_\ell \) leaves the elbow set, and \( \alpha^j_i (0 \leq \alpha^j_i \leq 1) \) becomes either 0 or 1.

2. An index \((i, j)\) in \( L_\ell \) or \( U_\ell \) joins the elbow set, and \( \hat{f}^j_i(x_i) \) is then \( y^j_i \).

When the first type of event happens, a candidate \( \lambda_{\ell+1} \) is obtained by setting \( \alpha^j_i \) in (3.10) to 0 or 1. For the second type of event, set the left-hand side of (3.13) to \( y^j_i \) and consider

\[
\lambda_{\ell+1} = \frac{\lambda_\ell (\hat{f}^j_i(x_i) - \hat{g}^j_i(x_i))}{y^j_i - \hat{g}^j_i(x_i)}.
\]

The next break point \( \lambda_{\ell+1} \) is determined by the largest \( \lambda < \lambda_\ell \) among the potential candidate values.

If there is at least one empty elbow set \( E^j_\ell \) at \( \lambda_\ell \), then the \( \alpha^j_i \) in non-empty sets stay the same until the next event occurs. The ensuing event that changes the elbow set is of the second type in this case, and specifically it is the event that a point for each \( j \) with empty \( E^j_\ell \) hits the margin \( \hat{f}^j_i(x_i) = y^j_i \) simultaneously. Such a point for each empty elbow set \( E^j_\ell \) is determined via the same argument as in the initialization process. Thus, the corresponding \( \lambda_{\ell+1} \) is identified by (4.6).

We stop the process and trace out the solution path if the upper set \( U_\ell \) is empty, since the empirical risk functional at \( \lambda_\ell \) is zero in this case, completely overfitting the data. As \( \lambda \) gets smaller, the upper sets are bound to become empty in separable problems. For instance, with the linear kernel, as \( \lambda \) gets smaller, the margin becomes
smaller so that no points fall into the upper sets. For non-separable problems, monitoring change in the solutions at consecutive break points can shorten the procedure. Inspection of (3.13) provides a rule that stops at \( \lambda_\ell \) if \( \max_j \left( \frac{1}{n} \sum_{i=1}^{n} |\hat{f}^{j}_{\lambda_\ell}(x_i) - \hat{g}^{j}_{\lambda_\ell}(x_i)| < \epsilon \right) \), where \( \epsilon \) denotes a prespecified tolerance for declaring no change between successive solutions.

From a practical point of view, early stopping may be desired if one does not attempt to find the entire solution path, but wishes to keep track of solutions only until \( \lambda \) gets small enough to be in the vicinity of theoretically optimal values of the least error rate. As a related issue, it is worth noting that the computational cost of characterizing the MSVM solution path essentially lies in solving a system of equations with at most \( |E_\ell| + k \) unknowns at each \( \lambda_\ell \). As \( \lambda \) decreases, the cardinality of the elbow set \( E_\ell \) tends to increase, an example of which is to be shown in the following section. This is another motivation for devising an early stopping rule. The idea of early stopping presupposes a reasonable data-driven measure of predictive accuracy of the solution at \( \lambda_\ell \). Such a measure helps us judge whether the optimal value of the regularization parameter has been attained or not. If attained, then we stop without completing the entire path.

The computational complexity of the path finding algorithm is proportional to the number of break points \( \lambda_\ell \). At each break point, a system of linear equations needs to be solved. In general, solving a system of linear equations with \( m \) unknowns takes \( O(m^3) \) operations, but it can be reduced to \( O(m^2) \) in this case due to an incremental change in the successive linear equations. For determining the next break point, evaluation of \( \hat{f}^j_{\lambda_\ell} \) and \( \hat{g}^j_{\lambda_\ell} \) at \( n \) data points needs to be done, which takes \( O((k-1)n|E_\ell^T|) \) operations. The effect of the class size \( k \) is felt throughout the computation in that
$|\mathcal{E}_t|$, the intermediate function evaluations, and the number of break points tend to increase in proportion to $(k - 1)$ when compared to the binary case.

### 3.4 An Illustrative Example

To illustrate the algorithm characterizing the MSVM solution path, we consider a simulated three-class example. The simulation setting is as follows. For Class 1, two covariates $X = (X_1, X_2)$ are generated from a normal distribution with mean $(3,0)$. For Class 2, $X$ comes from a mixture of two normal distributions with mean $(0,3)$ and $(1,1)$, respectively. The mixing proportion is 0.5. For Class 3, $X$ has a normal distribution with mean $(-1,1)$. For all three classes, $X_1$ and $X_2$ are independent and have variance 1. A training data set of size $n = 300$ was generated from the specified distributions with $n_j = 100$ for each class. The left panel of Figure 5.1 depicts the training data denoted by circles (Class 1 in green, 2 in magenta, and 3 in blue) in a scatter plot, as well as the theoretically optimal classification boundaries. A Monte Carlo estimate of the Bayes error rate of this example was approximately 0.1773 with standard error 0.007 based on a test data set of size 3,000 with 1,000 cases from each class. The MSVM with the Gaussian kernel $K(s, t) = \exp(-\gamma \|s - t\|^2)$ was applied to the simulated data, and the entire solution path was traced. Here the additional parameter $\gamma$ was fixed at 1.

Figure 3.2 shows how the test error rates, when evaluated over the test set, change as the regularization parameter $\lambda$ varies. Notice that the $x$-axis is $\log(1/\lambda)$, so $\lambda$ decreases in the positive direction of $x$. As $\lambda$ decreases from the initial value, the test error rates get smaller, reach the minimum around $\log(\lambda) = -6$, and begin to increase soon after that, clearly demonstrating that the choice of $\lambda$ is critical to the
performance of the solution. Note that the test error rate curve in the figure was cropped to show a portion of the full range of $\lambda$, since the test error rates rise up sharply as $\lambda$ approaches the smaller end. The minimum error rate achieved by the MSVM classifier was about 0.1743 at $\log(\lambda) \approx -5.886$. This illustrates that the MSVM equipped with a flexible kernel can achieve theoretically optimal accuracy if $\lambda$ is chosen appropriately. The right panel of Figure 5.1 shows the classification boundaries induced by the MSVM solution at this optimal $\lambda$.

Figure 3.1: Left: the boundaries of the Bayes classification rule in a scatter plot of the training data. Right: the classification boundaries determined by the MSVM at an appropriately chosen $\lambda$; Class 1: green, Class 2: magenta, and Class 3: blue.

In practice, we have to face the problem of choosing the regularization parameter in the absence of test cases. Data-driven tuning is in itself a long standing research topic, and we will not delve into the issue here. General approaches to tuning, equivalently
model selection, can be found in, for example, Efron (1986) and Wahba (1990). Just for illustration of the feasibility of tuning, 100 new data sets of size 300 were generated as random replicates of a tuning set. For each tuning set, λ was chosen to minimize the error rate over that tuning set. Figure 3.3 shows an estimated density curve of such $\hat{\lambda}$’s. Bimodality of the distribution is discernible, perhaps due to the presence of multiple dips in the test error rate plot of Figure 3.2. The mean of the test error rates at tuned $\hat{\lambda}$’s values was 0.1815 with standard deviation of 0.00525. The mean is slightly larger than the Bayes error rate of about 0.1773, but reasonably close to it.

The length of the solution path we monitor, or simply the number of break points of λ, depends on the stopping criterion used. The rule adopted in this implementation of the algorithm does not consider the two stopping criteria discussed in Section 4 only, but also employs a preset lower bound of λ and a maximum number of break points, which could be rather arbitrary but nonetheless useful in practice. Whichever
criterion is met first, the algorithm stops. Based on the current working rule, the preset lower bound of $\lambda$ was reached first before any other criteria for this example, and there were 3014 breaking points of $\lambda$. Availability of the entire solution path allows us to visualize the effect of the regularization parameter on the MSVM solution from various angles. For example, Figure 3.4 depicts the complete paths of $\hat{f}_1^\lambda(x_i)$, $\hat{f}_2^\lambda(x_i)$, and $\hat{f}_3^\lambda(x_i)$ for the isolated instance from Class 3 (blue) in the top left region. The paths start off at the initial solution of $(0, 0, 0)$, begin to diverge for a better fit to the data as $\lambda$ decreases, returning Class 2 as the predicted class for moderate values of $\lambda$. As $\lambda$ further decreases, the solution paths tend to follow the data too closely and $\hat{f}_3^\lambda(x_i)$ emerges as the maximum component. This is a snapshot of the spectrum.
of solutions ranging from the least to the most complete fit to the data as controlled by $\lambda$.

Figure 3.4: The entire paths of $\hat{f}_1^\lambda(x_i)$, $\hat{f}_2^\lambda(x_i)$, and $\hat{f}_3^\lambda(x_i)$ for an outlying instance $x_i$ from Class 3. The circles correspond to $\lambda$ with the minimum test error rate.

As mentioned before, the computational complexity of the proposed algorithm depends on the size of elbow set at each $\lambda_\ell$, and the elbow size is bound to increase as $\lambda$ gets smaller. Figure 3.5 shows how the size of elbow set changes as a function $\lambda$ for each class. The median sizes of elbow sets were 39, 30, and 30 for Classes 1, 2, and 3, respectively, while the maximum elbow sizes were 68, 51, and 61. Around the optimal value of $\lambda$, roughly 40 to 50% of data points were in the elbow set for this example. It is interesting to observe that the rate of increase in the size of each elbow set $E_\ell^j$ is almost constant until $\lambda$ reaches the optimal value around $\exp(-6)$, and it becomes smaller soon after that, then remains almost constant. It would be
desirable to have a theoretical explanation for this, because this might help us devise an early stopping rule.

![Figure 3.5: The size of elbow set $\mathcal{E}_\ell^j$ for three classes as a function of $\lambda$.](image)

Figure 3.5: The size of elbow set $\mathcal{E}_\ell^j$ for three classes as a function of $\lambda$. 
CHAPTER 4

SOLUTION PATH FOR WEIGHTED DATA

4.1 Modified Initialization

We discuss how to initialize $\alpha_j^i$ and $b^j$ for a large value of $\lambda$ in the general case. Let $w^j = \sum_{i=1}^{n} w_i L_{yi}^j$, the sum of the weighted costs in column $j$ of $L_w$. Initialization of $\alpha^j$ and $b^j$ depends on which class has the minimal total weighted cost. Analogous to the standard case, we define $\mathcal{M} = \arg\min_j w^j$, the set of classes with the smallest total cost and denote the smallest cost by $w^M = \min w^j$. Lemma 3 regards the case when $|\mathcal{M}| = 1$ while Lemma 4 is for $|\mathcal{M}| > 1$. These two lemmas generalize Lemmas 1 and 2 for initialization in Chapter 3 to cases with unequal misclassification costs or unequal data weights. Proofs parallel those for the standard case.

Lemma 3. Suppose that there is only one column $j^*$ of the cost matrix $L_w$ with the minimal column sum. For a sufficiently large $\lambda$, $b^j = 1$ if $j = j^*$, and $-1/(k - 1)$
otherwise. \( \alpha \) minimizes

\[
\sum_{j=1}^{k} (\alpha^j - \bar{\alpha})^T K (\alpha^j - \bar{\alpha})
\]

subject to

\[
0 \leq \alpha_i^j \leq w_i L_{y_i}^j \quad \text{for } j \neq j^* \text{ and } i \notin \mathcal{I}_j, \quad (4.1)
\]
\[
\alpha_i^{j^*} = w_i L_{y_i}^{j^*} \quad \text{for } i \notin \mathcal{I}_{j^*}, \quad (4.2)
\]
\[
\sum_i \alpha_i^j = w^j \quad \text{for } j \neq j^*, \quad (4.3)
\]

where \( \mathcal{I}_j \) denotes the index set of the observations in class \( j \).

**Proof.** For a sufficiently large \( \lambda \), the minimizer of the objective function of SVM (2.5) is a constant vector \( (b^1, \ldots, b^k) \) since \( c_i^j = 0 \). Then (2.5) reduces to \( \sum_{j=1}^{k} w^j \{b^j + 1/(k - 1)\} \) up to a multiplicative constant. If any \( b_j^j \) is smaller than \(-1/(k - 1)\), then the objective function can be made smaller. So, it is sufficient to find \( (b^1, \ldots, b^k) \) minimizing \( \sum_{j=1}^{k} w^j \{b^j + 1/(k - 1)\} = \sum_{j \neq j^*} (w^j - w^{j^*}) b_j^i + 1/(k - 1) \sum_{j=1}^{k} w^j \) for \( b_j^i \geq 1/(k - 1) \). The non negatively weighted sum of \( b_j^i \)’s for \( j \neq j^* \) is minimized when \( b_j^i = -1/(k - 1) \) for \( j \neq j^* \), and consequently \( b_j^{j^*} = 1 \) by the sum-to-zero constraint. For the rest of the proof, observe that \( \xi_i^{j^*} = (b_i^{j^*} - y_i^{j^*})_+ = k/(k - 1) \) for \( i \notin \mathcal{I}_{j^*} \), and thus \( \alpha_i^{j^*} = w_i L_{y_i}^{j^*} \) to satisfy the KKT condition (2.20). Then the constraint (2.18) is restated as \( \sum_i \alpha_i^j = w^j \) for all \( j \), and consequently \( \sum_{j=1}^{k} (\alpha^j)^T y^j = -1/(k - 1) \sum_{j=1}^{k} \sum_i \alpha_i^j = -k/(k - 1) w^j \) becomes constant. \( \square \)

**Lemma 4.** Suppose that there are more than one class in \( \mathcal{M} = \arg\min w^j \). For a sufficiently large \( \lambda \), \( b_j^i = -1/(k - 1) \) for \( j \notin \mathcal{M} \), and \( b_j^i \geq -1/(k - 1) \) for \( j \in \mathcal{M} \) with
\[ \sum_{j \in \mathcal{M}} b^j = (k - |\mathcal{M}|)/(k - 1). \]  \( \alpha \) minimizes

\[ \sum_{j=1}^{k} (\alpha^j - \bar{\alpha})^T K (\alpha^j - \bar{\alpha}) \]

subject to \[ 0 \leq \alpha^j_i \leq w_i L^j_{y_i} \quad \text{for } j \notin \mathcal{M} \text{ and } i \notin \mathcal{I}_j \]
\[ \alpha^j_i = w_i L^j_{y_i} \quad \text{for } j \in \mathcal{M} \text{ and } i \notin \mathcal{I}_j \]  \( (4.4) \)
\[ \sum_i \alpha^j_i = w^M \quad \text{for } j \notin \mathcal{M}. \]  \( (4.5) \)

\[ \sum_{j \in \mathcal{M}} b^j = (k - |\mathcal{M}|)/(k - 1). \]  \( \alpha \) minimizes

\[ \sum_{j=1}^{k} (\alpha^j - \bar{\alpha})^T K (\alpha^j - \bar{\alpha}) \]

subject to \[ 0 \leq \alpha^j_i \leq w_i L^j_{y_i} \quad \text{for } j \notin \mathcal{M} \text{ and } i \notin \mathcal{I}_j \]
\[ \alpha^j_i = w_i L^j_{y_i} \quad \text{for } j \in \mathcal{M} \text{ and } i \notin \mathcal{I}_j \]  \( (4.4) \)
\[ \sum_i \alpha^j_i = w^M \quad \text{for } j \notin \mathcal{M}. \]  \( (4.5) \)

**Proof.** By the same arguments used in the proof of Lemma 3, we search \((b^1, \ldots, b^k)\) that minimizes \( \sum_{j \notin \mathcal{M}} (w^j - w^j\star)b^j + 1/(k - 1) \sum_{j=1}^{k} w^j \) with \( b^j \geq -1/(k - 1) \). Hence, \( b^j = -1/(k - 1) \) for \( j \notin \mathcal{M} \) and the remaining \( b^j \)’s have only to satisfy \( \sum_{j \in \mathcal{M}} b^j = (k - |\mathcal{M}|)/(k - 1) \) by the sum-to-zero constraint. From this equality constraint on \( b^j \)’s with \( j \in \mathcal{M} \), we infer that there is at least one \( j^\star \in \mathcal{M} \) with \( b^{j^\star} > -1/(k - 1) \). Then the KKT condition \((2.20)\) again implies that \( \alpha^{j^\star}_i = w_i L^{j^\star}_{y_i} \) for \( i \notin \mathcal{I}_{j^\star} \) since \( \xi^{j^\star}_i = (b^{j^\star} - y^{j^\star}_i)_+ > 0 \). However, by \((2.18)\) \( \sum_i \alpha^j_i \) should be the same for all \( j \). This implies that \( \sum_i \alpha^j_i = w^{j^\star} = w^M \), and moreover \( \alpha^j_i = w_i L^j_{y_i} \) for other \( j \in \mathcal{M} \) and \( i \notin \mathcal{I}_j \) to have the same sum of \( w^M \). Note that \( w^M \) is the maximum possible sum of the Lagrange multipliers for \( j \in \mathcal{M} \). This proves \((4.4)\) and \((4.5)\). \( \square \)

**Remark 3.** Lemma 3 is a special case of Lemma 4. For computational ease, each \( b^j \) with \( j \in \mathcal{M} \) can be chosen to be \((k/|\mathcal{M}| - 1)/(k - 1)\).

### 4.2 Modified Path Finding Algorithm

We describe necessary modifications of the path finding algorithm in the standard case for weighted data.

The starting values of \( b^j \) and \( \lambda \) different from the limit solutions in the lemmas are determined by setting \( k \) equations as follows. As \( \lambda \) decreases, some \( \alpha^j_i = w_i L^j_{y_i} \)
with \( j \in \mathcal{M} \) is bound to decrease, and the corresponding data index enters the elbow set \( \mathcal{E}_w \). By the constraint (2.18) this change will reduce some \( \alpha^j_i \) for \( j \notin \mathcal{M} \) as well. However, as in the standard case, \( \alpha^j_i \) for \((i, j) \in \mathcal{E}_w\) stays the same until other components \( j \) without an index in \( \mathcal{E}_w \) have a point entering the margin. For the choice of a data index in the elbow set, we define \( i^*_j = l \) if \( j \) has an element \( l \) such that \((l, j) \in \mathcal{E}_w\), otherwise \( i^*_j = \arg\min_{i \in \mathcal{B}^j} H^j_i \), where \( \mathcal{B}^j = \{ i | \alpha^j_i = w_jL_{yi} \} \) and \( H^j_i = -\sum_{r=1}^{n}(\alpha^j_r - \bar{\alpha}_r)K(x_r, x_i) \). Then \( f^j(x_{i^*_j}) = -1/(k - 1) \) for \( j = 1, \ldots, k \) gives a set of \( k \) equations, and by solving for \( b^j \) and \( \lambda \), we have

\[
\lambda = \frac{k - 1}{kn} \sum_{j=1}^{k} \frac{1}{\sum_{r=1}^{n}(\alpha^j_r - \bar{\alpha}_r)K(x_r, x_{i^*_j})} \quad \text{and} \quad \quad (4.6)
\]

\[
b^j = -\frac{1}{k - 1} + \frac{1}{n\lambda} \sum_{r=1}^{n}(\alpha^j_r - \bar{\alpha}_r)K(x_r, x_{i^*_j}). \quad \quad (4.7)
\]

The piecewise linearity of the coefficient path remains true for the general case of unequal misclassification costs or different data weights, and hence the SVM path algorithm is still valid with the new upper bounds \( w_jL_{yi} \) on the Lagrange multipliers. Other details discussed in Chapter 3 for the standard case remain true for the generalized case as well.
CHAPTER 5

COPING WITH LARGE DATA SETS

5.1 Motivation

The major computational load of the path finding algorithm lies in solving the system of linear equations with \(|E^\ell| + k\) unknown variables. Its computational complexity is usually in the order of \(O((|E^\ell| + k)^3)\). When the data set is large and a flexible kernel is applied to allow for nonlinear boundaries, the elbow set can grow big to significantly slow down computation. Also, large data sets often lead to singularity of the kernel matrix \(K\) when data points are densely distributed, and this near singularity can hamper progression of the solution path. The dimension of the covariates can also add to the computational complexity. The elbow set size tends to become increasingly large when the input dimension is high even for a linear kernel. The effect of the dimension of the covariates on the elbow size will be illustrated in Section 7.1.

From a statistical standpoint of function estimation, the expression of the exact solutions in (2.2) and (2.4) suggest how they can be approximated efficiently in the presence of large number of data points. Notice that the \(n\) representers in \(\{K(x_i, \cdot), i = 1, \ldots, n\}\) (called a dictionary) are basis functions used to minimize
the penalized empirical risks in (2.1) and (2.3). When \( x_i \)'s are dense, the dictionary of the corresponding basis functions \( K(x_i, x) \) tends to be overcomplete with almost redundant elements. Thus, subsetting the basis functions would not degrade the accuracy of approximation much while it would lessen the computation substantially. The main issue is how to judiciously select a subset of basis functions from the overcomplete dictionary in a fast and inexpensive manner. Since each basis function corresponds to a data point, thinning data-dependent basis functions essentially amounts to finding \( I^* \subset \{1, \ldots, n\} \) with \( |I^*| \ll n \) such that \( \text{span}\{K(x_{i_\ell}, \cdot), i_\ell \in I^*\} \approx \text{span}\{K(x_i, \cdot), i = 1, \ldots, n\} \). This subset sampling is related to low-rank approximation to the kernel matrix or Gram matrix in the machine learning literature (Lee and Mangasarian 2001; Williams and Seeger 2001; Smola and Schölkopf 2000; Fine and Scheinberg 2002) and basis thinning in the smoothing literature (Xiang 1996). To determine the subset \( I^* \), Lee and Mangasarian (2001) and Williams and Seeger (2001) used random subsampling without replacement and Xiang (1996) used clustering while Smola and Schölkopf (2000) considered more explicit optimization and orthogonalization procedure for sparse greedy matrix approximation.

5.2 Subset Sampling Schemes

We detail a few schemes to specify \( I^* \) of size \( m \). We consider more systematic data reduction schemes than simple random sampling in order to minimize redundancy in the basis functions and to prevent numerical singularity due to closeness of a pair of data points. Cluster means from a \( k \)-means cluster analysis of the data with sufficiently large \( k \) naturally define a set of non-redundant basis functions. Note that the cluster means \( \{\bar{x}_{i_\ell}, \ell = 1, \ldots, m\} \) are not a subset of the original \( x_i \)'s, and this
data reduction scheme is also known as vector quantization in lossy data compression. For classification, the cluster analysis is to be done class by class so as to keep the class proportion in the reduced data the same as the original data. For out-of-sample prediction which will facilitate tuning, one may take an actual sample point from each cluster instead of the cluster mean and use the remaining for validation.

With less expensive computation than clustering, statistically equivalent blocks (Wilks 1962) in the sampling theory of order statistics may be used to generate a subsample well preserving the data distribution. Statistically equivalent blocks are the strata of equal coverage probabilities formed by quantiles of the projected data using some real-valued ordering functions. By choosing ordering functions that reflect major variances in the data such as the first few principal components, we can partition the sample space into relatively compact blocks. Then random subsampling from each block yields a representative subset with much reduced sampling variability analogous to stratified sampling. Figure 5.1 illustrates partitions of a simulated data set of size 200 into 40 blocks by \(k\)-means algorithm (left) and statistically equivalent blocks (right) using the first two principal components as the ordering functions.

5.3 Reformulation of Regularization Problems

Consider the following regularization problem with a loss \(L\) in an RKHS in general

\[
\min_{f \in \{1\} \oplus H_K} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(x_i)) + \lambda \|h\|_{H_K}^2, \tag{5.1}
\]

which subsumes the optimization for SVM in (2.1) and (2.3). Defining \(H_K^* = \text{span}\{K(x_{ij}, \cdot), i_j \in I^*\}\) for the reduced data set of size \(m\), now we seek \(f \in \{1\} \oplus H_K^*\) for (5.1). Depending on the loss \(\mathcal{L}\), this new optimization problem can be much smaller than the original problem over the full data. Regression with the squared
error loss is the case, for example. For the hinge loss, however, subsetting basis functions does not imply immediate reduction in computation due to the fact that the size of the dual problem to solve remains to be the same as the original problem. For further reformulation of (5.1), consider a reduced data set with weights, namely, $D_w = \{(x_{i\ell}, y_{i\ell}) \text{ with weight } w_{i\ell}, \ell = 1, \ldots, m \text{ and } i_{\ell} \in I^*\}$. $w_{i\ell}$ can be the size of the $\ell$th cluster for clustering or a constant for statistically equivalent blocks. We note that construction of such a reduced data set is closely related to data squashing (DuMouchel et al. 2000). It aims at obtaining pseudo data with weights matching the moments of the full data. In the data squashing process, the original data are first grouped into regions based on the values of the categorical variables and then further binned into some preferred subregions. Then pseudo points and weights are generated to make the likelihood function of the pseudo data as close as possible to

Figure 5.1: (a) Voronoi Tessellation of the centers of 40 clusters formed by $k$-means algorithm and (b) statistically equivalent blocks ($40 = 10 \times 4$) obtained by using principal components as the ordering functions.
the likelihood function of the original data based on square error loss. Although data squashing is a useful data compression procedure in its own right, it does not appear to be the best choice for reduced computation of non-likelihood based methods such as the SVM.

Now (5.1) is transformed into an optimization problem over the reduced data set with weights

$$\min_{f \in \{1\} \oplus H_{K}} \frac{1}{m} \sum_{i=1}^{m} w_i \mathcal{L}(y_i, f(x_i)) + \lambda \|h\|_{H_K}^2, \quad (5.2)$$

which involves only $m + 1$ unknowns. This reformulation will be adopted to deal with large data sets for the SVM, in particular, using the path-finding algorithm extended for data weights in Chapter 4. The approximate solution path is the solution path based on the subsample of the entire dataset.

Section 7.1 compares the performance of different subset sampling schemes in terms of accuracy and efficiency, providing empirical evidence for its feasibility. It also investigates the effect of Bayes error rate, dimensionality and data reduction factor on the approximate solution paths.
CHAPTER 6

PRACTICAL ISSUES IN IMPLEMENTATION

The proposed algorithm for approximate solution paths has been implemented in msvmpath (Cui 2007). The msvmpath is an R package which generates the complete regularized solution path of the SVM extending svmpath for the binary SVM. Some practical issues for its implementation are discussed briefly.

6.1 Stopping Rule

In Section 3.3, several stopping rules are discussed. As one of the rules, we may stop the path-finding algorithm when the upper sets become empty since it means that the empirical risk \( R_n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(x_i)) \) has attained 0. Once the empirical risk reaches zero, or it is near 0, there is no need for further progression of the solution path by decreasing \( \lambda \) because this would only increase the complexity of the solution. The following lemma provides justification for this statement, which says that the empirical risk is non-decreasing in \( \lambda \) and the complexity of the solution is non-increasing in \( \lambda \). In fact, it is a general property of regularization methods of the form, \( \min_f R_n(f) + \lambda J(f) \) and the proof is included here for self-containedness.

**Lemma 5.** Suppose that \( \hat{f}_\lambda \in \mathcal{H} \) minimizes \( R_n(f) + \lambda J(f) \) for a given \( \lambda \). Then, for \( \lambda_{k+1} < \lambda_k \), \( R_n(\hat{f}_{\lambda_{k+1}}) \leq R_n(\hat{f}_{\lambda_k}) \) and \( J(\hat{f}_{\lambda_{k+1}}) \geq J(\hat{f}_{\lambda_k}) \).
Proof. First, note that \( \hat{f}_\lambda \) is equivalently defined as the function minimizing \( R_n(f) \) subject to \( J(f) \leq C(\lambda) \), where \( C(\lambda) \) is a monotonically decreasing function of \( \lambda \). For every \( \lambda \), consider a set of candidate functions \( \mathcal{F}_\lambda = \{ f \in \mathcal{H} | J(f) \leq C(\lambda) \} \) for the solution. Since \( C(\lambda_{k+1}) > C(\lambda_k) \) for \( \lambda_{k+1} < \lambda_k \), \( \mathcal{F}_{\lambda_{k+1}} \) contains \( \mathcal{F}_{\lambda_k} \). In other words, as the regularization parameter decreases, the class of candidate functions becomes richer. Since \( f \) can be chosen among more candidates at \( \lambda_{k+1} \) than at \( \lambda_k \) we have:

\[
R_n(\hat{f}_{\lambda_{k+1}}) \leq R_n(\hat{f}_{\lambda_k}).
\]

Now, we will show \( J(\hat{f}_{\lambda_{k+1}}) \geq J(\hat{f}_{\lambda_k}) \). Suppose that \( J(\hat{f}_{\lambda_{k+1}}) < J(\hat{f}_{\lambda_k}) \). Then:

\[
R_n(\hat{f}_{\lambda_k}) + \lambda_k J(\hat{f}_{\lambda_k}) > R_n(\hat{f}_{\lambda_{k+1}}) + \lambda_k J(\hat{f}_{\lambda_{k+1}}) \geq R_n(\hat{f}_{\lambda_{k+1}}) + \lambda_k J(\hat{f}_{\lambda_{k+1}}).
\]

The second inequality follows from the above result that \( R_n(\hat{f}_{\lambda_{k+1}}) \leq R_n(\hat{f}_{\lambda_k}) \). However, the inequality \( R(\hat{f}_{\lambda_k}) + \lambda_k J(\hat{f}_{\lambda_k}) > R(\hat{f}_{\lambda_{k+1}}) + \lambda_k J(\hat{f}_{\lambda_{k+1}}) \) contradicts that \( \hat{f}_{\lambda_k} \) is the minimizer of \( R_n(f) + \lambda J(f) \). Therefore, \( J(\hat{f}_{\lambda_{k+1}}) \geq J(\hat{f}_{\lambda_k}) \) for \( \lambda_{k+1} < \lambda_k \). \( \square \)

In practice, finding the solution at an optimal value of \( \lambda \) is of main interest and the solution path does not need to be completed. Locating such an optimal value requires reasonably accurate assessment of error rates. Either setting aside a portion of the training data set or cross validation can be done for tuning. An added benefit of basis thinning is that we can use the subset of selected points for training and the rest for tuning as an “out-of-bag sample” as in bootstrap. Hence fitting and tuning can be done simultaneously and the tuning error can serve as an indicator for early stopping. Due to the choice of the points in the subset and the incorporation of the weights, this would induce a bias in the tuning error to some extent. Nonetheless, it is expected to be useful for implementation when combined with a liberal stopping criterion. The path may be terminated when the tuning error soars from a previous smallest value to a certain sub-optimal value, say, the average of the initial tuning
error and the smallest tuning error up to the current step after taking a sufficient
number of steps from the minimum.

6.2 Choice of Kernel and Its Parameter

There exists a trade-off between precision and computational efficiency in the
choice of a kernel. The default kernel for svmpath (Hastie 2004) and msvmpath (Cui
2007) in R is linear. The elbow size for the linear kernel tends to be small allowing for
a quick completion of the solution path unless the input dimension is high. However,
the linear kernel is often limited, and nonlinear kernels such as the radial kernel
generally yield lower test error rates than the linear kernel at the expense of increased
computational load. The size of the elbow sets for the radial kernel can grow as
large as $n(k - 1)$ depending on the kernel parameter, which may significantly hinder
the progress of the path. Hence, the parameter $\gamma$ in the radial kernel $K(x, x') =
\exp(-\gamma(x - x')^2)$ needs to be carefully specified though the tuning parameter $\lambda$
can compensate for its deviation from a proper range to some extent.

Proper choice of $\gamma$ depends on the distribution of inter-points distances. A con-
ceptual guidance for the choice of $\gamma$ can be stated that it has to be small enough to
yield underfitting at the initial steps of the path yet large enough to ensure overfit-
ting at the later steps. One may adjust $\gamma$ manually for this desired tuning pattern
by decreasing or increasing it gradually from an initial value. For general use, Brown
et al. (2000) suggest a rule of thumb to set $\gamma = 1/2\sigma^2$ where $\sigma$ is the median of the
distances between each point and its nearest neighbor in other classes. Alternatively,
the value of $\gamma$ can be chosen to maximize the between-class sum of squares (BSS) or
the numerator of a kernelized F-ratio. Let \( x_{ji} \) denote the \( i \)th observation from class \( j \) for \( j = 1, \ldots, k \) and \( i = 1, \ldots, n_j \), where \( n_j \) is the sample size of class \( j \).

From the analysis of variance, the between-class sum of squares is

\[
\sum_{j=1}^{k} n_j (\bar{x}_j - \bar{x})^2 = \sum_{j=1}^{k} \frac{1}{n_j} \left( \sum_{i=1}^{n_j} x_{ji} \right)^2 - \frac{1}{n} \left( \sum_{j=1}^{k} \sum_{i=1}^{n_j} x_{ji} \right)^2.
\]

Its kernelized version via a positive definite kernel \( K \) is then given by

\[
BSS = \sum_{j=1}^{k} \frac{1}{n_j} \sum_{i_1=1}^{n_j} \sum_{i_2=1}^{n_j} K(x_{ji_1}, x_{ji_2}) - \frac{1}{n} \sum_{j_1=1}^{k} \sum_{j_2=1}^{k} \sum_{i_1=1}^{n_{j_1}} \sum_{i_2=1}^{n_{j_2}} K(x_{j_1i_1}, x_{j_2i_2}). \tag{6.1}
\]

For the radial kernel, (6.1) can be simplified to

\[
\sum_{j=1}^{k} \left( \frac{1}{n_j} - \frac{1}{n} \right) \sum_{i_1 \neq i_2}^{n_j} K(x_{ji_1}, x_{ji_2}) - \frac{1}{n} \sum_{j_1 \neq j_2}^{k} \sum_{i_1=1}^{n_{j_1}} \sum_{i_2=1}^{n_{j_2}} K(x_{j_1i_1}, x_{j_2i_2}) + (k - 1).
\]

Ahn (2006) proposes a similar criterion \( D \) for the binary case, defined as the difference between the within-class similarity and the between-class similarity. We propose its extension to the multi-class case to be

\[
D = \sum_{j=1}^{k} \frac{1}{n_j(n_j - 1)} \sum_{i_1 \neq i_2}^{n_j} K(x_{ji_1}, x_{ji_2}) - \frac{1}{k - 1} \sum_{j_1 \neq j_2}^{k} \frac{1}{n_{j_1}n_{j_2}} \sum_{i_1=1}^{n_{j_1}} \sum_{i_2=1}^{n_{j_2}} K(x_{j_1i_1}, x_{j_2i_2}). \tag{6.2}
\]

Both of the criteria aim at the difference between the intra-class similarity and the inter-class similarity, however with different weights.

For comparison, the values of \( \gamma \) maximizing the criteria were computed over one simulated data set and five benchmark data sets. Figure 6.1 shows the \( BSS \) and \( D \) curves as a function of \( \log_2 \gamma \) for the three-class simulated example in Section 7.1. They behave very similarly with unique modes roughly around \( 2^{-3} \). Despite the heuristic nature of the criteria, the main advantage is that given training data, finding the modes of \( BSS \) and \( D \) can be automated easily by numerical optimization routines.
To measure the efficiency of those heuristic criteria in contrast to performance-based tuning, we also searched the optimal $\gamma$ minimizing either test error rates or cross validated error rates over a grid of equally spaced values on the logarithmic scale. Table 6.2 displays the optimal $\gamma$ values over the six data sets. Note that the optimal values chosen by $BSS$ and $D$ are very close to each other but somewhat different from those by grid search. For description of the data sets and details on how the error rates were computed, see Section 7.2. With the optimal values by grid search serving as a golden standard, the efficiency of $BSS$ in choosing $\gamma$ is defined as the ratio of the error rate at the $\gamma$ chosen by $BSS$ and the least error rate obtained from grid search. Table 6.2 shows the difference in the error rates at the $\gamma$ values chosen by $BSS$ and grid search. For a half of the data sets, the efficiency is more than 0.94, and for the rest it ranges from 0.72 to 0.77. This comparison suggests that the $BSS$ criterion can be as effective as the full-scale tuning for some cases and at least reasonable in providing a starting value of $\gamma$ for other cases.

### Table 6.1: Optimal $\gamma$ values determined by the criteria of $BSS$ and $D$ and grid search

<table>
<thead>
<tr>
<th>Data set</th>
<th>$BSS$</th>
<th>$D$</th>
<th>Grid Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>simulation</td>
<td>0.1438</td>
<td>0.1400</td>
<td>0.2500</td>
</tr>
<tr>
<td>glass</td>
<td>0.1190</td>
<td>0.1155</td>
<td>0.0625</td>
</tr>
<tr>
<td>waveform</td>
<td>0.0270</td>
<td>0.0264</td>
<td>0.0156</td>
</tr>
<tr>
<td>satimage</td>
<td>0.0239</td>
<td>0.0210</td>
<td>0.0625</td>
</tr>
<tr>
<td>vowel</td>
<td>0.1681</td>
<td>0.1601</td>
<td>1</td>
</tr>
<tr>
<td>zip code</td>
<td>0.0058</td>
<td>0.0070</td>
<td>0.0156</td>
</tr>
</tbody>
</table>

41
Figure 6.1: Heuristic criteria for selection of $\gamma$ in the radial kernel evaluated over the three-class simulated example; (a) $BSS$ and (b) $D$.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Grid Search</th>
<th>$BSS$</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>simulation</td>
<td>0.1880</td>
<td>0.2420</td>
<td>0.7769</td>
</tr>
<tr>
<td>glass</td>
<td>0.3105</td>
<td>0.3179</td>
<td>0.9766</td>
</tr>
<tr>
<td>waveform</td>
<td>0.1543</td>
<td>0.1632</td>
<td>0.9452</td>
</tr>
<tr>
<td>satimage</td>
<td>0.1225</td>
<td>0.1685</td>
<td>0.7270</td>
</tr>
<tr>
<td>vowel</td>
<td>0.3528</td>
<td>0.4589</td>
<td>0.7689</td>
</tr>
<tr>
<td>zip code</td>
<td>0.0837</td>
<td>0.0877</td>
<td>0.9446</td>
</tr>
</tbody>
</table>

Table 6.2: Test error rates or cross validated error rates at the optimal $\gamma$ chosen by grid search and $BSS$ and the efficiency of the $BSS$ criterion relative to grid search.
CHAPTER 7

NUMERICAL STUDIES

7.1 Simulation Study

7.1.1 Comparison of Subset Sampling Schemes

A simulation was carried out to compare different data reduction schemes described in Section 5.2 in terms of their prediction accuracy and sampling variability. The simulation setting in Section 3.4 was used with the sample size of 360 and 120 cases in each class, and 100 replicates of such a training set were generated. Random sampling, two variants of clustering and statistically equivalent blocks by using the covariates were compared. The reduction factor defined as the ratio of the sample size to the subset size was 6, that is, using 20 cases from each class in this example. In the first variant of clustering (Cluster 1), the closest point to each cluster mean was chosen while a random data point was chosen in the second variant (Cluster 2). The radial kernel was used with the fixed value of $\gamma = 1$ for the full data set and $\gamma = 0.3$ for the reduced data sets in order to control other extraneous factors which may affect the comparison. These values were numerically characterized to minimize the overall error rates under the simulation setting, taking different sample sizes into account. The value of $\lambda$ was chosen to minimize the tuning error rate evaluated over...
the “out-of-sample” of size 300. For evaluation of the performance of approximate SVM solutions, a test data set of size 3,600 with 1,200 in each class was generated.

Table 7.1 shows the mean and standard deviation of the test error rates of the approximate SVM solutions for the three data reduction schemes. Clustering gave lower test error rates than other schemes on average. Data reduction via statistically equivalent blocks seemed to work as well with a small standard deviation in the error rates. Random sampling resulted in inflated test error rates with a large standard deviation as expected. The last two columns in Table 7.1 are for the exact solutions when tuned by five-fold cross validation or by a separate tuning data set of the same size as the training data, and they serve as the golden standard against which approximate solutions are compared. Exact solutions were slightly better than the approximate solutions based only on one sixth of the data in the mean and standard deviation. For illustration of the computational efficiency gained by subset sampling, some statistics of solution paths were obtained. Table 7.2 shows the mean and standard deviation of the average elbow sizes and the path lengths of 100 replicates, which are the two major factors that determine the efficiency of the path finding algorithm. On average, subset sampling gave roughly 1.5 fold reduction in the elbow set size and more than four fold reduction in the path length compared with using the full data. In terms of the statistics of approximate solution paths, the three data reduction schemes were comparable. This subset sampling led to at least three fold reduction in the computing time, which includes preprocessing of the data and generation of the solution paths, without compromising the prediction accuracy. The computation was done on a linux machine with 2.1 GHz AMD Opteron CPU.
Table 7.1: Test error rates of the approximate SVM solutions with different data reduction schemes.

<table>
<thead>
<tr>
<th></th>
<th>RS</th>
<th>SEB</th>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>5-fold CV</th>
<th>Full Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.1963</td>
<td>0.1857</td>
<td>0.1833</td>
<td>0.1843</td>
<td>0.1820</td>
<td>0.1847</td>
</tr>
<tr>
<td>SD</td>
<td>0.0159</td>
<td>0.0089</td>
<td>0.0070</td>
<td>0.0086</td>
<td>0.0075</td>
<td>0.0086</td>
</tr>
</tbody>
</table>

NOTE: RS and SEB stand for random sampling and statistically equivalent blocks.

Table 7.2: Statistics of solution paths, average elbow set sizes, path lengths and computing time.

<table>
<thead>
<tr>
<th></th>
<th>RS</th>
<th>SEB</th>
<th>Cluster1</th>
<th>Cluster2</th>
<th>Full Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elbow size Mean</td>
<td>15.88</td>
<td>17.39</td>
<td>18.07</td>
<td>17.61</td>
<td>25.08</td>
</tr>
<tr>
<td>SD</td>
<td>3.49</td>
<td>3.43</td>
<td>2.84</td>
<td>3.08</td>
<td>14.37</td>
</tr>
<tr>
<td>Path length Mean</td>
<td>158.22</td>
<td>162.12</td>
<td>164.90</td>
<td>158.73</td>
<td>665.31</td>
</tr>
<tr>
<td>SD</td>
<td>42.53</td>
<td>36.94</td>
<td>34.68</td>
<td>39.71</td>
<td>484.51</td>
</tr>
<tr>
<td>Time(sec) Min</td>
<td>0.65</td>
<td>0.76</td>
<td>1.07</td>
<td>0.87</td>
<td>3.18</td>
</tr>
</tbody>
</table>

NOTE: RS and SEB stand for random sampling and statistically equivalent blocks.

7.1.2 Bayes Error, Dimensionality and Data Reduction Factor

For the scale-down approach by subset sampling, we need to decide the fraction of the data to be used for finding an approximate solution with nearly the same accuracy as the exact solution for the full dataset. Obviously the smaller the data fraction, the less computation is required. However, using too small a fraction would lead to an increase in the test error rate. Therefore, balance between computational efficiency and accuracy is necessary. In general, the quality of approximate solutions depends on various aspects of a given problem, which will determine the appropriate range of the data reduction factor. The following simulation investigates how the
dimensionality and intrinsic difficulty of the problem measured by the Bayes error rate affect error rates and computing times when combined with various levels of the data reduction factor.

For simplicity, consider a binary classification problem with two multivariate normal distributions as class conditional distributions and equal proportions. Under this circumstance, the Bayes error rate is given by $\Phi(-\delta/2)$ as a function of the Mahalanobis distance between the two distributions

$$\delta = \left\{ (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) \right\}^{1/2},$$

where $\mu_1$ and $\mu_2$ are the means and $\Sigma$ is the common covariance matrix. It is postulated that when the Bayes error rate is high or the input dimension is high, admissible levels of the reduction factor for attaining a desired accuracy would be very low due to the increasing difficulty of the problem. In the simulation, covering almost separable to inseparable problems, three levels of the Mahalanobis distance ($\delta = 1$, 3, and 5) were considered with the resulting Bayes error rates of 0.00621 (low), 0.0668 (intermediate) and 0.309 (high), respectively. For the dimension of the covariates $p$, 10 (low), 100 (intermediate), and 500 (high) were considered. Given each combination of $p$ and $\delta$, 500 data points with independent covariates for each class were generated. The fractions for data reduction varied from 10% to 100% with an increment of 10%. The first variant of clustering was used for approximate solution paths with the linear kernel. A separate tuning set of the same size was employed to examine the effect of the data reduction factor (RF) in comparison with the full data solution. A test set of size 5,000 for each class was simulated for evaluation of the error rate. Each combination of the three factors had ten replicates.

Figure 7.1 graphically summarizes the results. The solid lines denote the mean test error rate of the ten replicates as a function of the data fraction ($1/\text{RF}$) while the
dotted lines denote the standard deviation around the mean. The three panels are for low, intermediate and high Bayes error rates. In each scenario, as the dimension gets higher, the larger test error rates occurred requiring the reduction factor to be close to 1 for the minimum error rate. This trend is increasingly so as the Bayes rate increases. The convergence of the test error rates to the Bayes error rate was much slower for higher dimensions as the data fraction increases. As a result, to achieve a low test error rate, a small reduction factor would be recommended when either the dimension of the covariates or the Bayes error rate is high. On the contrary, when \( p = 10 \), data can be reduced significantly virtually with no loss in the prediction accuracy over a wide range of problems. For \( p = 100 \), the reduction factor can be still as large as 50\% for problems with low to intermediate Bayes error rates. However, when \( p = 500 \), data reduction would be feasible only for nearly separable problems.

Aside from the accuracy, computational efficiency is of practical importance. Figure 7.2 shows some statistics of solution paths, namely average elbow size, path length for the intermediate Bayes error rate case as the reduction factor varies. Similar results were obtained for the low Bayes error case. Note that elbow set sizes grow rapidly as the data fraction increases for the high dimensional case and the path lengths are almost linear in the data fraction.

7.2 Benchmark Study

The SVM path algorithm was applied to five benchmark data sets for classification so as to compare its performance with some top classification approaches.
Figure 7.1: Test error rates as a function of the data fraction (1/RF) for the two-class simulated example under three scenarios of (a) low (b) intermediate and (c) high Bayes error rate. The solid lines indicate the mean of ten replicates, and the dotted lines indicate the standard deviation around the mean.


Table 7.2 describes the data sets in terms of the size of training and test data, the number of covariates, and the number of classes. For the small to medium sized glass, waveform and, vowel data, five-fold CV was used to tune $\lambda$ while for satimage and zipcode data, 800 and 600 data points were used for training, respectively, and the rest for tuning since the training sets are huge. The data reduction approach for satimage and zipcode is cluster analysis with weights.

For the evaluation of performance, five-fold cross validated error rate is reported for glass. For the rest of the data sets, error rates over separate test sets are reported.
Figure 7.2: Statistics of solution paths as a function of the data fraction (1/RF): (a) average elbow size and (b) path length for the intermediate Bayes error case.

The radial kernel was used with the best $\gamma$ values chosen by grid search from Table 6.2. Note that there is potential downward bias in the evaluation as $\gamma$ was set to minimize test error rates.

Table 7.2 shows the results (either CV or test error rates). For comparison, the corresponding results of Adaboost algorithm from Friedman et al. (2004) are included. Boosting algorithms, in general, are known to have produced very competitive results in many applications. For **vowel** and **waveform** data, the MSVM refined with the solution path algorithm shows improved performance over boosting. Especially, the extent of improvement is significant for **vowel** data. But for **glass**, it fails to parallel the performance of boosting. For **satimage** and **zipcode**, the higher test error rates might have resulted from the high data reduction rate.
<table>
<thead>
<tr>
<th>Data set</th>
<th>Training set size</th>
<th>Test set size</th>
<th># of Inputs</th>
<th># of Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>glass</td>
<td>214</td>
<td>NA</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>waveform</td>
<td>300</td>
<td>5000</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>vowel</td>
<td>528</td>
<td>462</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>satimage</td>
<td>4435</td>
<td>2000</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>zipcode</td>
<td>7291</td>
<td>2007</td>
<td>256</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 7.3: Description of data sets

<table>
<thead>
<tr>
<th>Methods</th>
<th>glass</th>
<th>waveform</th>
<th>vowel</th>
<th>satimage</th>
<th>zipcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSVM Path</td>
<td>0.310</td>
<td>0.154</td>
<td>0.353</td>
<td>0.123</td>
<td>0.084</td>
</tr>
<tr>
<td>Real AdaBoost</td>
<td>0.234</td>
<td>0.182</td>
<td>0.496</td>
<td>0.091</td>
<td>0.025*</td>
</tr>
</tbody>
</table>

NOTE: The error rate of * was obtained by neural network (Hastie et al. 2001).

Table 7.4: Error rates of the MSVM assisted with the solution path algorithm

To have further insights into the structure of zipcode digit recognition problem, we considered subproblems nested within this ten-class problem. To identify natural subproblems, we first clustered the pixel-wise median vector of the digits in the entire training data (7291 observations) and found two major clusters of (3,5,6) and (4,7,8,9) as shown in Figure 7.2. Using the mean vectors gave the same clustering result. These subproblems have 1878 and 2483 cases in the training data set and 496 and 690 in the test set for the clusters of (3,5,6) and (4,7,8,9), respectively. For each subproblem, we varied the proportion of training data for generating approximate MSVM solution paths and used the rest for tuning. Table 7.2 displays a summary of the test error rates. Note that because of the memory limitation in R, only up to 60% of the training data could be fit for the subproblem of (4,7,8,9). As the results suggest,
there is trade-off between the training set size and the tuning set size in that the more data are used for fitting the solution path, the less data are left for tuning the regularization parameter, potentially leading to a subpotimal choice of the final classifier due to the increased variance in the error rate evaluation. On the whole, the classification accuracy was improved over the overall accuracy of the MSVM for digit classification when the subproblems were tackled. The results suggest that it may be a better strategy solving related subproblems and combining those classifiers to form a comprehensive classifier in a hierarchical fashion for the original ten-class problem. Construction of a proper family of subproblems requires structural understanding of the given problem.

<table>
<thead>
<tr>
<th>Problem</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3,5,6)</td>
<td>0.0665</td>
<td>0.0605</td>
<td>0.0524</td>
<td>0.0645</td>
<td>0.0645</td>
<td>0.0706</td>
<td>0.0807</td>
<td>0.0665</td>
<td>0.0444</td>
</tr>
<tr>
<td>(4,7,8,9)</td>
<td>0.0507</td>
<td>0.0551</td>
<td>0.0522</td>
<td>0.0493</td>
<td>0.0507</td>
<td>0.0580</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 7.5: Test error rates of the MSVM with different fractions of the training data for two subproblems of zipcode data
Figure 7.3: The hierarchical cluster analysis of the median vectors of the ten digits in zipcode. NOTE: Cluster 10 corresponds to the digit 0.
CHAPTER 8

DISCUSSION

Generally speaking, grid search depends largely on the researcher’s experience and subjective understanding of the given data. If a wide range of the tuning parameter $\lambda$ has to be covered, then there may be a limit in the resolution of the grid as increasingly large computation needs to be carried out. It can be time consuming to pin down the search region and capture the optimal value of $\lambda$. However, with the solution path algorithm, the entire set of solutions at all possible values of $\lambda$ can be computed efficiently. The algorithm is so flexible that it can be initiated at any point along the solution path and terminated as necessary. As a result, it provides a consistent and systematic way to explore the model space completely given the data.

It is conceptually attractive to have the whole solution path in perspective, and it is computationally effective to construct the path sequentially with the basic operation of solving a system of linear equations. Another type of a regularization problem, which may well benefit from the idea, is feature selection via an $l_1$ type penalty imposed on rescaling factors of features, in addition to the squared norm penalty for the maximal geometric margin. For example, such a sparse solution approach to feature selection for the MSVM is described in Lee et al. (2004a) by using functional ANOVA decomposition. Besides $\lambda$, there is another regularization parameter $\lambda_\theta$ in
the setting that governs the sum of non-negative weights $\theta_\nu$ assigned to each feature $x_\nu$. The sum of $\theta_\nu$ is closely tied to the model complexity of the number of features present in each model. In this case, the entire spectrum of models embraces the simplest model of a constant function to the least regularized one with all the features regardless of their relevance to prediction of $y$. It is particularly enticing to overlook how the features get in and out of the current model along the path, in conjunction with the prediction accuracy of the current model. This complete picture of a model path enables us to understand the data better, and especially it helps us to find out multiple good descriptions of sparse data. We remark here that this approach is closely related to the least angle regression proposed by Efron et al. (2004) for variable selection in multiple linear regression, despite the notable difference between the two settings. We also note in passing that for the special case of an $l_1$-norm linear MSVM, Wang and Shen (2005) recently studied an algorithm to construct the solution path for simultaneous fitting and feature selection.

We have extended the SVM solution path algorithm to incorporate different data weights and used the extended algorithm to find an approximate solution path for large data sets by subset sampling. This approach for an approximate solution can improve computational efficiency greatly while retaining the advantage of the path finding algorithm such as fine tuning of the regularization parameter $\lambda$. The quality of approximate solutions depends on data partitioning schemes, reduction factor, data dimensionality and the intrinsic difficulty of the problem at hand among others. Effective representation of the data appears to be the key to data reduction. Especially for high dimensional data, data geometry will play an important role for computation. For example, proper choice of the ordering functions for statistically
equivalent blocks largely depends on the data geometry. So does an ideal metric to be used in clustering data points and binning strategy for data squashing. Recent developments in dimensionality reduction such as nonlinear principal component analysis (Schölkopf et al. 1998) and data manifold learning (Belkin et al. 2006) can guide and improve data reduction procedures significantly.

On the theoretical side, the effect of subsampling and approximating the exact solution on inference and prediction needs to be studied. For penalized approximate cubic splines, Xiang (1996) proved that to attain the same convergence rate as the penalized cubic spline in a simple setting, the size of the reduced data set has to increase at least at the rate of $O(n^{4/15}/\lambda^{1/3})$. It is hard to draw a perfect parallel to this result for the SVM because the coefficients of the SVM are not given in an explicit form in general. However, it would be useful to have such a theoretical guidance to data reduction rates.

Numerically, the path finding algorithm may be further improved by adopting rank-one update or downdate of a matrix formed at each breakpoint of $\lambda$ for inversion. This reduces the computational complexity from $O((|E_{w}| + k)^3)$ to $O((|E_{w}| + k)^2)$. However, our preliminary study showed that such a rank-one update severely suffers from propagation of rounding errors, and further investigation is called for a possible remedy.
APPENDIX A

R PACKAGE, MSVMPATH MANUAL
Package 'msvmpath'

September 17, 2007

Version 0.3-1

Date 2007-2-6

Title The solution path algorithm of multiclass support vector machines. The entire set of codes are based on Hastie's svmpath. It is an add on package to svmpath.

Author Zhenhuan Cui <zhenhuan@stat.ou.edu>, Trevor Hastie

Maintainer Zhenhuan Cui <zhenhuan@stat.ou.edu>

Depends svmpath

Description This function gives the whole set of solutions to the multiclass support vector machines at every possible value of the tuning parameter through one fit of the data.

License GPL Version 2 or newer

---

cv.msvmpath Cross Validation for msvmpath

Description

This function cross validates msvmpath for the optimal value of the tuning parameter (lambda).

Usage

cv.msvmpath(x, y, fold = 10, kernel.function = poly.kernel,
cost = cost.dft, wt.obs = wt.dft,...)
Arguments

\(x\)  
a data matrix with \(n\) observations (rows) and \(p\) covariates (columns) for training.

\(y\)  
a vector with \(\{1,2,\ldots,k\}\) valued class labels for training.

\(\text{fold}\)  
the number of folds in cross validation. Values larger than the smallest class size are not allowed except for \(n\). If \(\text{fold} = n\), then leave-one-out cross validation is conducted.

\(\text{Nlambda}\)  
the number of equally spaced lambda values generated for cross validation.

\(\text{kernel.function}\)  
the kernel function. Only \text{poly.kernel} and \text{radial.kernel} are available now.

\(\text{gpw.init}\)  
2 to the power of \(\text{gpw.init}\) is the initial value of the parameter of the radial kernel.

\(\text{param.kernel}\)  
the parameter associated with the kernel function.

\(\text{eps}\)  
a small value for tolerance in the stopping rule.

\(\text{Nmoves}\)  
the maximal number of steps for the solution path.

\(\text{digits}\)  
the number of digits displayed.

\(\text{lambda.min}\)  
the smallest value of lambda for stopping the path.

\(\text{cost}\)  
a \(k\) by \(k\) cost matrix with \(ij\)th entry the cost of misclassifying class \(i\) as class \(j\). The default value is a \(k\) by \(k\) matrix with 0's along the diagonal and 1's elsewhere.

\(\text{wt.obs}\)  
a vector of length \(n\) with data weights. The default is a vector of ones for equal weights.

\(\text{lag}\)  
the number of steps after which the early stopping rule comes into play.

Details

This function finds the solution path for the given data set first to decide a proper range of lambda. Then the range of lambda is discretized into a pre-specified number of lambda values. Cross validation is then carried out over the set of lambda values.

Value

\(\text{opt.lambda}\)  
the optimal value of lambda chosen by cross validation.

\(\text{n.lambda}\)  
the set of lambda values for cross validation.

\(\text{cv.error}\)  
the overall cross validation error.

\(\text{path}\)  
the \text{msvmpath} object fitted with the given data set.

Author(s)

Zhenhuan Cui

See Also

\text{msvmpath, print.msvmpath, summary.msvmpath, coef.msvmpath, predict.msvmpath}
Examples

data(msvmpath.training)
data(msvmpath.tuning)
data(msvmpath.test)
x.training = as.matrix(msvmpath.training[,1:2])
y.training = msvmpath.training$y
x.tuning = as.matrix(msvmpath.tuning[,1:2])
y.tuning = msvmpath.tuning$y
x.test = as.matrix(msvmpath.test[,1:2])
y.test = msvmpath.test$y

mpath.cv = cv.msvmpath(x.training, y.training, fold = 5, kernel.function = radial.kernel)
y.cv.pred = predict(mpath.cv$xpath, x.test, mpath.cv$opt.lambda, type = "class")
test.error.rate.cv = mean(y.cv.pred != y.test)

**msvmpath**

*Characterize the Solution Path for a Multicategory SVM*

Description

This algorithm extends the binary SVM solution path to multicategory cases, based on Hastie's `svmpath`. `msvmpath` finds the entire solution path of the multicategory SVM at every possible value of the regularization parameter. This algorithm allows selection of the parameter value either with a separate tuning set or a subset of the training set determined by clustering output or simple random sample.

Usage

```r
msvmpath(x, y, kernel.function = poly.kernel, fract = 1, type = "cluster",
         tune.reut = FALSE, cost = cost.dft, wt.obs = wt.dft,...)
```

Arguments

- **x**: a data matrix with n observations (rows) and p covariates (columns) for training.
- **y**: a vector with \{1,2,...,k\} valued class labels for training.
- **kernel.function**: the kernel function. Only poly.kernel and radial.kernel are available now.
- **gpl.init**: 2 to the power of gpl.init is the initial value of the parameter of the radial kernel.
- **param.kernel**: the parameter associated with the kernel function.
- **K**: kernel matrix of the training dataset. It can be extended beyond poly.kernel and radial.kernel.
- **x.tune**: a data matrix for tuning (optional).
- **y.tune**: a vector of class labels for tuning (optional).
- **fract**: the fraction of the training set used to generate the solution path. The default value is 1, which means the entire training set is used.
- **type**: the type of subset sampling approach for large data sets. "cluster" for clustering based approach and "SRS" for simple random sample.
tune.rest  if fract<1, tune.rest indicates whether the tuning error is computed using the rest of the data. If TRUE, the rest of the data are used for tuning. The default value is FALSE.
K.tuning  kernel matrix for the pairs from the training set and the tuning set
eps  a small value for tolerance in the stopping rule
N.moves  the maximal number of steps for the solution path
digits  number of digits displayed
lambda.min  the smallest value of lambda for stopping the path
cost  a k by k cost matrix with ijth entry the cost of misclassifying class i as class j. The default is a k by k matrix with 0's along the diagonal and 1's elsewhere
wt.obs  a vector of length n with data weights. The default is a vector of ones for equal weights.
lag  the number of steps after which the early stopping rule comes into play

Value
An msvmpath object is returned, for which summary, predict, coef, and print methods are available.

alpha  the Lagrange multipliers at each break point of the tuning parameter along the path
alph0  a scaled intercept at each break point of the tuning parameter along the path
lambda  the sequence of break points of the tuning parameter
ind.opt  the location of the optimal lambda
opt.lambda  the optimal value of the tuning parameter computed with the tuning set
Loss  the empirical loss at each break point of the tuning parameter
Error  the training error at each break point of the tuning parameter
tuning.error  the tuning error at each break point if the tuning set is given
Size.Embow  the elbow size at each break point
Elbow  the set of indices in each elbow set at every break point of the tuning parameter
Path.Length  the length of the solution path
Move.to  the set to which an observation is moving at each step
Move.from  the set from which an observation is moving at each step
Step  the step number
wt  an n by p matrix with the item-wise product of the cost and the weight adjusted to the dimension of the training data

Warning
This algorithm is subject to machine errors if eps or lambda.min is too small. When the above problem occurs, increasing either eps or lambda.min can avoid the problem.
When the sample size, the number of classes or the number of covariates is large, this function can provide an approximate solution path.
Author(s)

This function is an extension of the function svmpath written by Trevor Hastie. Zhenhuan Cui generalized the R package svmpath to fit it in the framework of multivary SVM.

References


See Also

summary.msvmpath, cv.msvmpath, print.msvmpath, coef.msvmpath, predict.msvmpath

Examples

data(msvmpath.training)
data(msvmpath.tuning)
data(msvmpath.test)
x.train = as.matrix(msvmpath.training[,1:2])
y.train = msvmpath.training$y
x.tuning = as.matrix(msvmpath.tuning[,1:2])
y.tuning = msvmpath.tuning$y
x.test = as.matrix(msvmpath.test[,1:2])
y.test = msvmpath.test$y

#--------with a separate tuning set for the optimal lambda-----------------
msvpath = msvmpath(x.train, y.train,
kernel.function = radial.kernel, x.tune = x.tuning, y.tune = y.tuning)
summary(msvpath)
print(msvpath)
y.pred = predict(msvpath, x.test, msvpath$opt.lambda, type = "class")
test.error.rate = mean(y.pred != y.test)
test.error.rate

#--------split the training set for the optimal lambda-----------------
msvpath = msvmpath(x.train, y.train, fract = 0.5, tune.rest = TRUE,
kernel.function = radial.kernel)
summary(msvpath)
print(msvpath)
y.pred = predict(msvpath, x.test, msvpath$opt.lambda, type = "class")
test.error.rate.split = mean(y.pred != y.test)
test.error.rate.split

predict.msvmpath Make Prediction for an msvmpath Object

Description

This method computes the predicted values or class labels at new x values or the Lagrange multipliers at new lambda values.

Usage

predict.msvmpath(object, newx, lambda, type = c("function", "class", "alpha"))
Arguments

object       an msvmpath object
newx         x values at which the fitted values are computed or the class labels are predicted
lambda       lambda values at which the Lagrange multipliers are computed
type         type of prediction. If type="function" (default), fitted values are returned. If type="alpha", the Lagrange multipliers are computed. If type="class", class labels are returned.

Details

See predict.msvmpath

Value

This function returns either a matrix of the fitted values or a vector of the corresponding class labels at the new x, or the corresponding Lagrange multipliers for the input lambda values.

Author(s)

This function is an extension of the function predict.msvmpath written by Trevor Hastie. Zhenhuan Cui generalized predict.msvmpath to fit it in the framework of multiclass SVM.

See Also

msvmpath, cv.msvmpath, summary.msvmpath, coef.msvmpath, print.msvmpath

Examples

data(msvmpath.train)
data(msvmpath.tuning)
data(msvmpath.test)
x.train - as.matrix(msvmpath.train[,1:2])
y.train - msvmpath.train$y
x.tuning - as.matrix(msvmpath.tuning[,1:2])
y.tuning - msvmpath.tuning$y
x.test - as.matrix(msvmpath.test[,1:2])
y.test - msvmpath.test$y

msvpath - msvmpath(x.train, y.train,
kernel.function - radial.kernel, x.tune - x.tuning, y.tune - y.tuning,
error.margin - 0.02)
summary(msvpath)
print(msvpath)
y.pred - predict(msvpath, x.test, msvpath$opt.lambda, type - "class")
test.error.rate - mean(y.pred != y.test)
print.msvmpath  Print Method for an msvmpath Object

Description

This method prints the detailed summary of msvm solution path.

Usage

print.msvmpath(x, digits=6, ...)

Arguments

x      an msvmpath object
digits  the number of digits displayed

Value

This method prints the solution path class by class. For each class, it lists the step number, the class label, the observation involved, the event of the observation, the value of lambda, the loss incurred, the elbow size, and the training error. For the event of the observation, "E" stands for the elbow set; "U" stands for the upper set (left set in print.svmpath), "L" stands for the lower set (right set in print.svmpath). "->" indicates the direction of the movement. For example, "L->E" means the movement from the lower set to the elbow set.

Author(s)

This function is an extension of the function print.svmpath written by Trevor Hastie. Zhenhuan Cui generalized print.svmpath to fit it in the framework of multicategory SVM.

See Also

msvmpath, cv.msvmpath, summary.msvmpath, coef.msvmpath, predict.msvmpath

Examples

data(msvmpath.training)
data(msvmpath.tuning)
x.training = as.matrix(msvmpath.training[,1:2])
y.training = msvmpath.training$y
x.tuning = as.matrix(msvmpath.tuning[,1:2])
y.tuning = msvmpath.tuning$y

mpath = msvmpath(x.training, y.training,
kernal.function = radial.kernel, x.tune = x.tuning, y.tune = y.tuning,
error.margin = 0.02)
print(mpath)
summary.msvpath  Summary Method for an msvpath Object

Description

This method provides a brief summary of the solution path with a pre-determined number of steps.

Usage

summary.msvpath(object, nsteps = 5, digits = 6, ...)

Arguments

object  an msvpath object
nsteps  decides the length of the summary.
digits  the number of digits displayed

Details

This method lists the equally spaced steps with the first and last steps included.

Value

This function returns a data frame with the step number, the values of lambda, training error, elbow size, the number of support points, and the loss incurred.

Author(s)

This function is an extension of the function summary.svmpath written by Trevor Hastie. Zhenhuan Cui generalizes summary.svmpath to fit it in the framework of multicityatory SVM.

See Also

msvpath, cv.msvpath, print.msvpath, coef.msvpath, predict.msvpath

Examples

data(msvpath.training)
data(msvpath.tuning)
x.training = as.matrix(msvpath.training[,1:2])
y.training = msvpath.training$y
x.tuning = as.matrix(msvpath.tuning[,1:2])
y.tuning = msvpath.tuning$y

msvpath = msvpath(x.training, y.training,
kernel.function = radial.kernel, x.tune = x.tuning, y.tune = y.tuning,
error.margin = 0.02)
summary(msvpath)
Index

+ Topic classif
  msvmpath, 3
+ Topic methods
  cv.msvmpath, 1
  predict.msvmpath, 5
  print.msvmpath, 7
  summary.msvmpath, 8

  cv.msvmpath, 1

  msvmpath, 3

  predict.msvmpath, 5
  print.msvmpath, 7
  summary.msvmpath, 8


