Kernel Logistic Regression and the Import Vector Machine

Ji Zhu
Department of Statistics
University of Michigan
Ann Arbor, MI 48109-1092
jizhu@umich.edu

Trevor Hastie
Department of Statistics
Stanford University
Stanford, CA 94305
hastie@stat.stanford.edu

May 8, 2004

Abstract

The support vector machine (SVM) is known for its good performance in two-class classification, but its extension to multi-class classification is still an on-going research issue. In this paper, we propose a new approach for classification, called the import vector machine (IVM), which is built on kernel logistic regression (KLR). We show that the IVM not only performs as well as the SVM in two-class classification, but also can naturally be generalized to the multi-class case. Furthermore, the IVM provides an estimate of the underlying probability. Similar to the support points of the SVM, the IVM model uses only a fraction of the training data to index kernel basis functions, typically a much smaller fraction than the SVM. This gives the IVM a potential computational advantage over the SVM.

Keywords: classification, kernel methods, logistic regression, multi-class learning, radial basis, reproducing kernel Hilbert space (RKHS), support vector machines.

1 Introduction

In standard classification problems, we are given a set of training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, where the input $\mathbf{x}_i \in \mathcal{R}^p$ and the output y_i is qualitative and assumes values in a finite set $\mathcal{C}$, e.g. $\mathcal{C} = \{1, 2, \ldots, C\}$. We wish to find a classification rule from the training data, so that when given a new input $\mathbf{x}$, we can assign a class c from $\mathcal{C}$ to it. Usually it is assumed that the training data are an independently and identically distributed sample from an unknown probability distribution $P(X, Y)$.

The support vector machine (SVM) works well in two-class classification, i.e. $y \in \{-1, 1\}$, but its appropriate extension to the multi-class case is still an on-going research issue (e.g., Vapnik, 1998; Weston & Watkins, 1999; Bredensteiner & Bennett, 1999; Lee *et al.*, 2002). Another property of the SVM is that it only estimates $sign[p(\mathbf{x}) - 1/2]$ (Lin, 2002), while the probability $p(\mathbf{x})$ is often of interest itself, where $p(\mathbf{x}) = P(Y = 1|X = \mathbf{x})$ is the conditional probability of a point being in class 1 given $X = \mathbf{x}$. In this paper, we propose a new approach, called the import vector machine (IVM), to address the classification problem. We show that the IVM not only performs as well as the SVM in two-class classification, but also can naturally be generalized to the multi-class case. Furthermore, the IVM provides an estimate of the probability $p(\mathbf{x})$. Similar to the *support points* of the SVM, the IVM model uses only a fraction of the training data to index the kernel basis functions. We call these training data *import points*. The computational cost of the SVM is $O(n^2 n_s)$ (e.g., Kaufman, 1999), where n_s is the number of support points and n_s usually increases linearly with n, while the computational cost of the IVM is $O(n^2 m^2)$, where m is the number of import points. Since m does not tend to increase as n increases, the IVM can be faster than the SVM. Empirical results show that the number of import points is usually much less than the number of support points.

In section (2), we briefly review some results of the SVM for two-class classification and compare it with kernel logistic regression (KLR). In section (3), we propose our IVM algorithm. In section (4), we show some numerical results. In section (5), we generalize the IVM to the multi-class case.

2 Support vector machines and kernel logistic regression

The standard SVM produces a non-linear classification boundary in the original input space by constructing a linear boundary in a transformed version of the original input space. The dimension

of the transformed space can be very large, even infinite in some cases. This seemingly prohibitive computation is achieved through a positive definite reproducing kernel $K(\cdot,\cdot)$, which gives the inner product in the transformed space.

Many people have noted the relationship between the SVM and regularized function estimation in the reproducing kernel Hilbert spaces (RKHS). An overview can be found in Burges (1998), Evgeniou *et al.* (1999), Wahba (1999) and Hastie *et al.* (2001). Fitting an SVM is equivalent to:

$$\min_{f\in\mathcal{H}_K} \frac{1}{n}\sum_{i=1}^{n}[1-y_i f(\mathbf{x}_i)]_+ + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_K}. \tag{1}$$

where $\mathcal{H}_K$ is the RKHS generated by the kernel $K(\cdot,\cdot)$. The classification rule is given by $sign[f(\mathbf{x})]$. For the purpose of simple notation, we omit the constant term in $f(\mathbf{x})$.

By the representer theorem (Kimeldorf & Wahba, 1971), the optimal $f(\mathbf{x})$ has the form:

$$f(\mathbf{x}) = \sum_{i=1}^{n} a_i K(\mathbf{x},\mathbf{x}_i). \tag{2}$$

It often happens that a sizeable fraction of the n values of a_i can be zero. This is a consequence of the truncation property of the first part of criterion (1). This seems to be an attractive property, because only the points on the wrong side of the classification boundary, and those on the right side but near the boundary have an influence in determining the position of the boundary, and hence have non-zero a_i's. The corresponding $\mathbf{x}_i$'s are called *support points*.

Notice that (1) has the form $loss + penalty$. The loss function $(1-yf)_+$ is plotted in Figure 1, along with the negative log-likelihood (NLL) of the binomial distribution. As we can see, the NLL of the binomial distribution has a similar shape to that of the SVM: both increase linearly as yf gets very small (negative) and both encourage y and f to have the same sign. If we replace $(1-yf)_+$ in (1) with $\ln(1+e^{-yf})$, the NLL of the binomial distribution, the problem becomes a kernel logistic regression (KLR) problem:

$$\min_{f\in\mathcal{H}_K} \frac{1}{n}\sum_{i=1}^{n}\ln\left(1+e^{-y_i f(\mathbf{x}_i)}\right) + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_K}. \tag{3}$$

Because of the similarity between the two loss functions, we expect that the fitted function performs similarly to the SVM for two-class classification.

There are two immediate advantages of making such a replacement: (a) Besides giving a classification rule, KLR also offers a natural estimate of the probability $p(\mathbf{x}) = e^{f(\mathbf{x})}/(1+e^{f(\mathbf{x})})$, while the SVM only estimates $sign[p(\mathbf{x})-1/2]$ (Lin, 2002); (b) KLR can naturally be generalized to

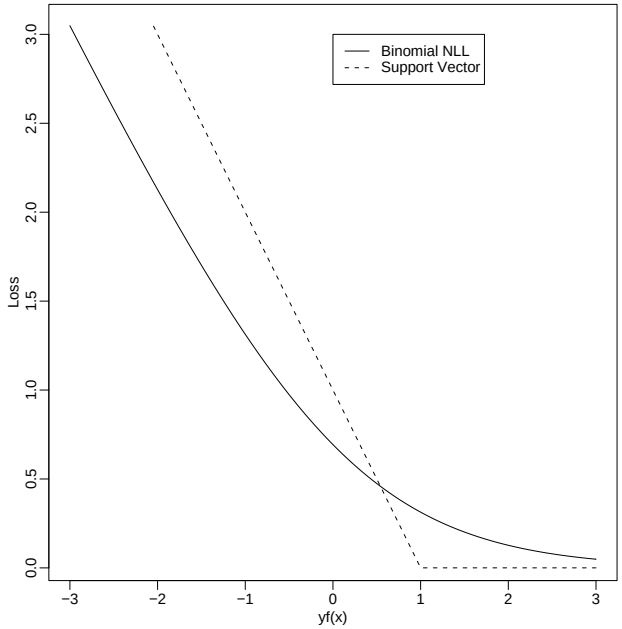

Figure 1: *Two loss functions,* $y \in \{-1, 1\}$.

the multi-class case through kernel multi-logit regression. However, because KLR compromises the hinge loss function of the SVM, it no longer has the support points property; in other words, all the a_i's in (2) are non-zero.

KLR is a well-studied problem; see Green & Yandell (1985), Hastie & Tibshirani (1990), Wahba *et al.* (1995) and references therein; however, they are all under the smoothing spline analysis of variance scheme.

We use a simulation example to illustrate the similar performances between KLR and the SVM. The data in each class are simulated from a mixture of Gaussian distribution (Hastie *et al.*, 2001): first we generate 10 means $\boldsymbol{\mu}_k$ from a bivariate Gaussian distribution $N((1,0)^T, \mathbf{I})$ and label this class $+1$. Similarly, 10 more are drawn from $N((0,1)^T, \mathbf{I})$ and labeled class -1. Then for each class, we generate 100 observations as follows: for each observation, we pick an $\boldsymbol{\mu}_k$ at random with probability $1/10$, and then generate a $N(\boldsymbol{\mu}_k, \mathbf{I}/5)$, thus leading to a mixture of Gaussian clusters for each class.

We use the radial basis kernel

$$K(\mathbf{x}_i, \mathbf{x}_{i'}) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_{i'}\|^2}{2\sigma^2}}. \tag{4}$$

The regularization parameter λ is chosen to achieve good misclassification error. The results are shown in Figure 2. The radial basis kernel produces a boundary quite close to the Bayes optimal boundary for this simulation. We see that the fitted model of KLR is quite similar in classification

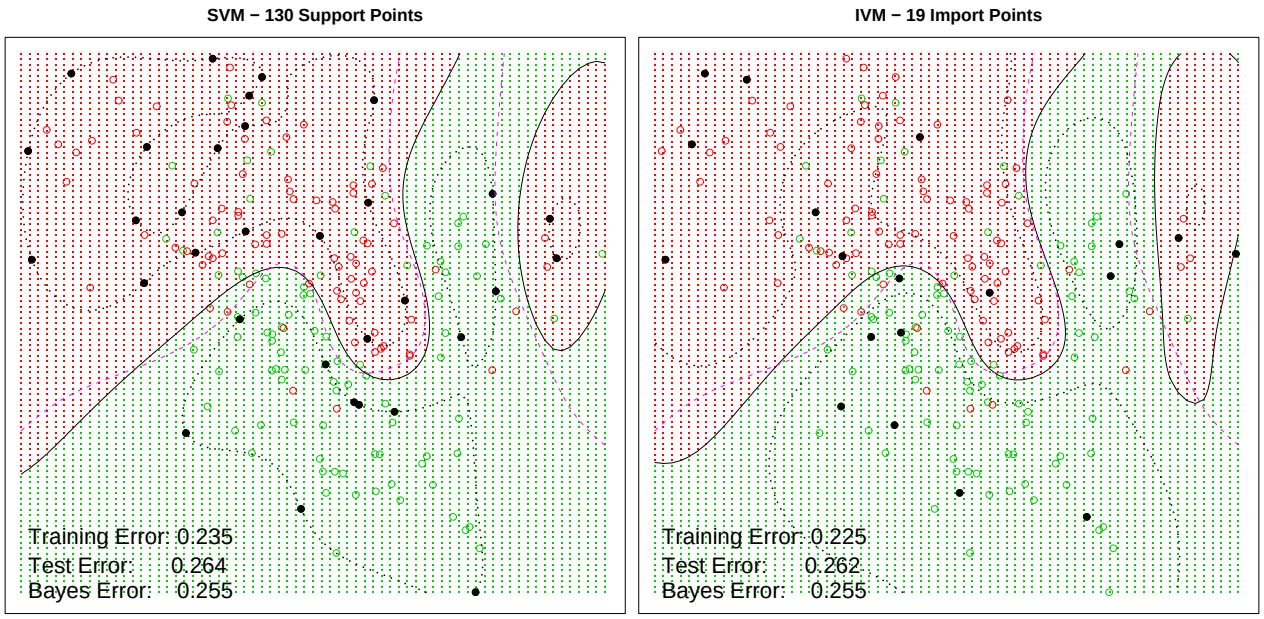

Figure 2: *The solid black lines are classification boundaries; the dashed purple lines are Bayes optimal boundaries. For the SVM, the dotted black lines are the edges of the margins and the black points are the points exactly on the edges of the margin. For the IVM, the dotted black lines are the* $p_1(\mathbf{x}) = 0.25$ *and* 0.75 *lines and the black points are the import points. Since the classification boundaries of KLR and the IVM are almost identical, we omit the picture of KLR here.*

performance to that of the SVM. In addition to a classification boundary, since KLR estimates the log-odds of class probabilities, it can also produce probability contours (Figure 2).

2.1 KLR as a margin maximizer

The SVM was initiated as a method to maximize the margin, i.e. $\min_i y_i f(\mathbf{x}_i)$, of the training data; KLR is motivated by the similarity in shape between the NLL of the binomial distribution and the hinge loss of the SVM. Then a natural question is: what does KLR do with the margin?

Suppose the dictionary of the basis functions of the transformed feature space is

$$\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_q(\mathbf{x})\},$$

where q is the dimension of the transformed feature space. Note if $q = p$ and $h_j(\mathbf{x})$ is the jth component of $\mathbf{x}$, the transformed feature space is reduced to the original input space. The classification boundary, a hyperplane in the transformed feature space, is given by

$$\{\mathbf{x} : f(\mathbf{x}) = \beta_0 + \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta} = 0\}.$$

Suppose the transformed feature space is so rich that the training data are separable, then the margin-maximizing SVM can be written as:

$$\max_{\beta_0,\boldsymbol{\beta},\|\boldsymbol{\beta}\|_2=1} \quad D \tag{5}$$

$$\text{subject to} \quad y_i\left(\beta_0 + \mathbf{h}(\mathbf{x}_i)^T\boldsymbol{\beta}\right) \geq D, \quad i = 1, \ldots, n \tag{6}$$

where D is the shortest distance from the training data to the separating hyperplane and is defined as the *margin* (Burges, 1998).

Now consider an equivalent setup of KLR:

$$\min_{\beta_0,\boldsymbol{\beta}} \quad \sum_{i=1}^{n} \ln\left(1 + e^{-y_i f(\mathbf{x}_i)}\right) \tag{7}$$

$$\text{subject to} \quad \|\boldsymbol{\beta}\|_2^2 \leq s \tag{8}$$

$$f(\mathbf{x}_i) = \beta_0 + \mathbf{h}(\mathbf{x}_i)^T\boldsymbol{\beta}, \quad i = 1, \ldots, n. \tag{9}$$

Then we have the following theorem:

Theorem 1 *Suppose the training data are separable, i.e.* $\exists\beta_0, \boldsymbol{\beta}$, *s.t.* $y_i(\beta_0 + \mathbf{h}(\mathbf{x}_i)^T\boldsymbol{\beta}) > 0, \ \forall i$. *Let the solution of (7)–(9) be denoted by* $\hat{\boldsymbol{\beta}}(s)$, *then*

$$\frac{\hat{\boldsymbol{\beta}}(s)}{s} \to \boldsymbol{\beta}^* \quad as \quad s \to \infty,$$

where $\boldsymbol{\beta}^*$ *is the solution of the margin-maximizing SVM (5)–(6), if* $\boldsymbol{\beta}^*$ *is unique.*

If $\boldsymbol{\beta}^*$ *is not unique, then* $\frac{\hat{\boldsymbol{\beta}}(s)}{s}$ *may have multiple convergence points, but they will all represent margin-maximizing separating hyperplanes.*

The proof of the theorem is delayed in the Appendix. Theorem 1 implies that KLR, similar to the SVM, can also be considered as a margin maximizer. We have also proved a more general theorem relating loss functions and margin maximizers in Rosset *et al.* (2004).

2.2 Computational considerations

Since (3) is convex, it is natural to use the Newton-Raphson method to fit KLR. In order to guarantee convergence, suitable bisection steps can be combined with the Newton-Raphson iterations. The drawback of the Newton-Raphson method is that in each iteration, an $n \times n$ matrix needs to be inverted. Therefore the computational cost of KLR is $O(n^3)$. Recently Keerthi *et al.* (2002) proposed a dual algorithm for KLR which avoids inverting huge matrices. It follows the spirit of

the popular sequential minimal optimization (SMO) algorithm (Platt, 1999). Preliminary computational experiments show that the algorithm is robust and fast. Keerthi *et al.* (2002) describes the algorithm for two-class classification; we have generalized it to the multi-class case (Zhu & Hastie, 2003).

Although the sequential minimal optimization method helps reduce the computational cost of KLR, in the fitted model (2), all the a_i's are non-zero. Hence, unlike the SVM, KLR does not allow for data compression and does not have the advantage of less storage and quicker evaluation.

In this paper, we propose an import vector machine (IVM) model that finds a sub-model to approximate the full model (2) given by KLR. The sub-model has the form:

$$f(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{S}} a_i K(\mathbf{x}, \mathbf{x}_i) \tag{10}$$

where $\mathcal{S}$ is a subset of the training data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, and the data in $\mathcal{S}$ are called *import points*. The advantage of this sub-model is that the computational cost is reduced, especially for large training data sets, while not jeopardizing the performance in classification; and since only a subset of the training data are used to index the fitted model, data compression is achieved.

Several other researchers have also investigated techniques in selecting the subset $\mathcal{S}$. Lin *et al.* (2000) divided the training data into several clusters, then randomly selected a representative from each cluster to make up $\mathcal{S}$. Smola & Schölkopf (2000) developed a greedy technique to sequentially select m columns of the kernel matrix $[K(\mathbf{x}_i, \mathbf{x}_{i'})]_{n \times n}$, such that the span of these m columns approximates the span of $[K(\mathbf{x}_i, \mathbf{x}_{i'})]_{n \times n}$ well in the Frobenius norm. Williams & Seeger (2001) proposed randomly selecting m points of the training data, then using the Nystrom method to approximate the eigen-decomposition of the kernel matrix $[K(\mathbf{x}_i, \mathbf{x}_{i'})]_{n \times n}$, and expanding the results back up to n dimensions. None of these methods uses the output y_i in selecting the subset $\mathcal{S}$ (i.e., the procedure only involves $\mathbf{x}_i$). The IVM algorithm uses both the output y_i and the input $\mathbf{x}_i$ to select the subset $\mathcal{S}$, in such a way that the resulting fit approximates the full model well. The idea is similar to that used in Luo & Wahba (1997), which also used the output y_i to select a subset of the training data, but under the regression scheme.

3 Import vector machines

In KLR, we want to minimize:

$$H = \frac{1}{n}\sum_{i=1}^{n} \ln\left(1 + e^{-y_i f(\mathbf{x}_i)}\right) + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_K}. \tag{11}$$

Let

$$\begin{aligned}
p_i &= \frac{1}{1 + e^{-y_i f(\mathbf{x}_i)}}, \quad i = 1, \ldots n && (12)\\
\mathbf{a} &= (a_1, \ldots, a_n)^T && (13)\\
\mathbf{p} &= (p_1, \ldots, p_n)^T && (14)\\
\mathbf{y} &= (y_1, \ldots, y_n)^T && (15)\\
\mathbf{K}_1 &= (K(\mathbf{x}_i, \mathbf{x}_{i'}))_{i,i'=1}^{n} && (16)\\
\mathbf{K}_2 &= \mathbf{K}_1 && (17)\\
\mathbf{W} &= \mathrm{diag}\,(p_1(1-p_1), \ldots, p_n(1-p_n)) && (18)
\end{aligned}$$

With some abuse of notation, using (2), (11) can be written in a finite dimensional form:

$$H = \frac{1}{n}\mathbf{1}^T \ln\left(\mathbf{1} + e^{-\mathbf{y}\cdot(\mathbf{K}_1\mathbf{a})}\right) + \frac{\lambda}{2}\mathbf{a}^T\mathbf{K}_2\mathbf{a}, \tag{19}$$

where "$\cdot$" denotes element-wise multiplication. To find $\mathbf{a}$, we set the derivative of H with respect to $\mathbf{a}$ equal to 0, and use the Newton-Raphson method to iteratively solve the score equation. It can be shown that the Newton-Raphson step is a weighted least squares step:

$$\mathbf{a}^{(k)} = (\frac{1}{n}\mathbf{K}_1^T\mathbf{W}\mathbf{K}_1 + \lambda\mathbf{K}_2)^{-1}\mathbf{K}_1^T\mathbf{W}\mathbf{z} \tag{20}$$

where $\mathbf{a}^{(k)}$ is the value of $\mathbf{a}$ in the kth step, and

$$\mathbf{z} = \frac{1}{n}\left(\mathbf{K}_1\mathbf{a}^{(k-1)} + \mathbf{W}^{-1}(\mathbf{y}\cdot\mathbf{p})\right). \tag{21}$$

3.1 Basic algorithm

As mentioned in section 2, we want to find a subset $\mathcal{S}$ of $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, such that the sub-model (10) is a good approximation of the full model (2). Since searching for every subset $\mathcal{S}$ is a combinatorial problem and computationally prohibitive, we use the following greedy forward strategy: we start with the null model, i.e. $\mathcal{S} = \emptyset$, then iteratively build up $\mathcal{S}$ one element at a time. Basically, we look for a data point among $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \setminus \mathcal{S}$, so that if it is added into the

current $\mathcal{S}$, the new sub-model will decrease the regularized negative log-likelihood the most:

Algorithm 1 *Basic IVM Algorithm*

1. Let $\mathcal{S} = \emptyset$, $\mathcal{L} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, $k = 1$.

2. For each $\mathbf{x}_l \in \mathcal{L}$, let

$$f_l(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{S} \cup \{\mathbf{x}_l\}} a_i K(\mathbf{x}, \mathbf{x}_i)$$

   Use the Newton-Raphson method to find $\mathbf{a}$ to minimize

$$H(\mathbf{x}_l) = \frac{1}{n} \sum_{i=1}^{n} \ln\left(1 + \exp(-y_i f_l(\mathbf{x}_i))\right) + \frac{\lambda}{2} \|f_l(\mathbf{x})\|^2_{\mathcal{H}_K} \tag{22}$$

$$= \frac{1}{n} \mathbf{1}^T \ln\left(\mathbf{1} + \exp(-\mathbf{y} \cdot (\mathbf{K}_1^l \mathbf{a}))\right) + \frac{\lambda}{2} \mathbf{a}^T \mathbf{K}_2^l \mathbf{a} \tag{23}$$

   where the regressor matrix

$$\mathbf{K}_1^l = \left(K(\mathbf{x}_i, \mathbf{x}_{i'})\right)_{n \times k}, \ \mathbf{x}_i \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_{i'} \in \mathcal{S} \cup \{\mathbf{x}_l\};$$

   the regularization matrix

$$\mathbf{K}_2^l = \left(K(\mathbf{x}_i, \mathbf{x}_{i'})\right)_{k \times k}, \mathbf{x}_i, \mathbf{x}_{i'} \in \mathcal{S} \cup \{\mathbf{x}_l\};$$

   and $k = |\mathcal{S}| + 1$.

3. Find

$$\mathbf{x}_{l^*} = \text{argmin}_{\mathbf{x}_l \in \mathcal{L}} H(\mathbf{x}_l).$$

   Let $\mathcal{S} = \mathcal{S} \cup \{\mathbf{x}_{l^*}\}$, $\mathcal{L} = \mathcal{L} \setminus \{\mathbf{x}_{l^*}\}$, $H_k = H(\mathbf{x}_{l^*})$, $k = k + 1$.

4. Repeat steps (2) and (3) until H_k converges.

The points in $\mathcal{S}$ are the import points.

3.2 Revised algorithm

The above algorithm is computationally feasible, but in step (2) we need to use the Newton-Raphson method to find $\mathbf{a}$ iteratively. When the number of import points k becomes large, the Newton-Raphson computation can be expensive. To reduce this computation, we use a further approximation.

Instead of iteratively computing $\mathbf{a}^{(k)}$ until it converges, we can just do a one-step iteration, and use it as an approximation to the converged one. This is equivalent to approximating the negative binomial log-likelihood with a different weighted quadratic loss function at each iteration. To get a good approximation, we take advantage of the fitted result from the current "optimal" $\mathcal{S}$, i.e., the sub-model when $|\mathcal{S}| = k-1$, and use it to compute $\mathbf{z}$ in (21). This one-step update is similar to the score test in generalized linear models (GLM); but the latter does not have a penalty term. The updating formula allows the weighted regression (20) to be computed in $O(nm)$ time.

Hence, we have the revised steps (1) and (2) for the basic algorithm:

Algorithm 2 *Revised Steps (1) and (2)*

(1*) Let $\mathcal{S} = \emptyset$, $\mathcal{L} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, $k = 1$. Let $\mathbf{a}^{(0)} = \mathbf{0}$, hence $\mathbf{z} = 2\mathbf{y}/n$.

(2*) For each $\mathbf{x}_l \in \mathcal{L}$, correspondingly augment $\mathbf{K}_1$ with a column, and $\mathbf{K}_2$ with a column and a row. Use the current sub-model from iteration $(k-1)$ to compute $\mathbf{z}$ in (21) and use the updating formula (20) to find $\mathbf{a}$. Compute (23).

3.3 Stopping rule for adding point to $\mathcal{S}$

In step (4) of the basic algorithm, we need to decide whether H_k has converged. A natural stopping rule is to look at the regularized NLL. Let $H_1, H_2, \ldots$ be the sequence of regularized NLL's obtained in step (3). At each step k, we compare H_k with $H_{k-\Delta k}$, where Δk is a pre-chosen small integer, for example $\Delta k = 1$. If the ratio $\frac{|H_k - H_{k-\Delta k}|}{|H_k|}$ is less than some pre-chosen small number ϵ, for example, $\epsilon = 0.001$, we stop adding new import points to $\mathcal{S}$.

3.4 Choosing the regularization parameter λ

So far, we have assumed that the regularization parameter λ is fixed. In practice, we also need to choose an "optimal" λ. We can randomly split all the data into a training set and a tuning set, and use the misclassification error on the tuning set as a criterion for choosing λ. To reduce the computation, we take advantage of the fact that the regularized NLL converges faster for a larger λ. Thus, instead of running the entire revised algorithm for each λ, we propose the following procedure, which combines both adding import points to $\mathcal{S}$ and choosing the optimal λ:

Algorithm 3 *Simultaneous Selection of $\mathcal{S}$ and λ*

1. Start with a large regularization parameter λ.

2. Let $\mathcal{S} = \emptyset$, $\mathcal{L} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, $k = 1$. Let $\mathbf{a}^{(0)} = \mathbf{0}$, hence $\mathbf{z} = 2\mathbf{y}/n$.

3. Run steps (2*), (3) and (4) of the revised Algorithm 2, until the stopping criterion is satisfied at $\mathcal{S} = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\}$. Along the way, also compute the misclassification error on the tuning set.

4. Decrease λ to a smaller value.

5. Repeat steps (3) and (4), starting with $\mathcal{S} = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\}$.

We choose the optimal λ as the one that corresponds to the minimum misclassification error on the tuning set.

4 Numerical results

In this section, we use both simulation and real data to illustrate the IVM method.

4.1 Simulation results

The data are generated in the same way as Figure 2. The simulation results are shown in Figure 3 – Figure 5.

Figure 3 shows how the tuning parameter λ is selected. The optimal λ is found to be equal to 1 and corresponds to a misclassification rate 0.262. Figure 4 fixes the tuning parameter to $\lambda = 1$ and finds 19 import points. Figure 2 compares the results of the SVM and the IVM: the SVM has 130 support points, and the IVM uses 19 import points; they give similar classification boundaries. Figure 5 is for the same simulation but different sizes of training data: $n = 200, 400, 600, 800$. We see that as the size of training data n increases, the number of import points does not tend to increase.

Remarks: The support points of the SVM are those which are close to the classification boundary or misclassified and usually have large weights $p(\mathbf{x})(1 - p(\mathbf{x}))$. The import points of the IVM are those that decrease the regularized NLL the most, and can be either close to or far from the classification boundary. This difference is natural, because the SVM is only concerned with the classification $sign\,[p(\mathbf{x}) - 1/2]$, while the IVM also focuses on the unknown probability $p(\mathbf{x})$. Though points away from the classification boundary do not contribute to determining the position of the classification boundary, they may contribute to estimating the unknown probability $p(\mathbf{x})$.

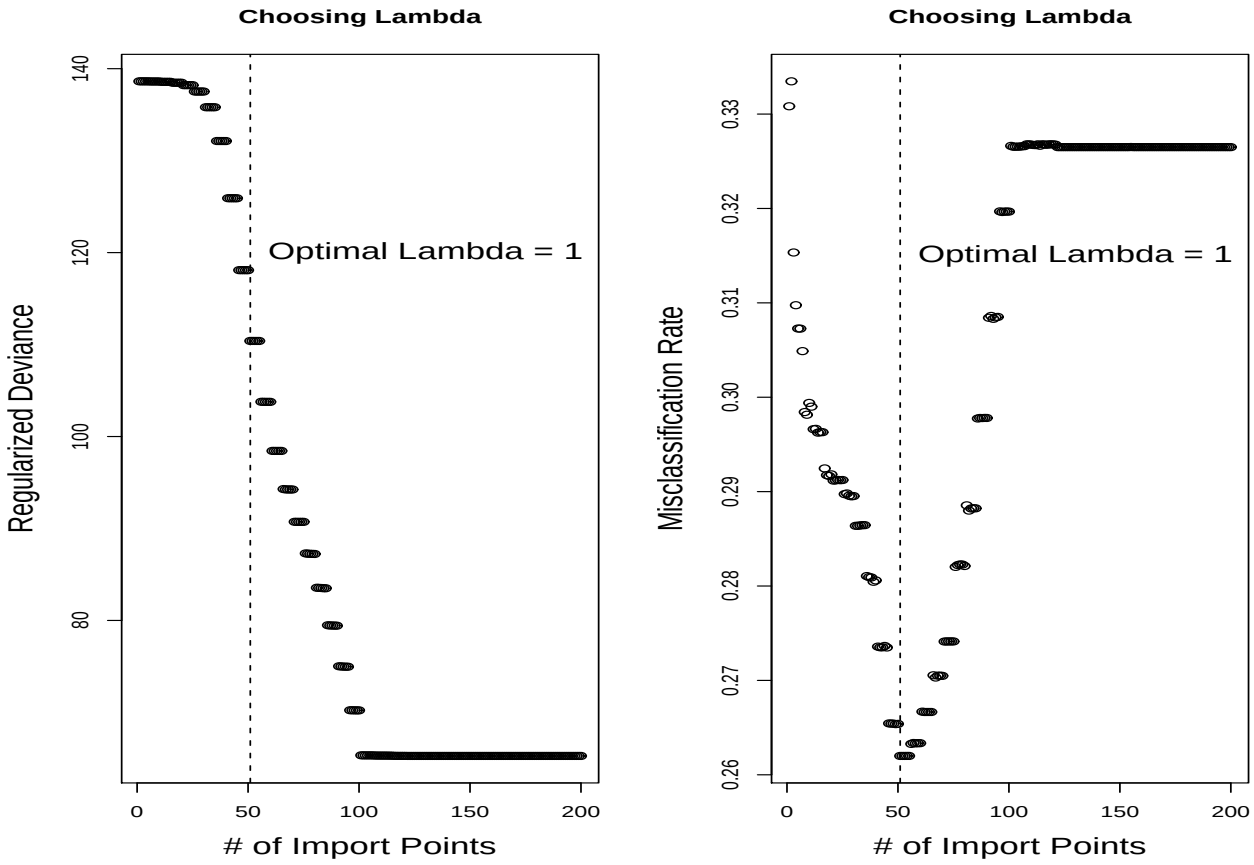

Figure 3: *Radial kernel is used.* $n = 200$, $\sigma^2 = 0.7$, $\Delta k = 3$, $\epsilon = 0.001$, λ *decreases from* e^{10} *to* e^{-10}. *The minimum misclassification rate* 0.262 *is found to correspond to* $\lambda = 1$.

The total computational cost of the SVM is $O(n^2 n_s)$ (e.g., Kaufman, 1999), where n_s is the number of support points, while the computational cost of the IVM method is $O(n^2 m^2)$, where m is the number of import points. Since m does not tend to increase as n increases, as illustrated in Figure 5, the computational cost of the IVM can be smaller than that of the SVM.

4.2 Real data results

In this section, we compare the performances of the IVM and the SVM on some real datasets. Ten benchmark datasets are used for this purpose: Banana, Breast-cancer, Flare-solar, German, Heart, Image, Ringnorm, Splice, Thyroid, Titanic, Twonorm and Waveform. Detailed information about these datasets can be found in Rätsch *et al.* (2000).

Table 1 contains a summary of these datasets. Radial kernel (4) is used throughout these datasets. The parameters σ and λ are fixed at specific values that are optimal for the SVM's

Table 1: Summary of the ten benchmark datasets. n is the size of the training data, p is the dimension of the original input, σ^2 is the parameter of the radial kernel, λ is the tuning parameter, and N is the size of the test data.

Dataset	n	p	σ^2	λ	N
Banana	400	2	1	3.16×10^{-3}	4900
Breast-cancer	200	9	50	6.58×10^{-2}	77
Flare-solar	666	9	30	0.978	400
German	700	20	55	0.316	300
Heart	170	13	120	0.316	100
Image	1300	18	3	0.002	1010
Ringnorm	400	20	10	10^{-9}	7000
Thyroid	140	5	3	0.1	75
Titanic	150	3	2	10^{-5}	2051
Twonorm	400	20	40	0.316	7000
Waveform	400	21	20	1	4600

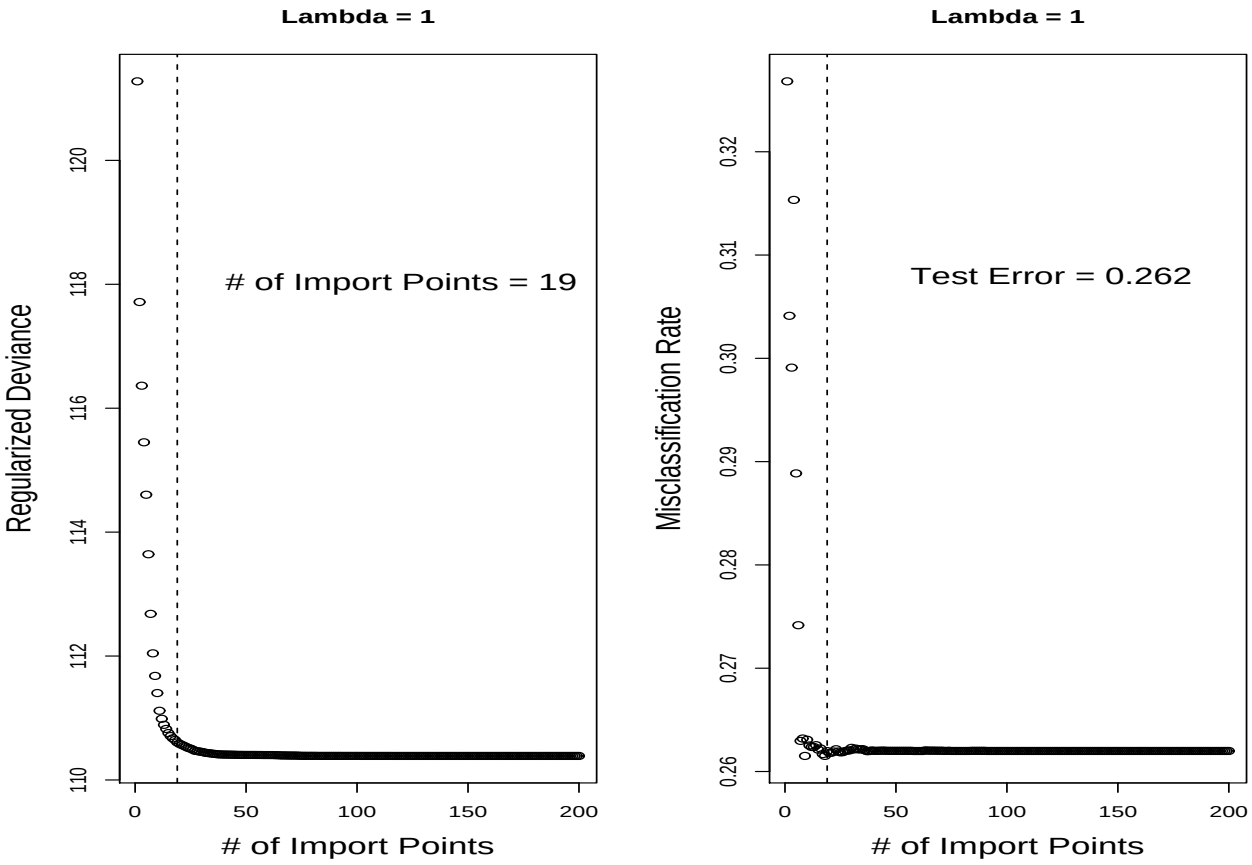

Figure 4: *Radial kernel is used.* $n = 200$, $\sigma^2 = 0.7$, $\Delta k = 3$, $\epsilon = 0.001$, $\lambda = 1$. *The stopping criterion is satisfied when* $|\mathcal{S}| = 19$.

generalization performance (Rätsch *et al.*, 2000). Each dataset has 20 realizations of the training and test data. The results are in Table 2 and Table 3. The number outside each bracket is the mean over 20 realizations of the training and test data, and the number in each bracket is the standard error. From Table 2, we can see that the IVM performs as well as the SVM in classification on these benchmark datasets. From Table 3, we can see that the IVM typically uses a much smaller fraction of the training data than the SVM to index kernel basis functions. This may give the IVM a computational advantage over the SVM.

5 Multi-class case

In this section, we briefly describe a generalization of the IVM to multi-class classification. Suppose there are C classes. The conditional probability of a point being in class c given $X = \mathbf{x}$ is denoted

Table 2: Comparison of classification performance of SVM and IVM on ten benchmark datasets.

Dataset	SVM Error (%)	IVM Error (%)
Banana	10.78(±0.68)	10.34(±0.46)
Breast-cancer	25.58(±4.50)	25.92(±4.79)
Flare-solar	32.65(±1.42)	33.66(±1.64)
German	22.88(±2.28)	23.53(±2.48)
Heart	15.95(±3.14)	15.80(±3.49)
Image	3.34(0.70)	3.31(±0.80)
Ringnorm	2.03(±0.19)	1.97(±0.29)
Thyroid	4.80(±2.98)	5.00(±3.02)
Titanic	22.16(±0.60)	22.39(±1.03)
Twonorm	2.90(±0.25)	2.45(±0.15)
Waveform	9.98(±0.43)	10.13(±0.47)

Table 3: Comparison of number of kernel basis used by SVM and IVM on ten benchmark datasets.

Dataset	# of SV	# of IV
Banana	90(±10)	21(±7)
Breast-cancer	115(±5)	14(±3)
Flare-solar	597(±8)	9(±1)
German	407(±10)	17(±2)
Heart	90(±4)	12(±2)
Image	221(±11)	72(±18)
Ringnorm	89(±5)	72(±30)
Thyroid	21(±2)	22(±3)
Titanic	69(±9)	8(±2)
Twonorm	70(±5)	24(±4)
Waveform	151(±9)	26(±3)

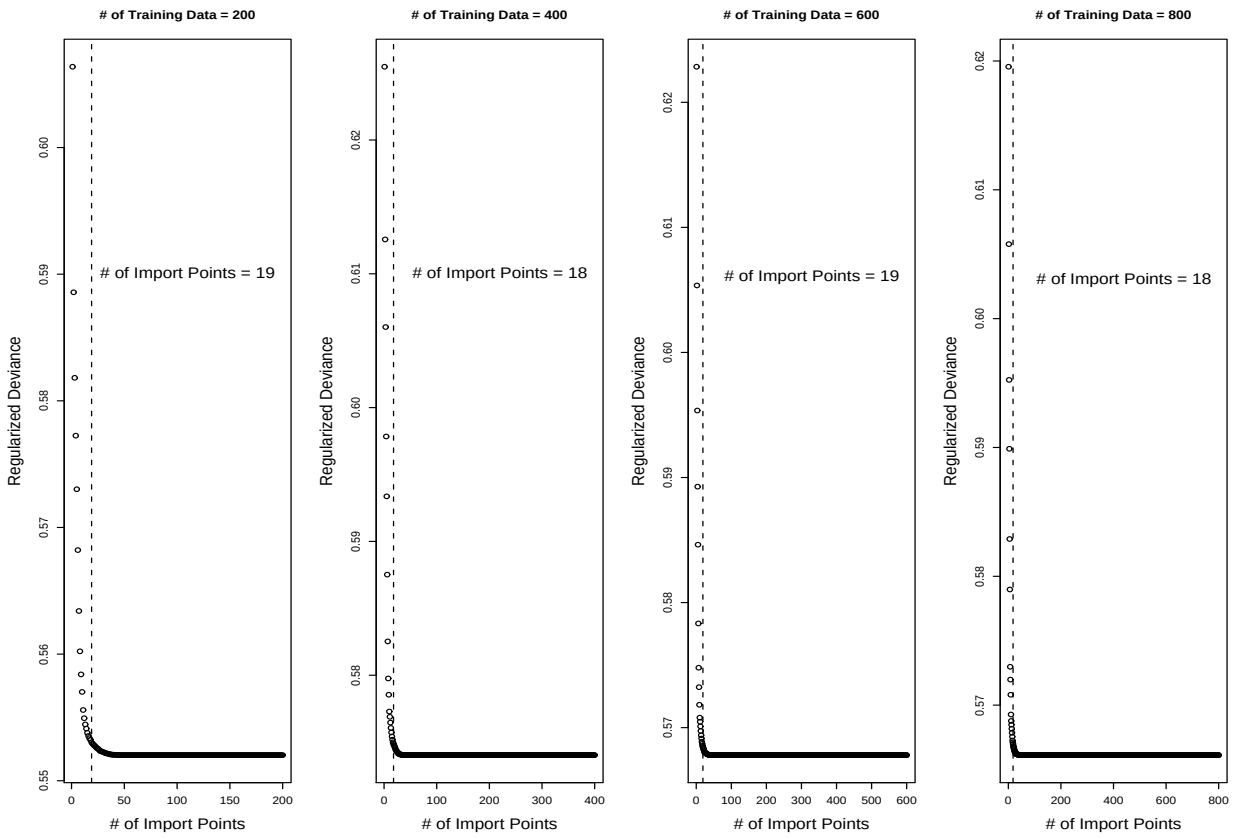

Figure 5: *The data are generated in the same way as Figure 2 – Figure 4. Radial kernel is used.* $\sigma^2 = 0.7$, $\lambda = 1$, $\Delta k = 3$, $\epsilon = 0.001$. *The sizes of training data are* $n = 200, 400, 600, 800$, *and the corresponding numbers of import points are* $19, 18, 19, 18$.

as $p_c(\mathbf{x}) = P(Y = c|X = \mathbf{x})$. Hence the Bayes classification rule is given by:

$$c(\mathbf{x}) = \text{argmax}_{c\in\{1,\dots,C\}} p_c(\mathbf{x})$$

The model has the form:

$$p_1(\mathbf{x}) = \frac{e^{f_1(\mathbf{x})}}{\sum_{c=1}^{C} e^{f_c(\mathbf{x})}}, \tag{24}$$

$$p_2(\mathbf{x}) = \frac{e^{f_2(\mathbf{x})}}{\sum_{c=1}^{C} e^{f_c(\mathbf{x})}}, \tag{25}$$

$$\vdots \tag{26}$$

$$p_C(\mathbf{x}) = \frac{e^{f_C(\mathbf{x})}}{\sum_{c=1}^{C} e^{f_c(\mathbf{x})}}, \tag{27}$$

where $f_c(\mathbf{x}) \in \mathcal{H}_K$; $\mathcal{H}_K$ is the RKHS generated by a positive definite kernel $K(\cdot,\cdot)$. Notice that $f_1(\mathbf{x}), \dots, f_C(\mathbf{x})$ are not identifiable in this model, for if we add a common term to each $f_c(\mathbf{x})$,

$p_1(\mathbf{x}), \ldots, p_C(\mathbf{x})$ will not change. To make $f_c(\mathbf{x})$ identifiable, we consider the symmetric constraint

$$\sum_{c=1}^{C} f_c(\mathbf{x}) = 0. \tag{28}$$

Then the multi-class KLR fits a model to minimize the regularized negative log-likelihood

$$\begin{aligned} H &= -\frac{1}{n}\sum_{i=1}^{n} \ln p_{y_i}(\mathbf{x}_i) + \frac{\lambda}{2}\|\mathbf{f}\|^2_{\mathcal{H}_K} && (29) \\ &= \frac{1}{n}\sum_{i=1}^{n}\left[-\mathbf{y}_i^T \mathbf{f}(\mathbf{x}_i) + \ln\left(e^{f_1(\mathbf{x}_i)} + \cdots + e^{f_C(\mathbf{x}_i)}\right)\right] + \frac{\lambda}{2}\|\mathbf{f}\|^2_{\mathcal{H}_K} && (30) \end{aligned}$$

where $\mathbf{y}_i$ is a binary C-vector with values all zero except a 1 in position c if the class is c, and

$$\begin{aligned} \mathbf{f}(\mathbf{x}_i) &= (f_1(\mathbf{x}_i), \ldots, f_C(\mathbf{x}_i))^T, && (31) \\ \|\mathbf{f}\|^2_{\mathcal{H}_K} &= \sum_{c=1}^{C} \|f_c\|^2_{\mathcal{H}_K}. && (32) \end{aligned}$$

Using the representer theorem (Kimeldorf & Wahba, 1971), one can show that $f_c(\mathbf{x})$, which minimizes H, has the form

$$f_c(\mathbf{x}) = \sum_{i=1}^{n} a_{ic} K(\mathbf{x}_i, \mathbf{x}). \tag{33}$$

Hence, (29) becomes

$$H = \frac{1}{n}\sum_{i=1}^{n}\left[-\mathbf{y}_i^T (\mathbf{K}_1(i,)\mathbf{A})^T + \ln\left(\mathbf{1}^T e^{(\mathbf{K}_1(i,)\mathbf{A})^T}\right)\right] + \frac{\lambda}{2}\sum_{c=1}^{C} \mathbf{a}_c^T \mathbf{K}_2 \mathbf{a}_c \tag{34}$$

where $\mathbf{A} = (\mathbf{a}_1 \cdots \mathbf{a}_C)$, $\mathbf{K}_1$ and $\mathbf{K}_2$ are defined in the same way as in the two-class case; and $\mathbf{K}_1(i,)$ is the ith row of $\mathbf{K}_1$. Notice that in this model, the constraint (28) is not necessary anymore, for at the minimum of (29), $\sum_{c=1}^{C} f_c(\mathbf{x}) = 0$ is automatically satisfied.

5.1 Multi-class KLR and Multi-class SVM

Similar to Theorem 1, a connection between the multi-class KLR and a multi-class SVM also exists.

In going from the two-class SVM to the multi-class classification, many researchers have proposed various procedures.

In practice, the one-vs-rest scheme is often used: given C classes, the problem is divided into a series of C one-vs-rest problems, and each one-vs-rest problem is addressed by a different class-specific SVM classifier (e.g., "class 1" vs. "not class 1"); then a new sample takes the class of the

classifier with the largest real valued output $c^* = \text{argmax}_{c=1,\dots,C} f_c$, where f_c is the real valued output of the cth SVM classifier.

Instead of solving C problems, Vapnik (1998) and Weston & Watkins (1999) generalized (1) by solving one single optimization problem:

$$\max_{f_c} \quad D \tag{35}$$

$$\text{subject to} \quad f_{y_i}(\mathbf{x}_i) - f_c(\mathbf{x}_i) \geq D(1 - \xi_{ic}), \tag{36}$$

$$i = 1, \dots n, \quad c = 1, \dots C, c \neq y_i \tag{37}$$

$$\xi_{ic} \geq 0, \sum_i \sum_{c \neq y_i} \xi_{ic} \leq \lambda \tag{38}$$

$$\sum_{c=1}^{C} \|f_c\|^2_{\mathcal{H}_K} = 1. \tag{39}$$

Recently, Lee *et al.* (2002) pointed out that (35) – (39) is not always Bayes optimal. They proposed an algorithm that implements the Bayes classification rule and estimates $\text{argmax}_c P(Y = c|X = \mathbf{x})$ directly.

Here we propose a theorem that illustrates the connection between the multi-class KLR and one version of the multi-class SVM.

Theorem 2 *Suppose the training data are pairwise separable, i.e.* $\exists f_c(\mathbf{x})$*, s.t.* $f_{y_i}(\mathbf{x}_i) - f_c(\mathbf{x}_i) > 0$, $\forall i$, $\forall c \neq y_i$*. Then as* $\lambda \to 0$*, the classification boundary given by the multi-class KLR (29) will converge to that given by the multi-class SVM (35) – (39), if the latter is unique.*

The proof of the theorem is very similar to that of Theorem 1, we omit it here. Note that in the case of separable classes, (35) – (39) is guaranteed to be Bayes optimal.

5.2 Multi-class IVM

The multi-class IVM procedure is similar to the two-class case (we omit the details), and the computational cost is $O(Cn^2m^2)$. Figure 6 is a simulation of the multi-class IVM. The data in each class are generated from a mixture of Gaussians (Hastie *et al.*, 2001).

6 Discussion

The support vector machine has been very successful for two-class classification and gained a lot of attention in the machine learning society in the past ten years. Many papers have been published

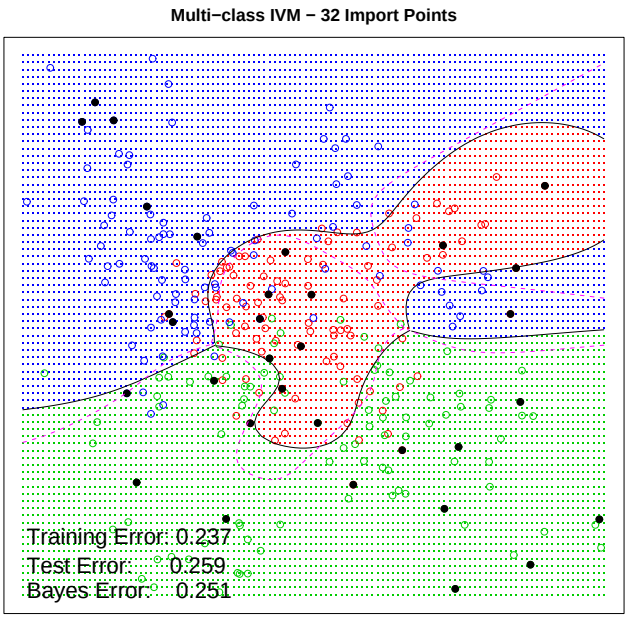

Figure 6: *Radial kernel is used.* $C = 3$, $n = 300$, $\lambda = 0.368$, $|\mathcal{S}| = 32$.

to explain why the support vector machine performs well in two-class classification. Most of this literature concentrates on the concept of margin. Various misclassification error bounds have been derived based on the margin (e.g., Vapnik, 1995; Vapnik, 1998; Bartlett & Shawe-Taylor, 1999; Shawe-Taylor & Cristianini, 1999).

However, our view in this paper is a little different from that based on the concept of margin. Several researchers have noted the relationship between the support vector machine and regularized function estimation in RKHS (e.g., Evgeniou *et al.*, 1999; Wahba, 1999; Hastie *et al.*, 2001). The regularized function estimation problem contains two parts: a loss function and a regularization term. What is special with the support vector machine is the loss function, i.e. the hinge loss. The margin maximizing property of the support vector machine derives from the hinge loss function. Lin (2002) pointed out that the hinge loss is Bayes consistent, i.e. the population minimizer of the loss function agrees with the Bayes rule in terms of classification. We believe this is a big step in explaining the success of the support vector machine, because it implies the support vector machine is trying to implement the Bayes rule. However, this is only part of the story; we believe that the regularization term has also played an important role in the support vector machine's success.

Regularization is an essential component in modern data analysis, in particular when the number of basis functions is large, possibly larger than the number of observations, and non-regularized

fitting is guaranteed to give badly over-fitted models. The enlarged feature space in the support vector machine allows the fitted model to be flexible, and the regularization term controls the complexity of the fitted model; the L_2 nature of the regularization term in the support vector machine allows the fitted model to have a finite representation, even if the fitted model is in an infinite dimensional space. Hence we propose that by replacing the hinge loss of the support vector machine with the negative binomial log-likelihood, which is also Bayes consistent, we should be able to get a fitted model that performs similarly to the support vector machine. The resulting kernel logistic regression is something our statisticians are very familiar with (e.g., Green & Yandell, 1985; Hastie & Tibshirani, 1990; Wahba *et al.*, 1995). We all understand why it can work well. The same reasoning could be applied to the support vector machine. The import vector machine algorithm is just a way to compress the data and reduce the computational cost.

Kernel logistic regression is not the only model that performs similarly to the support vector machine, replacing the hinge loss with any sensible loss function will give similar result, for example, the exponential loss function of boosting (Freund & Schapire, 1997), the squared error loss (e.g., Buhlmann & Yu, 2001; Zhang & Oles, 2001; Mannor *et al.*, 2002) and the $1/yf$ loss for distance weighted discrimination (Marron & Todd, 2002). These loss functions are all Bayes consistent. The negative binomial log-likelihood and the exponential loss are also margin-maximizing loss functions; but the squared error loss and the $1/yf$ loss are not.

To summarize, margin maximization is by nature a non-regularized objective, and solving it in high dimensional space is likely to lead to over-fitting and bad prediction performance. This has been observed in practice by many researchers, in particular Breiman (1999). Our conclusion is that margin maximization is not the only key to the support vector machine's success; the regularization term has played an important role.

Acknowledgments

We thank Saharon Rosset, Dylan Small, John Storey, Rob Tibshirani, and Jingming Yan for their helpful comments. We are also grateful for the three reviewers and one associate editor for their comments that helped improve the paper. Ji Zhu was partially supported by the Stanford Graduate Fellowship. Trevor Hastie is partially supported by grant DMS-9803645 from the National Science Foundation, and grant RO1-CA-72028-01 from the National Institutes of Health.

Appendix

Proof of Theorem 1

For the purpose of simple notation, we omit the constant β_0 in the proof. We define

$$G(\boldsymbol{\beta}) \equiv \sum_{i=1}^{n} \ln\left(1 + e^{-y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}}\right).$$

Lemma 3 *Consider the optimization problem (7)–(9), let the solution be denoted by $\hat{\boldsymbol{\beta}}(s)$. If the training data are separable, i.e. $\exists \boldsymbol{\beta}$, s.t. $y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta} > 0$, $\forall i$, then $y_i \mathbf{h}(\mathbf{x}_i)^T \hat{\boldsymbol{\beta}}(s) > 0$, $\forall i$ and $\|\hat{\boldsymbol{\beta}}(s)\|_2 = s$ for all $s > s_0$, where s_0 is a fixed positive number. Hence, $\left\| \frac{\hat{\boldsymbol{\beta}}(s)}{s} \right\|_2 = 1$.*

Proof

Suppose $\exists i^*$, *s.t.* $y_{i^*} \mathbf{h}(\mathbf{x}_{i^*})^T \boldsymbol{\beta} \leq 0$, then

$$\begin{aligned} G(\boldsymbol{\beta}) &\geq \ln\left(1 + e^{-y_{i^*} \mathbf{h}(\mathbf{x}_{i^*})^T \boldsymbol{\beta}}\right) && (40) \\ &\geq \ln 2. && (41) \end{aligned}$$

On the other hand, by the separability assumption, we know there exists $\boldsymbol{\beta}^*$, $\|\boldsymbol{\beta}^*\|_2 = 1$, *s.t.* $y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}^* > 0$, $\forall i$. Then for $s > s_0 = -\ln(2^{1/n} - 1) / \min_i \left(y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}^*\right)$, we have

$$\begin{aligned} G(s\boldsymbol{\beta}^*) &= \sum_{i=1}^{n} \ln\left(1 + e^{-y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}^* s}\right) && (42) \\ &< \sum_{i=1}^{n} \frac{\ln 2}{n} = \ln 2. && (43) \end{aligned}$$

Since $G(\hat{\boldsymbol{\beta}}(s)) \leq G(s\boldsymbol{\beta}^*)$, we have, for $s > s_0$, $y_i \mathbf{h}(\mathbf{x}_i)^T \hat{\boldsymbol{\beta}}(s) > 0$, $\forall i$.

For $s > s_0$, if $\|\hat{\boldsymbol{\beta}}(s)\|_2 < s$, we consider to scale up $\hat{\boldsymbol{\beta}}(s)$ by letting

$$\hat{\boldsymbol{\beta}}'(s) = \frac{\hat{\boldsymbol{\beta}}(s)}{\|\hat{\boldsymbol{\beta}}(s)\|_2} s.$$

Then $\|\hat{\boldsymbol{\beta}}'(s)\|_2 = s$, and

$$\begin{aligned} G\left(\hat{\boldsymbol{\beta}}'(s)\right) &= \sum_{i=1}^{n} \ln\left(1 + e^{-y_i \mathbf{h}(\mathbf{x}_i)^T \hat{\boldsymbol{\beta}}'(s)}\right) && (44) \\ &< \sum_{i=1}^{n} \ln\left(1 + e^{-y_i \mathbf{h}(\mathbf{x}_i)^T \hat{\boldsymbol{\beta}}(s)}\right) && (45) \\ &= G\left(\hat{\boldsymbol{\beta}}(s)\right), && (46) \end{aligned}$$

which is a contradiction. Hence $\|\hat{\boldsymbol{\beta}}(s)\|_2 = s$. □

Now we consider two separating candidates $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$, such that $\|\boldsymbol{\beta}_1\|_2 = \|\boldsymbol{\beta}_2\|_2 = 1$. Assume that $\boldsymbol{\beta}_1$ separates better, i.e.:

$$d_1 := \min_i y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}_1 > d_2 := \min_i y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}_2 > 0.$$

Lemma 4 *There exists some $s_0 = S(d_1, d_2)$ such that $\forall s > s_0$, $s\boldsymbol{\beta}_1$ incurs smaller loss than $s\boldsymbol{\beta}_2$, in other words:*

$$G(s\boldsymbol{\beta}_1) < G(s\boldsymbol{\beta}_2).$$

Proof

Let

$$s_0 = S(d_1, d_2) = \frac{\ln n + \ln 2}{d_1 - d_2},$$

then $\forall s > s_0$, we have

$$\begin{aligned}
\sum_{i=1}^{n} \ln\left(1 + e^{-y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}_1 s}\right) &\leq n \ln\left(1 + e^{-s \cdot d_1}\right) && (47)\\
&\leq n \exp(-s \cdot d_1) && (48)\\
&< \frac{1}{2} \exp(-s \cdot d_2) && (49)\\
&\leq \ln\left(1 + e^{-s \cdot d_2}\right) && (50)\\
&\leq \sum_{i=1}^{n} \ln\left(1 + e^{-y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}_2 s}\right). && (51)
\end{aligned}$$

The first and the last inequalities imply

$$G(s\boldsymbol{\beta}_1) < G(s\boldsymbol{\beta}_2).$$

□

Given these two lemmas, we can now prove that any convergence point of $\frac{\hat{\boldsymbol{\beta}}(s)}{s}$ must be a margin maximizing separator. Assume $\boldsymbol{\beta}^*$ is a convergence point of $\frac{\hat{\boldsymbol{\beta}}(s)}{s}$. Denote $d^* := \min_i y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}^*$. Since the training data are separable, clearly $d^* > 0$ (since otherwise $G(s\boldsymbol{\beta}^*)$ does not even converge to 0 as $s \to \infty$).

Now, assume some $\tilde{\boldsymbol{\beta}}$ with $\|\tilde{\boldsymbol{\beta}}\|_2 = 1$ has bigger minimal margin $\tilde{d} > d^*$. By continuity of the minimal margin in $\boldsymbol{\beta}$, there exists some open neighborhood of $\boldsymbol{\beta}^*$:

$$N_{\boldsymbol{\beta}^*} = \{\boldsymbol{\beta} : \|\boldsymbol{\beta} - \boldsymbol{\beta}^*\|_2 < \delta\},$$

and an $\epsilon > 0$, such that:

$$\min_i y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta} < \tilde{d} - \epsilon, \quad \forall \boldsymbol{\beta} \in N_{\boldsymbol{\beta}^*}.$$

Now by Lemma 4 we get that there exists some $s_0 = S(\tilde{d}, \tilde{d} - \epsilon)$ such that $s\tilde{\boldsymbol{\beta}}$ incurs smaller loss than $s\boldsymbol{\beta}$ for any $s > s_0,\ \boldsymbol{\beta} \in N_{\boldsymbol{\beta}^*}$. Therefore $\boldsymbol{\beta}^*$ cannot be a convergence point of $\frac{\hat{\boldsymbol{\beta}}(s)}{s}$.

We conclude that any convergence point of the sequence $\frac{\hat{\boldsymbol{\beta}}(s)}{s}$ must be a margin maximizing separator. If the margin maximizing separator is unique then it is the only possible convergence point, and therefore:

$$\lim_{s\to\infty} \frac{\hat{\boldsymbol{\beta}}(s)}{s} = \arg \max_{\|\boldsymbol{\beta}\|_2=1} \min_i y_i \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta}.$$

□

In the case that the margin maximizing separating hyperplane is not unique, this conclusion can easily be generalized to characterize a unique solution by defining tie-breakers: if the minimal margin is the same, then the second minimal margin determines which model separates better, and so on. Only in the case that the whole order statistics of the margins is common to many solutions can there really be more than one convergence point for $\frac{\hat{\boldsymbol{\beta}}(s)}{s}$.

References

[1] Bartlett, P. & Shawe-Taylor, J. (1999) Generalization performance of support vector machines and other pattern classifiers. In B. Schöelkopf, C. Burges & A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press.

[2] Bredensteiner, E. & Bennett, K. (1999) Multicategory classification by support vector machines. *Computational Optimization and Applications*, **12**, 35–46.

[3] Breiman, L. (1999) Prediction games and arcing algorithms. *Neural Computation*, **7**:1493-1517.

[4] Buhlmann, P. & Yu, B. (2001) Boosting with the L_2 loss: regression and classification. *Journal of American Statistical Association*. Accepted.

[5] Burges, C. (1998) A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery*, **2**. Kluwer Academic Publishers, Boston.

[6] Evgeniou, T., Pontil, M., & Poggio, T. (1999) Regularization networks and support vector machines. In A.J. Smola, P. Bartlett, B. Schölkopf & C. Schuurmans, editors, *Advances in Large Margin Classifiers.* MIT Press.

[7] Freund, Y. & Schapire, R. (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1):119-139.

[8] Green, P. & Yandell, B. (1985) Semi-parametric generalized linear models. *Proceedings 2nd International GLIM Conference*, Lancaster, Lecture notes in Statistics No. **32**, 44–55. Springer-Verlag, New York.

[9] Hastie, T. & Tibshirani, R. (1990) *Generalized Additive Models.* Chapman and Hall.

[10] Hastie, T., Tibshirani, R., & Friedman, J.(2001) *The elements of statistical learning.* Springer.

[11] Kaufman, L. (1999) Solving the quadratic programming problem arising in support vector classification. In B. Schölkopf, C. Burges & A. Smola, eds, *Advances in Kernel Methods - Support Vector Learning.* MIT Press.

[12] Keerthi, S., Duan, K., Shevade, S. & Poo, A. (2002) A fast dual algorithm for kernel logistic regression. *International Conference on Machine Learning*, **19**.

[13] Kimeldorf, G. & Wahba, G. (1971) Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.* **33**, 82–95.

[14] Lee, Y., Lin, Y. & Wahba, G. (2002) Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of American Statistical Association.* To appear.

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

[16] Lin, X., Wahba, G., Xiang, D., Gao, F., Klein, R. & Klein B. (2000), Smoothing spline ANOVA models for large data sets with Bernoulli observations and the randomized GACV. *Annals of Statistics*, **28**, 1570–1600.

[17] Luo, Z. & Wahba, G. (1997) Hybrid adaptive splines. *Journal of American Statistical Association*, **92**, 107–116.

[18] Mannor, S., Meir, R. & Zhang, T. (2002) The consistency of greedy algorithms for classification. In *Proceedings of the European Conference on Computational Learning Theory.*

[19] Marron, J. & Todd, M. (2002) Distance weighted discrimination. *Technical Report No. 1339.* School of Operations Research and Industrial Engineering, Cornell University.

[20] Platt, J. (1999) Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges & A. Smola, eds, *Advances in Kernel Methods - Support Vector Learning.* MIT Press.

[21] Rätsch, G., Onoda, T. & Müller, K. (2000) Soft margins for adaboost. *Machine Learning*, 1–35.

[22] Rosset, S., Zhu, J. & Hastie, T. (2004) Margin maximizing loss functions. *Neural Information Processing Systems*, **16**. To appear.

[23] Shawe-Taylor, J. & Cristianini, N. (1999) Margin distribution bounds on generalization. In *Proceedings of the European Conference on Computational Learning Theory.*

[24] Smola, A. & Schölkopf, B. (2000) Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning.* Morgan Kaufmann Publishers.

[25] Vapnik, V. (1995) *The Nature of Statistical Learning Theory.* Springer Verlag, Berlin.

[26] Vapnik, V. (1998) *Statistical Learning Theory.* Wiley, New York.

[27] Wahba, G. (1999) Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In B. Schölkopf, C. Burges & A. Smola, eds, *Advances in Kernel Methods - Support Vector Learning.* MIT Press.

[28] Wahba, G., Gu, C., Wang, Y., & Chappell, R. (1995) Soft classification, a.k.a. risk estimation, via penalized log-likelihood and smoothing spline analysis of variance. In D.H. Wolpert, editor, *The Mathematics of Generalization.* Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley Publisher.

[29] Weston, J. & Watkins, C. (1999) Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks.*

[30] Williams, C. & Seeger, M (2001) Using the Nystrom method to speed up kernel machines. In T. K. Leen, T. G. Diettrich & V. Tresp, editors, *Advances in Neural Information Processing Systems 13.* MIT Press.

[31] Zhang, T. & Oles, F. (2001) Text categorization based on regularized linear classification methods. *Information Retrieval*, **4**:5-31.

[32] Zhu, J. & Hastie, T. (2004) Classification of gene microarrays by penalized logistic regression. *Biostatistics.* In press.