DEPARTMENT OF STATISTICS
North Carolina State University
2501 Founders Drive, Campus Box 8203
Raleigh, NC 27695-8203

Institute of Statistics Mimeo Series No. 2601

April 27, 2007

Support Vector Machines With Adaptive L_q Penalty

Yufeng Liu
Department of Statistics and Operations Research, UNC-Chapel Hill, NC 27599

Hao Helen Zhang
Department of Statistics, North Carolina State University, Raleigh, NC 27695

Jeongyoun Ahn and Cheolwoo Park
Department of Statistics, University of Georgia, Athens, GA 30602

Supported in part by National Science Foundation grants.

Support Vector Machines With Adaptive L_q Penalty

Yufeng Liu
Department of Statistics and Operations Research
University of North Carolina

Hao Helen Zhang
Department of Statistics
North Carolina State University

Cheolwoo Park and Jeongyoun Ahn
Department of Statistics
University of Georgia

January 31, 2007

Abstract

The standard Support Vector Machine (SVM) minimizes the hinge loss function subject to the L_2 penalty or the roughness penalty. Recently, the L_1 SVM was suggested for variable selection by producing sparse solutions (Bradley and Mangasarian, 1998; Zhu et al., 2003). These learning methods are non-adaptive since their penalty forms are pre-determined before looking at data, and they often perform well only in a certain type of situation. For instance, the L_2 SVM generally works well except when there are too many noise inputs, while the L_1 SVM is more preferred in the presence of many noise variables. In this article we propose and explore an adaptive learning procedure called the L_q SVM, where the best $q > 0$ is automatically chosen by data. Both two- and multi-class classification problems are considered. We show that the new adaptive approach combines the benefit of a

class of non-adaptive procedures and gives the best performance of this class across a variety of situations. Moreover, we observe that the proposed L_q penalty is more robust to noise variables than the L_1 and L_2 penalties. An iterative algorithm is suggested to solve the L_q SVM efficiently. Simulations and real data applications support the effectiveness of the proposed procedure.

Keywords: adaptive penalty, classification, shrinkage, support vector machine, variable selection.

1 Introduction

Classification, a supervised learning approach, is one of the most useful statistical tools for information extraction. Among numerous classification methods, the support vector machine (SVM) is a popular choice and has attracted much attention in recent years. As an important large margin classifier, SVM was originally proposed by V. Vapnik and his colleagues (Boser et al, 1992; Vapnik, 1998) using the idea of searching for the optimal separating hyperplane with maximum separation. It has been successfully applied in various disciplines including engineering, biology, and medicine, and now enjoys great popularity in both machine learning and statistics communities.

Consider a general K-class classification problem in which a training dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$, i.i.d. realizations from $P(\mathbf{X}, Y)$, is given. Here $\mathbf{x}_i \in S \subset \Re^d$ is the input vector and y_i indicates its class label from $1, \ldots, K$. The goal is to construct a classifier which can be used for prediction of y with a new input $\mathbf{x}$. For simplicity, we begin with binary classification problems with $K = 2$ and the class label is coded as $Y \in \{\pm 1\}$. Using the training set, one needs to construct a function f, mapping from S to $\Re$, such that $\text{sign}(f(\mathbf{x}))$ is the classification rule. As the ideal classifier, the Bayes rule minimizes the expected misclassification rate, i.e., $P(Yf(\mathbf{X}) < 0) = 1/2E[1 - \text{sign}(Yf(\mathbf{X}))]$. Consequently, the 0-1 loss, i.e. $1/2(1 - \text{sign})$, on the *margin* $Yf(\mathbf{X})$ is the ultimate loss for accurate classification. However, it is nonconvex and discontinuous, thus very difficult to implement. In practice, convex surrogates are used to obtain good classifiers efficiently. The convex hinge loss of SVM is among them. Under the general regularization framework, the standard binary

SVM solves the following problem

$$\min_f \ \frac{1}{n}\sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i) + \lambda \|f\|_2^2, \tag{1}$$

where $l(f(\mathbf{x}_i), y_i) = [1 - y_i f(\mathbf{x}_i)]_+$ is the convex hinge loss, $\|f\|_2^2$, the L_2 penalty of f, is a regularization term serving as the roughness penalty of f, and $\lambda > 0$ is a tuning parameter which controls the trade-off between the goodness of data fit measured by l and the complexity of f in terms of $\|f\|_2^2$, c.f., Wahba (1998). Lin (2002) showed that binary SVM directly estimates the Bayes classifier $\text{sign}(P(Y = +1|\mathbf{x}) - 1/2)$ rather than $P(Y = +1|\mathbf{x})$ itself.

When the number of classes K is more than two, we need to deal with multi-classification problems. Such problems are frequently encountered in many scientific studies. A good scheme should be powerful in discriminating several classes altogether. Since the binary SVM is not directly applicable in this case, numerous multi-classification procedures have been proposed in the literature. One popular approach, known as "one-versus-rest", proposes to solve the K-class problem by training K separate binary classifiers. However, as argued by Lee, Lin, and Wahba (2004), an approach of this sort may perform poorly in the absence of a dominating class, since the conditional probabilities of all classes are smaller than 1/2. This calls for alternative multicategory SVM methodologies that treat all classes simultaneously. In the literature, there are a number of different multicategory SVM generalizations; for instance, Weston and Watkins (1999), Crammer and Singer (2001), Lee et al. (2004), and others.

Since the L_2 penalty is used in the standard SVM, the resulting classifier utilizes all input variables. This can be a drawback when there are many noise variables among the inputs (Efron et al., 2004). In that situation, those methods for simultaneous classification and variable selection are more preferable to achieve good sparsity and better accuracy. Bradley and Mangasarian (1998) and Zhu et al. (2003) proposed the L_1 SVM for binary problems and showed that variable selection and classification can be conducted jointly through the L_1 penalty. Wang and Shen (2006) extended the idea to multicategory problems. Ikeda and Murata considered the L_q penalty with $q \geq 1$. In practice, a learning procedure with a fixed (non-adaptive) penalty form has its advantages over others only under certain situations, because different types of penalties may suit best for different

data structures. This motivates us to consider an adaptive penalty for binary and multi-class SVMs. We focus on the class of L_q SVMs, $q > 0$, which includes both the L_1 and L_2 penalties as special cases in addition to many other choices. Since the best choice of q varies from problem to problem, we propose to treat q as a tuning parameter and select it adaptively. Numerical studies show that the choice of q is indeed an important factor on the classification performance, and the adaptive approach works as good as or better than any fixed q across a variety of situations.

The rest of this paper is organized as follows. In Section 2, we review the general L_q penalty and its properties in linear regression problems. Section 3 proposes the adaptive L_q SVM and discusses the choice of (λ, q). Both binary and multi-class problems are studied. A local quadratic approximation algorithm is introduced in Section 4. Section 5 presents simulation studies, and real examples are illustrated in Section 6. Some final discussion is given in Section 7.

2 The L_q Penalty and Its Use in Regression

To motivate our methodology, we first explore properties of the L_q penalty in the context of regression problems. Throughout the paper, we assume the function $f(\mathbf{x})$ lies in some linear space spanned by basis functions $\{B_j(\mathbf{x}), j = 1, \ldots, M\}$, i.e., $f(\mathbf{x}) = \sum_{j=1}^{M} w_j B_j(\mathbf{x})$. For linear regression or classification problems, the B_j's are original inputs; alternatively, they can be some nonlinear transformations of a single input or several inputs in $\mathbf{x}$. The L_q penalty on f is defined as

$$\|f\|_q^q = \sum_{j=1}^{M} |w_j|^q.$$

When $q = 0$, the corresponding penalty is discontinuous at the origin and consequently is not easy to compute. Thus we consider $q > 0$ in the paper. In the context of linear regressions, the least squares subject to the L_q penalty with $q > 0$ was first studied by Frank and Friedman (1993) and is known as bridge regression. Fu (1998) and Knight and Fu (2000) studied asymptotic properties and the computation of bridge estimators. When $q = 1$, the approach reduces to the LASSO (Tibshirani, 1996) and is named as basis pursuit in wavelet regression (Chen et al., 1999). For $q \leq 1$, the bridge estimator tends

to shrink small $|w|$'s to exact zeros and hence selects important variables. As pointed out by Theorem 2 in Knight and Fu (2000), when $q > 1$ the amount of shrinkage towards zero increases with the magnitude of the regression coefficients being estimated. In practice, in order to avoid unacceptable large bias for large parameters, the value of q is often chosen not too large. In our numerical examples, we concentrate on $q \in (0, 2]$.

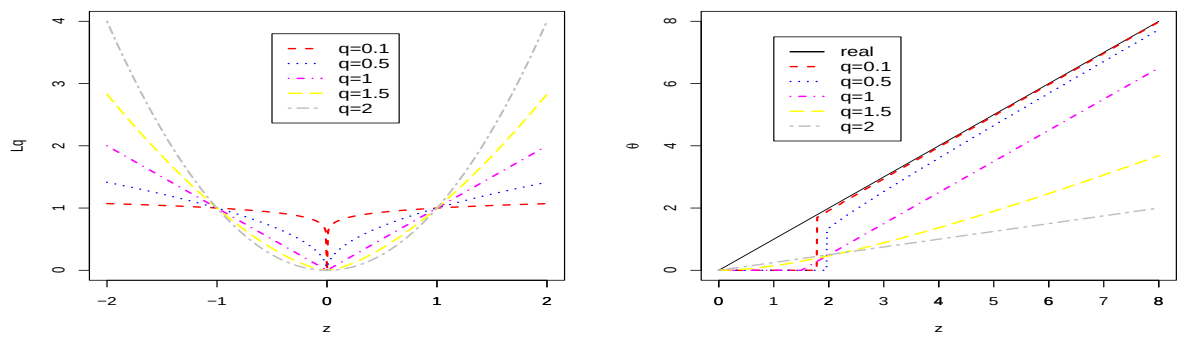

Figure 1: Plots of L_q penalties with different q's (left panel) and the corresponding solutions $\hat{\theta} = \text{argmin}_\theta F_q(\theta)$ (right panel) with $\lambda = 3$, where $F_q(\theta) = (\theta - z)^2 + \lambda|\theta|^q$.

To illustrate the effect of L_q penalties with different q's, we consider a simple linear regression model with one parameter θ and one observation $z = \theta + \epsilon$, where ϵ is a random error with mean 0 and variance σ^2. Without any penalty, the best linear unbiased estimator (BLUE) $\hat{\theta}$ for the parameter θ is z itself. When the L_q penalty is used, we need to solve $\text{argmin}_\theta F_q(\theta)$, where $F_q(\theta) = (\theta - z)^2 + \lambda|\theta|^q$. In Figure 1, we plot the form of the L_q penalty and the corresponding minimizer of $F_q(\theta)$ for various values of q. The L_q function is convex if and only if $q \geq 1$, and not differentiable at $z = 0$ when $q \leq 1$. The singularity property at the origin is crucial for the shrinkage solution to be a thresholding rule (Fan and Li, 2001). If $z = 0$, then the minimizer $\hat{\theta} = 0$. Otherwise, when $z \neq 0$, the behavior of the L_q penalty severely depends on the choice of q, as illustrated in the left plot of Figure 1. If $q \geq 1$, the larger q is, the more penalties are imposed on $|\theta|$'s which are larger than 1 and less penalties are imposed on $|\theta|$'s which are smaller than 1. The situation is opposite for $q < 1$. The following are several special cases for q:

- When $q = 2$, we have the ridge solution $\hat{\theta} = z/(\lambda + 1)$. Note $\hat{\theta}$ is biased and $\text{Var}(\hat{\theta}) = 1/(\lambda + 1)^2\text{Var}(z)$. Therefore $\hat{\theta}$ is better than z when the bias is smaller

compared to variance deduction.

- When $q = 1$, we obtain the lasso solution $\hat{\theta} = \text{sign}(z)[|z| - \lambda/2]_+$. This gives us a thresholding rule, because small $|z|$ leads to a zero solution.

- When $q \in (0, 1)$, we can conclude that $\hat{\theta} = 0$ if and only if $\lambda > |z|^{2-q}(\frac{2}{2-q})[\frac{2(1-q)}{2-q}]^{1-q}$, that is when $|z| < [\lambda(\frac{2-q}{2})(\frac{2-q}{2(1-q)})^{1-q}]^{1/(2-q)}$ (Knight and Fu, 2000).

- When $q = 0$, minimizing $(\theta - z)^2 + \lambda I(|\theta| \neq 0)$ gives $\hat{\theta} = zI(|z| < \sqrt{\lambda})$. This penalty is known as the entropy penalty in wavelet (Donoho and Johnstone, 1994; Antoniadis and Fan, 2001).

For other values of q, it is not easy to get a closed form for $\hat{\theta}$. For $q > 1$, $F(\theta)$ is a strictly convex function and there has only one unique minimizer. It is not hard to show that $\hat{\theta} \neq 0$ if $z \neq 0$, for any $q > 1$. Therefore the L_q penalty with $q > 1$ does not threshold. The right plot in Figure 1 plots the minimizer of $F_q(\theta)$ for different q's with $\lambda = 3$. For $q > 1$, we observe that the solution $|\hat{\theta}|$ is shrunk downward but never becomes zero unless $z = 0$. When $q = 1$, the original estimator is shrunk by a constant and hence variable selection can be achieved. When $q < 1$, the L_q penalty may achieve better sparsity than the L_1 penalty because larger penalty is imposed on small coefficients than the L_1 penalty.

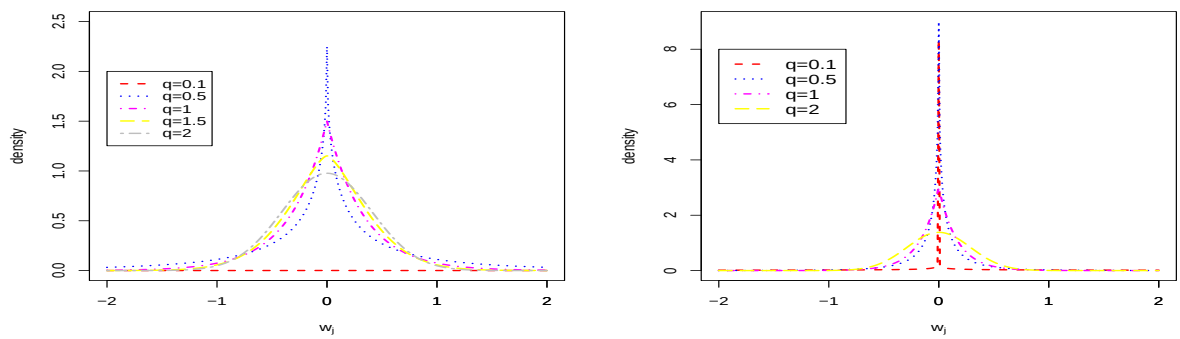

Figure 2: Plots of the density function $\pi_{\lambda,q}(w_j)$ with $\lambda = 3$ (left panel) and 6 (right panel).

The L_q penalty $\sum_{j=1}^{M} |w_j|^q$ has a Bayesian interpretation if we view $\lambda \sum_{j=1}^{M} |w_j|^q$ as negative logarithm of the prior distribution $\exp(-\lambda \sum_{j=1}^{M} |w_j|^q)$ of $\boldsymbol{w}$ subject to a constant.

In general, we can show that the density function of the prior distribution of w_j is

$$\pi_{\lambda,q}(w_j) = \frac{q\lambda^{1/q}}{2\Gamma(1/q)} \exp\left(-\frac{|w_j|^q}{\lambda^{-1}}\right). \tag{2}$$

Two special cases are the normal prior ($q = 2$) and the double exponential prior ($q = 1$), as pointed out by Tibshirani (1996) and Fu (1998). In Figure 2, we plot the densities $\pi_{\lambda,q}$ for different choices of (λ, q). We can observe $\pi_{\lambda,q}$ has more mass around 0 as q gets smaller with a spike at zero only when $q \leq 1$. As a result, the corresponding posterior estimators of w_j with $q \leq 1$ are more likely to be 0.

3 The L_q SVM

3.1 Binary Classification

For binary classification problems with $y \in \{\pm 1\}$, we propose to solve the following SVM with the adaptive L_q penalty

$$\min_f \ \frac{1}{n}\sum_{i=1}^{n} c(-y_i)[1 - y_i f(\mathbf{x}_i)]_+ + \lambda \|f\|_q^q, \tag{3}$$

where $f(\mathbf{x}) = \sum_{j=1}^{M} w_j B_j(\mathbf{x})$, $c(+1)$ and $c(-1)$ are respectively the costs for false positive and false negative. Different from the standard binary SVM, there are two tuning parameters λ and q in (3). The parameter λ, playing the same role as in the non-adaptive SVM, controls the tradeoff between minimizing the hinge loss and the penalty on f. Another tuning parameter q determines the penalty function on f. Here $q \in (0, 2]$ is regarded as a tuning parameter, and it can be adaptively chosen by data together with λ. Lin et al. (2002) showed that the minimizer of $E\{c(-Y)[1 - Yf(\mathbf{X})]_+\}$ is $\text{sign}(P(Y = +1|\mathbf{x}) - \frac{c(-1)}{c(+1)+c(-1)})$, where $[u]_+ = u$ if $u \geq 0$ and 0 otherwise. Clearly, when equal costs are employed, (3) reduces to the standard case.

As mentioned in the previous section, a proper choice of q is important and depends on the nature of data. If there are many noise input variables, the L_q penalty with $q \leq 1$ is desired since it automatically selects important variables and removes many noise variables, consequently the resulting classifier has good generalization and interpretability. On the other hand, if all the covariates are important, it may be more preferable to use

$q > 1$ to avoid unnecessary variable deletion. Therefore, q should be chosen adaptively by data. Figure 3 plots the contours of the normalizing constant $\frac{q\lambda^{1/q}}{2\Gamma(1/q)}$ in $\pi_{\lambda,q}(\theta)$ given in (2) as a function of (λ, q). For a fixed q, the prior distribution with a larger λ tends to put more mass around 0. This amounts to putting a larger weight on the regularization term. For a fixed λ of reasonable size, the prior distribution with a smaller q tends to put more mass around 0, thus more shrinkage on the estimated coefficients can be expected. In summary, (λ, q) interacts much with each other, indicating that a good λ for one q may not be a proper choice for a different q. In practice, we can use cross validation or a separate validation set to tune λ and q together. More discussions about tuning parameters λ and q are provided in Section 3.3.

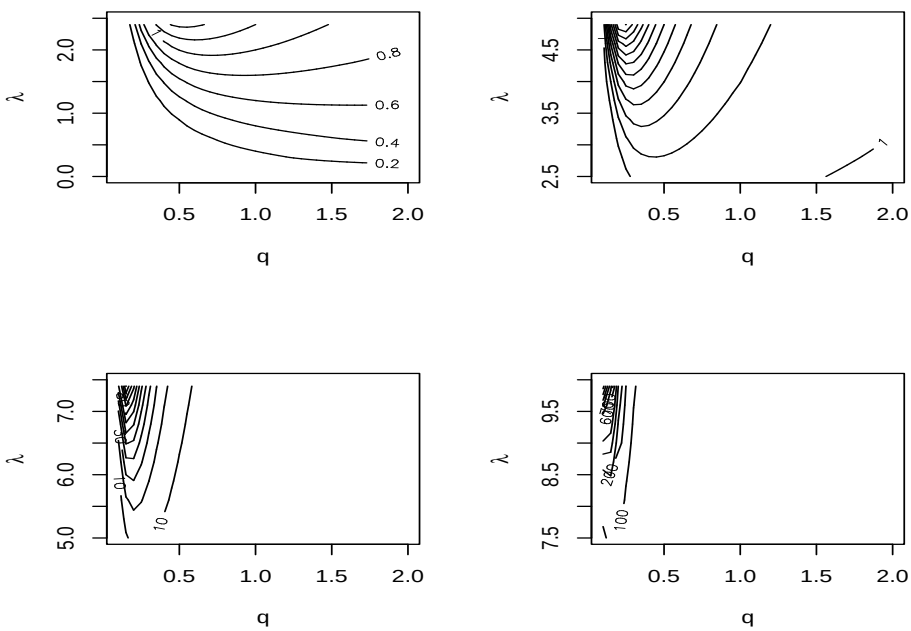

Figure 3: Contour plots of the density coefficient $\frac{q\lambda^{1/q}}{2\Gamma(1/q)}$ in (2) with $q \in (0, 2]$ and $\lambda \in (0, 10)$.

3.2 Multiclass L_q SVM

Consider the multiclass classification problem with K possible class labels $\{1, \ldots, K\}$. Given the training set, we need to learn a function $\phi(\mathbf{x}) : \Re^d \to \{1, \ldots, K\}$ to distinguish

K classes. Let $p_k(\mathbf{x}) = P(Y = k|\mathbf{X} = \mathbf{x})$ be the conditional probability of class k given $\mathbf{X} = \mathbf{x}$, for $k = 1, \ldots, K$. Represent c_{kl} as the cost for classifying an observation in class k to class l. Note that all c_{kk} $(k = 1, \ldots, K)$ entries are set to be 0 since a correct decision should not be penalized. The Bayes rule, minimizing the expected cost of misclassifications

$$E\left[c_{Y\phi(\mathbf{X})}\right] = E_{\mathbf{X}}\left[\sum_{k=1}^{K} c_{k\phi(\mathbf{x})} P(Y = k|\mathbf{X} = \mathbf{x})\right] = E_{\mathbf{X}}\left[\sum_{k=1}^{K} c_{k\phi(\mathbf{x})} p_k(\mathbf{x})\right],$$

is given by

$$\phi_B(\mathbf{x}) = \arg\min_{k=1,\ldots,K}\left[\sum_{l=1}^{K} c_{kl} p_k(\mathbf{x})\right]. \tag{4}$$

When the misclassification costs are all equal, that is, $c_{kl} = 1$ for $l \neq k$, the Bayes rule simplifies to

$$\phi_B(\mathbf{x}) = \arg\min_{k=1,\ldots,K}[1 - p_k(\mathbf{x})] = \arg\max_{k=1,\ldots,K} p_k(\mathbf{x}), \tag{5}$$

which can be interpreted as minimizing the expected misclassification rate $E\{Y \neq \phi(\mathbf{X})\}$.

For multi-classification problems, we need to estimate a K-dimensional function vector $\boldsymbol{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))'$. A sum-to-zero constraint $\sum_{k=1}^{K} f_k(\mathbf{x}) = 0$ for any $\mathbf{x} \in S$ is employed to ensure uniqueness of the solution. Each $f_k(\mathbf{x})$ is assumed to be lying in the space spanned by a number of basis functions, i.e., $f_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj} B_j(\mathbf{x})$. Then we consider a multivariate hinge loss function $\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}[f_k(\mathbf{x}_i) + 1]_+ c_{y_i k}$. This loss function was also adopted by Lee et al. (2004) and it is shown to be Fisher consistent. For simplicity of the notations, we only illustrate the multiclass L_q SVM for the linear case. The extension to nonlinear classifications is straightforward using basis expansion. Moreover, we focus on equal costs with $c_{y_i k} = I(k \neq y_i)$. Denote the linear decision function as $f_k(\mathbf{x}) = b_k + \boldsymbol{w}_k^T \mathbf{x}$, where $\boldsymbol{w}_k = (w_{k1}, ..., w_{kd})^T$ and $k = 1, ..., K$. The sum-to-zero constraint $\sum_{k=1}^{K} f_k(\mathbf{x}) = 0$ is equivalent to $(\sum_{k=1}^{K} b_k = 0, \sum_{k=1}^{K} \boldsymbol{w}_k = \mathbf{0})$. Then the optimization problem becomes

$$\min_{\{(\boldsymbol{w}_k, b_k)_{k=1,\ldots,K}\}} \quad \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}[\boldsymbol{w}_k^T \mathbf{x}_i + b_k + 1]_+ I(k \neq y_i) + \lambda \sum_{k=1}^{K}\sum_{j=1}^{d}|w_{kj}|^q, \tag{6}$$

$$\text{subject to} \quad \sum_{k=1}^{K} b_k = 0, \quad \sum_{k=1}^{K} w_{kj} = 0, \quad \text{for} \quad j = 1, ..., d. \tag{7}$$

The final decision rule for classifying $\mathbf{x}$ is $\hat{\phi}(\mathbf{x}) = \arg\max_{k=1,\ldots,K} \hat{f}_k(\mathbf{x})$. As a remark, we note that problem (6) can be extended for the unequal cost case with $I(k \neq y_i)$ replaced by $c_{y_i k}$.

3.3 Parameter Tuning

For fixed parameters λ and q, let $\hat{\phi}_{\lambda,q}(\mathbf{x})$ be the optimal solution of (3) or (6). In particular, when $K = 2$, $\hat{\phi}(\mathbf{x}) = \text{sign}(\hat{f}(\mathbf{x}))$ where f plays the same role as $f_2 - f_1$ when the label is switched from $\{-1, +1\}$ to $\{1, 2\}$; when $K > 2$, $\hat{\phi}(\mathbf{x}) = \arg\max_{k=1,\cdots,K} \hat{f}_k(\mathbf{x})$. With equal-cost assumptions, the generalization performance of $\hat{\phi}(\mathbf{x})$ is evaluated by the expected misclassification rate

$$\text{MISRATE}(\lambda, q) = E_P \left[Y \neq \hat{\phi}_{\lambda,q}(\mathbf{X}) \right]. \tag{8}$$

Here $\hat{\phi}_{\lambda,q}$ is considered fixed and the expectation is taken over future, unobserved $(\mathbf{X}, Y)$'s. The best parameters are the pair which minimizes (8). However, (8) is not directly computable since P is generally unknown. In the literature, one approach to approximate (8) is to generate a separate tuning set of size n', which is assumed to follow the same distribution as the training set, and compute

$$\frac{1}{n'} \sum_{j=1}^{n'} I(y'_j \neq \hat{\phi}(\mathbf{x}'_j)).$$

Another popular method is the cross validation. In our numerical examples, we generate separate tuning sets in simulated examples, where the true joint distribution $P(\mathbf{X}, Y)$ is known, and use five-fold cross validation in real examples. A two-dimensional grid of (λ, q) will be searched over to find the best tuning parameters.

4 Local Quadratic Approximation Algorithm

When $q = 2$, the optimization problems (3) and (6) can be solved by quadratic programming (QP). In the literature, the dual rather than primal problems are often easier to handle. When $q = 1$, (3) and (6) can be reduced to linear programming (LP). Many standard software packages are available to solve them. Except for these two special cases, the optimization problems (3) and (6) are essentially nonlinear programming (NLP) problems,

which are not easy to solve in general. In this section, we suggest a universal algorithm which solves (3) and (6) for any $q > 0$. As mentioned previously, when $q < 1$ the function $\|f\|_q^q$ is not convex in $\boldsymbol{w}$. Therefore standard optimization routines may fail to minimize the L_q SVM. We propose to use the local quadratic approximation for the objective function and minimize (3) or (6) via iterative quadratic optimization. More details are given in the Appendix.

For simplicity, define $p_\lambda(z) = \lambda|z|^q$ for any fixed q. Using the fact that $z_+ = \frac{z+|z|}{2}$ and the proxy $|z| \approx \frac{1}{2}\frac{z^2}{|z_0|} + \frac{1}{2}|z_0|$ with a nonzero z_0 that is close to z, there are the following approximations:

$$\begin{aligned} z_+ &\approx \frac{1}{4}\frac{z^2}{|z_0|} + \frac{1}{2}z + \frac{1}{4}|z_0|, \\ p_\lambda(|z|) &\approx p_\lambda(|z_0|) + \frac{1}{2}\frac{p'_\lambda(|z_0|)}{|z_0|}(z^2 - z_0^2). \end{aligned}$$

Define the augmented input $\tilde{\mathbf{x}}_i = [1, \mathbf{x}_i^T]^T$, $V_i = [\tilde{\mathbf{x}}_i^T, \cdots, \tilde{\mathbf{x}}_i^T]^T$ as $K-1$ copies of $\tilde{\mathbf{x}}_i$, and $a_{ik} = I(k \neq y_i)$ for $i = 1, ..., n$, $k = 1, ..., K$. Define the vector $\mathbf{v} = (v_1, ..., v_d)^T$ with $v_j = \frac{p'_\lambda(|\sum_{k=1}^{K-1} w_{kj}^0|)}{|\sum_{k=1}^{K-1} w_{kj}^0|}$, where w_{kj}^0 denotes the initial value of w_{kj}. For $j = 1, ..., d$, let $\mathbf{s}_j = \mathbf{1}_{K-1} \otimes \mathbf{t}_j$, where $\mathbf{1}_{K-1}$ is a vector of 1 with length $K-1$, $\mathbf{t}_j$ is the $d+1$-dimensional zero vector except the $(j+1)$th entry being one, and $\otimes$ denotes the Kronecker product. Furthermore, the collection of parameters is denoted by $\boldsymbol{\eta} = [\boldsymbol{\eta}_1^T, ..., \boldsymbol{\eta}_{K-1}^T]^T$, where $\boldsymbol{\eta}_k = [b_k, \boldsymbol{w}_k^T]^T$ for $k = 1, ..., K$. After plugging the equation constraints (7) into (6), we can update $\boldsymbol{\eta}$ by iteratively minimizing the quadratic approximations until convergence. For fixed (λ, q), the local quadratic approximation (LQA) algorithm to solve (6) is summarized as the following three steps:

Step 1: Set $\ell = 1$ and the initial value $\boldsymbol{\eta}^{(1)}$.

Step 2: Let $\boldsymbol{\eta}_0 = \boldsymbol{\eta}^{(\ell)}$. Minimize $F(\boldsymbol{\eta}) = \boldsymbol{\eta}^T \boldsymbol{Q} \boldsymbol{\eta} + \boldsymbol{\eta}^T \boldsymbol{L}$ to obtain $\boldsymbol{\eta}^{(\ell+1)}$, where $\boldsymbol{Q}$ and $\boldsymbol{L}$ are defined in the appendix.

Step 3: Set $\ell = \ell + 1$ and go to Step 2 until convergence.

The algorithm stops when there is little change in $\boldsymbol{\eta}^{(k)}$, say, $\sum_j |\eta_j^{(k+1)} - \eta_j^{(k)}| < \epsilon$, where ϵ is a pre-selected small positive value. In our numerical examples, $\epsilon = 10^{-3}$ is used. Based on our experience, the coefficients of the discriminant functions given by linear discriminant analysis (LDA) provide a good starting value for $\boldsymbol{\eta}^{(1)}$. As a remark,

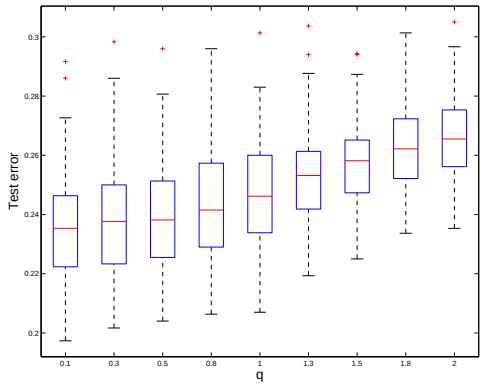

Figure 4: Plot of classification errors in Example 1 as q increases.

we note that the LQA algorithm is very efficient although it is a local algorithm and it may not find the global optimum. Our numerical results in Section 5 suggest that the LQA algorithm works effectively for the proposed L_q SVM.

5 Simulations

In this section, we demonstrate performance of the adaptive L_q SVM, and compare it with those of L_1 and L_2 SVMs under different settings. Three binary classification examples are considered in Section 5.1, with two linear cases and one nonlinear case. One three-class example is illustrated in Section 5.2. The grid search is implemented to find the best tuning parameters (λ, q) based on some independent tuning sets with $q \in (0, 2]$.

5.1 Binary Classification

Example 1 (Linear; many noise variables). We generate the input $\mathbf{x}$ uniformly from the hypercube $[0, 1]^{20}$, and the class label y is assigned by $\text{sign}(f(\mathbf{x}))$, where $f(\mathbf{x}) = 2x_1 + 4x_2 + 4x_3 - 4.8$. Thus the input space is $S = [0, 1]^{20}$, but only the first three variables are important and the rest seventeen variables are noise variables. As a result, the true model size is 3. Both training and tuning sample sizes are 400. For each classifier, we compute its testing error based on its prediction accuracy on an independent testing set of size 3000. The experiment is repeated for 100 times; the average testing error and the average model size are summarized in Table 1. The numbers in the parentheses are the

standard deviations of the estimates.

Since only three out of twenty variables are important, variable shrinkage is necessary in this example to achieve an accurate and sparse classifier. From Table 1, the L_1 SVM performs some model shrinkage and has the average model size 11.79; it also shows better classification accuracy than the L_2 SVM. However, compared with the L_q SVM, the L_1 SVM does not give enough shrinkage. We can see, among the three procedures, the L_q SVM performs the best by producing the sparsest model with the average size 5.28 and the smallest testing error. Furthermore, the resulting L_q SVM classifier never misses any of the three important variables over all 100 runs. In this example, the average q selected by data is 0.4074; hence the data requires more shrinkage than that given by the non-adaptive L_1 penalty.

Method	Test Error	Model Size
Bayes rule	0.2216 (0.0070)	3
L_1 SVM	0.2578 (0.0265)	11.79 (3.40)
L_2 SVM	0.2673 (0.0136)	19.97 (0.17)
L_q SVM	**0.2415** (0.0380)	**5.28** (4.11)

Table 1: Classification accuracy and variable selection results for Example 1.

Figure 4 illustrates how the testing errors change as q increases. Clearly, the testing errors tend to be smaller when q gets closer to 0. This is due to the fact that there are many noise input variables and smaller q's give more shrinkage thus better classification accuracy.

Example 2 (Linear; varying number of sample sizes). The data generation mechanism is as follows. First, generate class label Y with $P(Y = +1) = P(Y = -1) = 1/2$. After the class label is obtained, with probability 0.7, the first three variables $\{x_1, x_2, x_3\}$ are drawn from $x_i \sim yN(i, 1)$ and the second three variables $\{x_4, x_5, x_6\}$ are drawn from $x_i = N(0, 1)$ $(i = 1, 2, 3)$; with probability 0.3 the first three variables are drawn from $x_i = N(0, 1)$ and the second three variables are drawn from $x_i = yN(i-3, 1)$ $(i = 4, 5, 6)$. The remaining noise variables are drawn from $N(0, 20)$ independently.

In this example, numbers of noise variables are increased up to 48. The results based

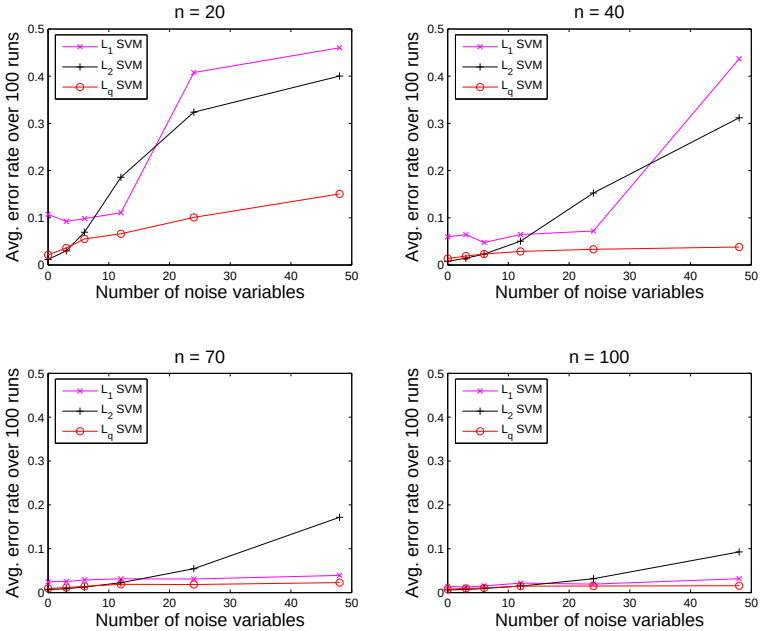

Figure 5: Plots of classification errors in Example 2 as the number of noise variables increases with $n = 20, 40, 70, 100$.

on 100 replications are plotted in Figure 5 with training sample sizes $n = 20, 40, 70, 100$. The tuning sample sizes are same as the corresponding training sample sizes. Testing errors are estimated using independent testing sets of size 3000. On each plot, the x-axis represents the number of noise variables and y-axis represents testing errors of different classifiers. As we can see from these plots, as the number of noise variables increases, the classification task becomes more challenging and consequently the testing errors of all three methods increase. However, the testing error of the L_q SVM increases the slowest and thus its performance becomes more and more superior than the other two methods. When we increase the training sample size, all methods perform better with corresponding testing errors decreasing. Among the three methods, the L_q SVM appears to improve the fastest as n gets bigger. Clearly, the L_q SVM performs the best compared to the other two methods in this example. Moreover, the L_q SVM selects 5–9 variables consistently as we increase the number of variables or decreases the sample size. Thus it is a rather robust classification procedure.

It is interesting to point out that for cases of small sample sizes with n=20 and 40, the L_2 SVM sometimes outperforms the L_1 SVM even when the number of noise variables

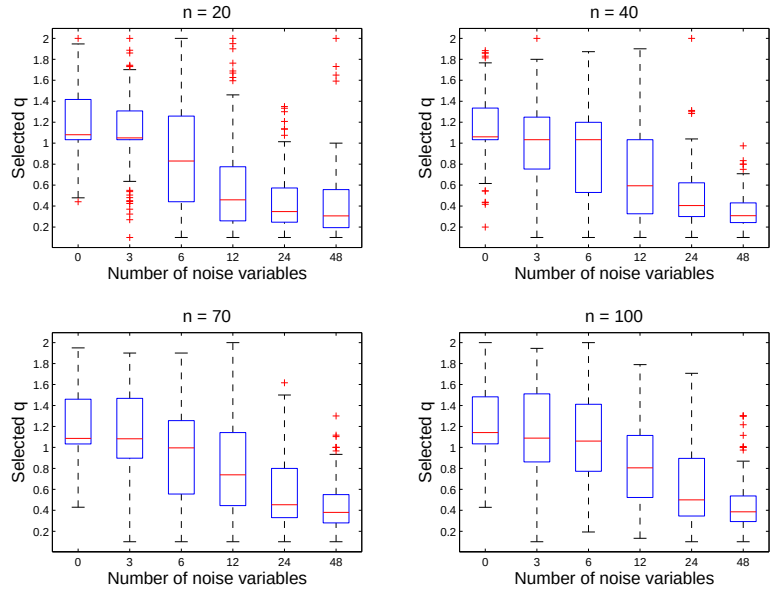

Figure 6: Plot of best selected q's in Example 2 as the numbers of noise variables increase.

is large. One possible explanation is that classification performance has large numerical variability due to small sample sizes. When n gets large, we expect more stability in the results which generally better reflect asymptotic behaviors of different classifiers. In the bottom row of Figure 5, when n increases to 70 and 100, the L_1 SVM clearly demonstrates the overall advantages over the L_2 SVM, as expected, especially when the number of noise variables becomes large. Another possible explanation is that correlations exist among input variables and such correlations can cause difficulties for the L_1 penalty in selecting all correct variables (Zou et al., 2005).

In Figure 6, we plot the best selected q's as the number of noise variables increases. It is clear from the plots that the average selected q's tend to get smaller as the numbers of noise variables increase. This matches our expectation since further shrinkage is needed when there are more noise input variables.

Example 3 (Nonlinear). In this nonlinear example, the data are generated in the following way: First of all, two important variables x_1 and x_2 are generated independently and uniformly from $[0, 1]$. Secondly, the label y is assigned to either class according to the values of $y^* = (x_1 - .5)^2 + (x_2 - .5)^2$. In particular, we set $y = 1$ if $y^* < .07$, $y = -1$ if $y^* > 0.13$, and set y to be either $+1$ or -1 with equal probabilities if $.07 \leq y^* \leq 0.13$. After that, we add m noise variables generated from $N(0, 1)$ to the input vector, where

$m = 0, 20, 40, ...100$.

Polynomial embedding is used to fit three SVM methods; in particular, we map $\{x_j\}_{j=1}^d$ to $\{(x_j, x_j^2)\}_{j=1}^d$. The training, tuning, and testing sample sizes are respectively 200, 200, 3000. Figure 7 shows the results from 100 repetitions of the experiment. The left panel displays how the average testing errors change as the number of noise variables increases for three procedures. The performance of the L_q SVM is quite robust against the increase of noise variables, while the accuracy of the L_1 and L_2 SVMs deteriorates rapidly. The average number of selected variables for each method is shown on the right panel. As observed, the L_q SVM has the smallest model size among the three methods, with the average selected q around 0.25. Moreover, the L_q SVM selects all important variables (x_1, x_1^2, x_2, x_2^2) in all replications. In contrast, the L_2 SVM has no feature selection property so that it includes all noise variables. The L_1 SVM has smaller model sizes than the L_2 SVM, but still keeps some noise variables.

An illustrating plot is given in Figure 8. We plot the projected classification boundaries given by three methods on the two-dimensional space spanned by x_1 and x_2 for one particular data set with $m = 20$. Clearly, the boundary of the L_q SVM is the closest to the Bayes boundary, followed by that of the L_1 SVM. The boundary of the L_2 SVM is the worst in this case.

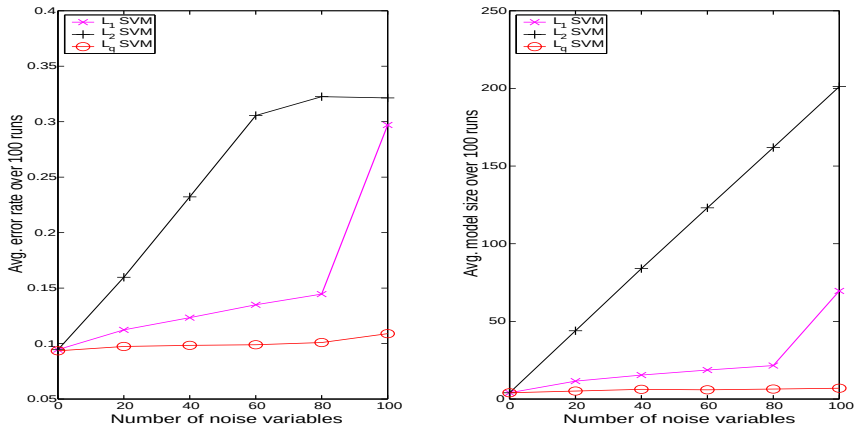

Figure 7: Plots of average misclassification rates and model sizes for Example 3 over 100 runs.

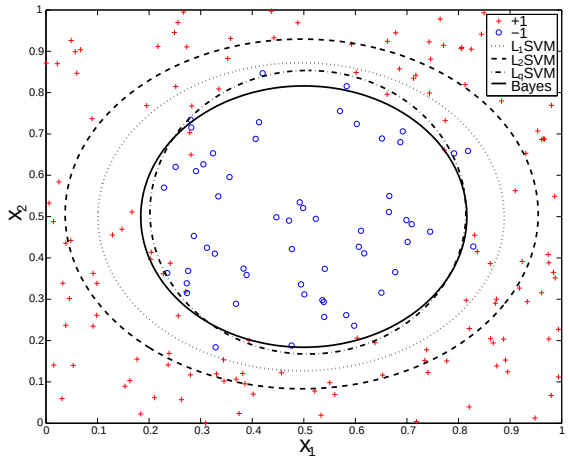

Figure 8: Plot of typical projected decision boundaries on the two-dimensional space spanned by x_1 and x_2 given by the L_1, L_2, and L_q SVMs in Example 3.

5.2 Multi-class Classification

Example 4. Consider one multi-class example with $K = 3$. The training data is generated from a mixture of bivariate Gaussian distributions. For class $k = 1, 2, 3$, we generate $\mathbf{x}$ independently from $N(\boldsymbol{\mu}_k, \sigma^2 I_2)$, respectively with $\boldsymbol{\mu}_1 = (\sqrt{3}, 1)$, $\boldsymbol{\mu}_2 = (-\sqrt{3}, 1)$, $\boldsymbol{\mu}_3 = (0, -2)$, and $\sigma^2 = 2$. Sample sizes are 100 for the training and tuning data and 1000 for the testing data. We report the testing errors and the standard deviations for all three methods in Table 2 based on 100 replications. Three SVM classifiers give comparable performance. And the L_q SVM performs slightly better than the other two in view of its smallest testing error and variation. The average q in this case is 0.783.

Method	Test Error	Model Size
Bayes	0.1845 (0.0013)	2
L_1 SVM	0.2246 (0.0053)	2 (0)
L_2 SVM	0.2214 (0.0048)	2 (0)
L_q SVM	**0.2155** (0.0040)	2 (0)

Table 2: Classification accuracy for Example 4.

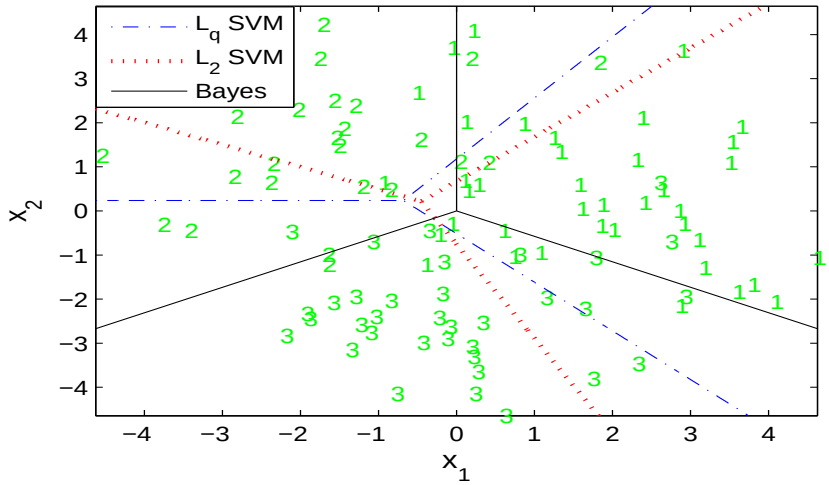

Figure 9: Classification boundaries given by the Bayes rule, the L_2 and L_q SVMs in Example 4.

In Figure 9, we plot the classification boundaries given by the Bayes rule, the L_2 SVM, and the L_q SVM for one particular dataset. The boundary of the L_1 SVM is not plotted since it is very close to that of the L_2 SVM. Symbols "1", "2", and "3" in the plot represent points from three different classes; the solid, dotted, and dashed lines correspond to the Bayes rule, the L_2 SVM, and the L_q SVM respectively. As shown by the plot, the boundary of the L_q SVM is closer to the boundary of the Bayes rule than the L_2 SVM.

6 Real Data

We apply the proposed L_q SVM, together with the non-adaptive L_1 and L_2 SVMs, to three real data sets from the UCI benchmark repository. The first two examples are for binary classification, and the third one is a multi-class problem. Relevant information about these three data sets is: Statlog heart disease data (`hea`; binary, 13 variables, $n = 270$), Pima Indians diabetes data (`pid`; binary, 8 variables, $n = 768$), and Balance scale data (`bal`; three class, 4 variables, $n = 625$). More details can be found at http://www.ics.uci.edu/∼mlearn/MLRepository.html.

For the Pima Indians diabetes dataset, some variables have "impossible" observations as one referee pointed out. For example, Wahba et al. (1995) found that 11 instances

of 0 body mass index and 5 instances of 0 plasma glucose, and they deleted those cases and included the remaining 752 observations. Besides these unrealistic observations, some other variables like diastolic blood pressure and skin-fold thickness also have unrealistic zero values. In particular, the variable serum insulin has 374 (almost 50%) zero values. To keep the sample size reasonably large, we remove the variable insulin and all the cases with unrealistic zero values in variables 2,3,4,6 to get a reduced dataset. We have examined both the full dataset (`pid-c`) as well as the reduced dataset (`pid-r`; binary, 7 variables, $n = 532$).

Since there are no separate testing sets available for these data sets, we randomly divide each data set into three parts and train the classifier on the first 2/3 and test on the remaining 1/3. Five-fold cross validation within the training set is used to choose (λ, q). We repeat this process 10 times and report the average testing errors for three classifiers in Table 3. For all four data sets, the adaptive L_q SVM yields either equivalent good or slightly better performance than the L_1 and L_2 SVMs. The average q is respectively 1.29, 1.35, 1.20, and 1.67 for the four datasets.

	`hea`	`pid-c`	`pid-r`	`bal`
L_1 SVM	.170 (.032)	.240 (.015)	.207 (.026)	.124 (.021)
L_2 SVM	.166 (.033)	.240 (.015)	**.204** (.028)	**.123** (.016)
L_q SVM	**.160** (.018)	**.233** (.010)	**.204** (.022)	**.123** (.025)

Table 3: Classification results for real data sets hea, pid-c, pid-r, and bal.

7 Discussion

In this paper, we propose a new adaptive SVM classification method with the L_q penalty. The L_q SVM allows a flexible penalty form chosen by data; hence the classifier is built based on the best q for any specific application. A unified algorithm is introduced to solve the L_q SVM. Both our simulated and real examples show that the choice of q does play an essential role in improving the accuracy as well as structure of the resulting classifier. Overall, the L_q SVM enjoys better accuracy than the L_1 and L_2 SVMs.

The procedure of selecting (λ, q) is an important step in implementing the L_q SVM. Currently, we apply a grid search coupled with cross validation to the tuning procedure. It is possible, however, to design a more efficient method such as the downhill search for tuning. Further investigation will be pursued in the future.

Acknowledgement

The authors would like to thank two anonymous reviewers for their constructive comments and suggestions. Yufeng Liu's research was partially supported by the National Science Foundation Grant DMS-0606577 and the UNC Junior Faculty Development Award. Hao Helen Zhang's research was partially supported by the National Science Foundation Grants DMS-0405913 and DMS-0645293.

Appendix: Derivation of the LQA Algorithm

By adopting the sum-to-zero constraint, for each i, we have

$$\begin{aligned}\sum_{k=1}^{K}[\boldsymbol{w}_k^T\mathbf{x}_i + b_k + 1]_+ &= \sum_{k=1}^{K-1}[\boldsymbol{w}_k^T\mathbf{x}_i + b_k + 1]_+ + [\boldsymbol{w}_K^T\mathbf{x}_i + b_K + 1]_+ \\ &= \sum_{k=1}^{K-1}[\boldsymbol{w}_k^T\mathbf{x}_i + b_k + 1]_+ + [-\sum_{k=1}^{K-1}\boldsymbol{w}_k^T\mathbf{x}_i - \sum_{k=1}^{K-1} b_k + 1]_+.\end{aligned}$$

Then using the fact that $z_+ = \frac{z+|z|}{2}$ and the approximation $|z| \approx \frac{1}{2}\frac{z^2}{|z_0|} + \frac{1}{2}|z_0|$, it is easy to have $z_+ \approx \frac{1}{4}\frac{z^2}{|z_0|} + \frac{1}{2}z + \frac{1}{4}|z_0|$, $p_\lambda(|z|) \approx p_\lambda(|z_0|) + \frac{1}{2}\frac{p'_\lambda(|z_0|)}{|z_0|}(z^2 - z_0^2)$, where z_0 is some non-zero value close to z. By absorbing the constraints into the objective function, the LQA algorithm iteratively minimizes

$$F(\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_K) = A_1 + A_2 + B_1 + B_2 + C_1 + C_2$$

with

$$
\begin{aligned}
A_1 &= \frac{1}{4n}\sum_{i=1}^{n}\sum_{k=1}^{K-1}\frac{a_{ik}}{|\boldsymbol{\eta}_k^{0T}\tilde{\mathbf{x}}_i+1|}(\boldsymbol{\eta}_k^T\tilde{\mathbf{x}}_i+1)^2,\\
A_2 &= \frac{1}{2n}\sum_{i=1}^{n}\sum_{k=1}^{K-1}(\boldsymbol{\eta}_k^T\tilde{\mathbf{x}}_i+1)a_{ik},\\
B_1 &= \frac{1}{4n}\sum_{i=1}^{n}\frac{a_{iK}}{|-\sum_{k=1}^{K-1}\boldsymbol{\eta}_k^{0T}\tilde{\mathbf{x}}_i+1|}(-\sum_{k=1}^{K-1}\boldsymbol{\eta}_k^T\tilde{\mathbf{x}}_i+1)^2,\\
B_2 &= \frac{1}{2n}\sum_{i=1}^{n}(-\sum_{k=1}^{K-1}\boldsymbol{\eta}_k^T\tilde{\mathbf{x}}_i+1)a_{iK},\\
C_1 &= \lambda\sum_{j=1}^{d}\sum_{k=1}^{K-1}|w_{kj}|^q,\\
C_2 &= \lambda\sum_{j=1}^{d}|\sum_{k=1}^{K-1}w_{kj}|^q.
\end{aligned}
$$

After some matrix algebra, we can get $A_1 = \boldsymbol{\eta}^T Q_{A_1}\boldsymbol{\eta} + \boldsymbol{\eta}^T L_{A_1} + \text{constant}$, $A_2 = \boldsymbol{\eta}^T L_{A_2} + \text{constant}$, $B_1 = \boldsymbol{\eta}^T Q_{B_1}\boldsymbol{\eta} + \boldsymbol{\eta}^T L_{B_1} + \text{constant}$, $B_2 = \boldsymbol{\eta}^T L_{B_2} + \text{constant}$, $C_1 = \boldsymbol{\eta}^T Q_{C_1}\boldsymbol{\eta} + \text{constant}$, and $C_2 = \frac{1}{2}\sum_{j=1}^{d} v_j(\boldsymbol{\eta}^T\mathbf{s}_j)(\mathbf{s}_j^T\boldsymbol{\eta}) = \boldsymbol{\eta}^T Q_{C_2}\boldsymbol{\eta}$, where
$Q_{A_1} = \frac{1}{4n}\text{diag}\left[\sum_{i=1}^{n}\frac{a_{i1}}{|\boldsymbol{\eta}_1^{0T}\tilde{\mathbf{x}}_i+1|}\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^T, \sum_{i=1}^{n}\frac{a_{i2}}{|\boldsymbol{\eta}_2^{0T}\tilde{\mathbf{x}}_i+1|}\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^T, \cdots, \sum_{i=1}^{n}\frac{a_{i,K-1}}{|\boldsymbol{\eta}_{K-1}^{0T}\tilde{\mathbf{x}}_i+1|}\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^T\right]$,
$Q_{B_1} = \frac{1}{4n}\sum_{i=1}^{n}\frac{a_{iK}}{|\sum_{k=1}^{K-1}\boldsymbol{\eta}_k^{0T}\tilde{\mathbf{x}}_i-1|}V_iV_i^T$, $Q_{C_1} = \text{diag}[U_1, U_2, ..., U_{K-1}]$,
with $U_k = \frac{1}{2}\text{diag}[0, \frac{p'_\lambda(|w_{k1}^0|)}{|w_{k1}^0|}, \frac{p'_\lambda(|w_{k2}^0|)}{|w_{k2}^0|}, \cdots, \frac{p'_\lambda(|w_{kd}^0|)}{|w_{kd}^0|}]$, $Q_{C_2} = \frac{1}{2}\sum_{j=1}^{d} v_j\mathbf{s}_j\mathbf{s}_j^T$,
$L_{A_1} = \frac{1}{2n}\left[\sum_{i=1}^{n}\frac{a_{i1}}{|\boldsymbol{\eta}_1^{0T}\tilde{\mathbf{x}}_i+1|}\tilde{\mathbf{x}}_i^T, \sum_{i=1}^{n}\frac{a_{i2}}{|\boldsymbol{\eta}_2^{0T}\tilde{\mathbf{x}}_i+1|}\tilde{\mathbf{x}}_i^T, \cdots, \sum_{i=1}^{n}\frac{a_{i,K-1}}{|\boldsymbol{\eta}_{K-1}^{0T}\tilde{\mathbf{x}}_i+1|}\tilde{\mathbf{x}}_i^T\right]^T$,
$L_{A_2} = \frac{1}{2n}\left[\sum_{i=1}^{n}a_{i1}\tilde{\mathbf{x}}_i^T, \sum_{i=1}^{n}a_{i2}\tilde{\mathbf{x}}_i^T, \cdots, \sum_{i=1}^{n}a_{i,K-1}\tilde{\mathbf{x}}_i^T\right]^T$,
$L_{B_1} = -\frac{1}{2n}\sum_{i=1}^{n}\frac{a_{iK}}{|\sum_{k=1}^{K-1}\boldsymbol{\eta}_k^{0T}\tilde{\mathbf{x}}_i-1|}V_i$, and $L_{B_2} = -\frac{1}{2n}\sum_{i=1}^{n}V_ia_{iK}$.
Therefore, $F(\boldsymbol{\eta}) = \boldsymbol{\eta}^T\boldsymbol{Q}\boldsymbol{\eta} + \boldsymbol{\eta}^T\boldsymbol{L}$, where $\boldsymbol{Q} = Q_{A_1} + Q_{B_1} + Q_{C_1} + Q_{C_2}$ and $\boldsymbol{L} = L_{A_1} + L_{A_2} + L_{B_1} + L_{B_2}$. The desired algorithm then follows.

References

Antoniadis, A. and Fan, J. (2001). Regularization of Wavelets Approximations. *Journal of the American Statistical Association.* **96**, 939-967.

Boser, B., Guyon, I., and Vapnik, V. N. (1992). A training algorithm for optimal margin

classifiers. *The Fifth Annual Conference on Computational Learning Theory.* Pittsburgh ACM, 142-152.

Bradley, P. and Mangasarian, O. (1998). Feature selection via concave minimization and support vector machines. In J Shavlik(eds), *ICML'98.* Morgan Kaufmann.

Chen, S., Donoho, D. L., and Saunders, M. A. (1999). Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, **20**, 1, 33-61.

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

Donoho, D. and Johnstone, I. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, **81**, 425-455.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, **32**(2), 407-499.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, **96**, 456, 1348-1360.

Frank, I. E. and Friedman, J. H. (1993). An statistical view of some chemometrics regression tools. *Technometrics*, **35**, 109-135.

Fu, W. J. (1998). Penalized regressions: the bridge vs the lasso. *Journal of Computational and Graphical Statistics*, **7**, 3, 397-416.

Ikeda, K. and Murata, N. (2005). Geometrical properties of ν support vector machines with different norms. *Neural Computation*, **17**(11), 2508-2529.

Knight, K. and Fu, W. J. (2000). Asymptotics for lasso-type estimators. *Annals of Statisitics*, **28**(5), 1356-1378.

Lee, Y., Lin, Y., and Wahba, G. (2004). Multicategory Support Vector Machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association.* **99**, 465: 67-81.

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

Lin, Y., Lee, Y., and Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning.* **46**, 191-202.

Tibshirani, R. J. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of*

Royal Statistical Society, Ser. B, **58**, 267-288.

Vapnik, V. (1998). Statistical learning theory. *Wiley.*

Wahba, G. (1998). Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV. In: B. Schölkopf, C. J. C. Burges and A. J. Smola (eds), *Advances in Kernel Methods: Support Vector Learning*, MIT Press, 125-143.

Wahba, G., Gu, C., Wang, Y., and Chappell, R. (1995). Soft Classification, a.k.a. Risk Estimation, via Penalized Log Likelihood and Smoothing Spline Analysis of Variance. In "Computational Learning Theory and Natural Learning Systems," Volume 3, T. Petsche, Ed, MIT Press pp. 127-158.

Wang, L. and Shen, X. (2006). Multi-category Support vector machines, feature selection, and solution path. *Statistica Sinica.* 16, 617-633.

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

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

Zou, H. and Trevor Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society*, Series B, **67**, 301-320.