Machine Learning, 46, 191–202, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

Support Vector Machines for Classification in Nonstandard Situations

YI LIN yilin@stat.wisc.edu
YOONKYUNG LEE yklee@stat.wisc.edu
GRACE WAHBA wahba@stat.wisc.edu
Department of Statistics, University of Wisconsin, Madison, 1210 West Dayton Street, Madison, WI 53706-1685, USA

Editor: Nello Cristianini

Abstract. The majority of classification algorithms are developed for the standard situation in which it is assumed that the examples in the training set come from the same distribution as that of the target population, and that the cost of misclassification into different classes are the same. However, these assumptions are often violated in real world settings. For some classification methods, this can often be taken care of simply with a change of threshold; for others, additional effort is required. In this paper, we explain why the standard support vector machine is not suitable for the nonstandard situation, and introduce a simple procedure for adapting the support vector machine methodology to the nonstandard situation. Theoretical justification for the procedure is provided. Simulation study illustrates that the modified support vector machine significantly improves upon the standard support vector machine in the nonstandard situation. The computational load of the proposed procedure is the same as that of the standard support vector machine. The procedure reduces to the standard support vector machine in the standard situation.

Keywords: support vector machine, classification, Bayes rule, *GCKL*, *GACV*

1. Introduction

In supervised learning, we are given a training data set of ℓ examples, and for each example $i, i = 1, 2, \ldots, \ell$, in the training set, we observe an input vector $\mathbf{x}_i \in R^n$, and a label y_i indicating one of several given classes to which the example belongs. We wish to learn a classification rule from the training set, so that we can assign a class label to any new subjects we encounter in the future. In this paper we confine ourselves to the binary classification problem: there are only two classes. This is a special but most common classification problem, and it is the problem for which the support vector machine (SVM) algorithm was originally developed. (The extension of the SVM to the multi-class classification problem is not straightforward, and deserves a complete treatment). Without loss of generality, we assume the two class labels are –1 and 1, and the associated classes are called negative class and positive class, respectively.

It is often the case that the class label is not uniquely determinable for a particular point $\mathbf{x}_i \in R^n$: Different subjects with the same input vector may belong to different classes. A probability model is needed for this situation, and most learning methods developed so far are designed under the following "standard" probability model:

The target population to which the classification rule will be applied in the future has an (unknown) probability distribution $P(\mathbf{x}, y)$. The examples in the training set are assumed to be an independently and identically distributed sample from the distribution $P(\mathbf{x}, y)$, or equivalently, they are independent random realizations of the random pair $(\mathbf{X}, Y)$ that has cumulative probability distribution $P(\mathbf{x}, y)$. The performance of any classification rule is measured by its expected misclassification rate on the target population, which means we assume the cost of different types of misclassification are the same.

Many real world situations, however, are nonstandard, in that the above model is not valid for those situations. The most common violations of the standard assumptions are:

1. Different types of misclassification may have different costs. One type of misclassification is often more serious than another. This should be taken into account when constructing the classification rules. In particular, the expected cost of future misclassification, rather than the expected misclassification rate, should be used to measure the performance of the classifier. One example for this is the diagnosis of disease. In such a situation misclassifying a diseased person as not having the disease may have a different consequence than misclassifying a healthy person as having the disease.
2. The target population may not have the same distribution as the one from which the training examples were (randomly) selected. One possible reason for this is the so called "population drift": the population evolves over time. It is impossible to give a general solution to deal with population drift, since the cause of the drift varies from problem to problem and there is no general way of modeling it. Another possible reason is the sampling bias: in some situations, the examples in the training set are sampled from in a way that is not completely random, but that makes sure the training set contains roughly the same number of examples from each class. Thus the smaller class is over-sampled and the bigger class down-sized in an attempt to have a more "balanced" sample. In this situation, a certain proportion of the examples are randomly selected from the positive class, and the rest of the examples are randomly selected from the negative class. These proportions do not reflect the actual proportions of positive and negative subjects in the target population. A case in point is the study of rare diseases. The proportion of people with a rare disease is usually a number that we know or have some information on, but the number is so small that if we took a random sample from the general population, the sample would contain very few diseased examples. One common approach then is to sample from the disease population and the healthy population separately. We can take a random sample of a pre-specified size from the disease population, and take another random sample of comparable size from the healthy population. The training data set we get is the combination of the two. Thus the proportions in the training set is different from that in the general population. Notice, however, we usually have information on what these proportions are.

Hand (1997) gives a nice explanation for these issues. Karakoulas and Shawe-Taylor (1999) provided an approach based on statistical learning theory. They considered the case that there exist hyperplanes that completely separate the data, and that costs for different misclassification are different. They provided insight into how to choose the hyperplane and threshold in such a situation. Brown et al. (2000) considered a nonstandard situation. However, they dealt with the problem in a different way than what is in this paper.

In this paper we will consider the nonstandard situation by taking unequal costs and sampling bias into account. Let the cost of false positive (a subject from the negative class assigned the positive label 1) be c^+, and the cost of false negative be c^-. The costs may or may not be equal. Later on we will see we do not require exact numbers for c^+ and c^-, what matters is the ratio of the two.

Again let the probability distribution of the target population be $P(\mathbf{x}, y)$. Let $g^+(\mathbf{x})$ be the probability density of $\mathbf{X}$ for the positive population, i.e., the conditional density of $\mathbf{X}$ given $Y = 1$. Let $g^-(\mathbf{x})$ be similarly defined. The unconditional ("prior") probabilities of the positive class and negative class in the target population are denoted by π^+ and π^- respectively. We assume we know the proportions π^+ and π^-. This is often the case as can be seen from the above examples.

Let $p(\mathbf{x}) = Pr(Y = 1 \mid \mathbf{X} = \mathbf{x})$ be the conditional probability of a randomly chosen subject from the target population belonging to the positive class given $\mathbf{X} = \mathbf{x}$. We obviously have $p(\mathbf{x}) = 1 - Pr(Y = -1 \mid \mathbf{X} = \mathbf{x})$. By Bayes formula, we have

$$p(\mathbf{x}) = \frac{\pi^+ g^+(\mathbf{x})}{\pi^+ g^+(\mathbf{x}) + \pi^- g^-(\mathbf{x})} \tag{1}$$

A classification rule is a mapping from R^n to $\{-1, 1\}$. That is, it assigns any input vector $\mathbf{x}$ a class label. When the costs for false positive and false negative are c^+ and c^-, the inaccuracy of any classification rule ϕ when applied to the target population is characterized by the expected (or average) cost

$$E\{c^+[1 - p(\mathbf{X})]1(\phi(\mathbf{X}) = 1) + c^- p(\mathbf{X})1(\phi(\mathbf{X}) = -1)\} \tag{2}$$

Here $1(\cdot)$ is the indicator function: it assumes the value 1 if its argument is true, and 0 if otherwise.

If we knew the function $p(\mathbf{x})$, we would be able to identify the optimal classification rule that minimizes (2). This optimal rule is called the Bayes rule and is given by

$$\phi_B(\mathbf{x}) = \begin{cases} +1 & \text{if } \dfrac{p(\mathbf{x})}{1 - p(\mathbf{x})} > \dfrac{c^+}{c^-} \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

Since in general we do not know $p(\mathbf{x})$, we have to learn from the training set a classification rule. As explained earlier, it is sometimes the case that the training examples are not a random sample from the target population: the examples are chosen in such way so that a pre-specified proportion of positive examples and negative examples are included in the sample, whereas within each class the examples are randomly chosen. Denote these pre-specified proportions for the training sample by π_s^+ and π_s^-. These may not match the population proportions π^+ and π^-. Let $(\mathbf{X}_s, Y_s)$ be a random pair whose distribution is the same as the distribution where the sample examples come from. Then, the conditional probability of a subject in the sample belonging to the positive class given $\mathbf{X}_s = \mathbf{x}$ is

$$p_s(\mathbf{x}) = \frac{\pi_s^+ g^+(\mathbf{x})}{\pi_s^+ g^+(\mathbf{x}) + \pi_s^- g^-(\mathbf{x})} \tag{4}$$

A common method of deriving classification rules is to approximate the Bayes rule by estimating the relevant quantities in (3) from the training sample. Since it is the quantities related to the training sample that can be directly estimated, it is helpful to re-express the Bayes rule in terms of $p_s(\mathbf{x})$ instead of $p(\mathbf{x})$. By (1), (3), (4), we get

$$\phi_B(\mathbf{x}) = \begin{cases} +1 & \text{if } \dfrac{p_s(\mathbf{x})}{1 - p_s(\mathbf{x})} > \dfrac{c^+}{c^-} \dfrac{\pi_s^+}{\pi_s^-} \dfrac{\pi^-}{\pi^+} \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

For notational purpose, define a function L on the two element set $\{-1, 1\}$:

$$L(-1) = c^+ \pi_s^+ \pi^- \quad \text{and} \quad L(1) = c^- \pi_s^- \pi^+. \tag{6}$$

Then the Bayes rule can be expressed as

$$\phi_B(\mathbf{x}) = sign\left[p_s(\mathbf{x}) - \frac{L(-1)}{L(-1) + L(1)} \right]. \tag{7}$$

Logistic regression and some other statistical methods estimate $p_s(\mathbf{x})$ from the training sample. To get an estimated Bayes rule in the standard situation of equal misclassification costs and no sampling bias, we just need to compare the estimated $p_s(\mathbf{x})$ with $1/2$. In the non-standard situation, we need to compare the estimated $p_s(\mathbf{x})$ with $L(-1)/(L(-1) + L(1))$. Notice in this case the estimation procedure is the same, all that is needed is a change of threshold after the estimate is obtained. This is not the case for other methods such as the (nonlinear) support vector machines. The standard support vector machine estimates $sign[p_s(\mathbf{x}) - 1/2]$, the Bayes rule in the standard situation. See Lin (1999). In the nonstandard situation, we can not estimate the Bayes rule (7) from the estimated $sign[p_s(\mathbf{x}) - 1/2]$. Therefore in the nonstandard case, we need to modify the estimation procedure of support vector machines to get an estimated Bayes rule.

2. The standard support vector machines

The support vector machine methodology was introduced in Boser, Guyon, and Vapnik (1992), and is receiving increasing attention in recent years. For a tutorial on support vector machines for classification, see Burges (1998). Here we give a brief summary of the standard support vector machines for classification, starting from the simple linear support vector machines and moving on to the nonlinear support vector machines.

When the two classes of points in the training set can be separated by a linear hyperplane, it is natural to use the hyperplane that separates the two groups of points in the training set by the largest margin. This amounts to the hard margin linear support vector machine: Find $\mathbf{w} \in R^n$, $b \in R$, to minimize $\|\mathbf{w}\|^2$, subject to

$$(\mathbf{x}_i \cdot \mathbf{w}) + b \geq +1 \quad \text{for } y_i = +1; \tag{8}$$

$$(\mathbf{x}_i \cdot \mathbf{w}) + b \leq -1 \quad \text{for } y_i = -1; \tag{9}$$

Once such $\mathbf{w}$ and b are found, our classification rule is $sign[(\mathbf{w} \cdot \mathbf{x}) + b]$.

When the points in the training data set are not linearly separable, constraints (8) and (9) can not be satisfied simultaneously. We can introduce nonnegative slack variables ξ's to overcome this difficulty, and this results in the soft margin linear support vector machine: Find $\mathbf{w} \in R^n$, $b \in R$, and ξ_i, $i = 1, 2, \ldots, \ell$, to minimize $(1/\ell)(\sum_{i=1}^{\ell} \xi_i)^q + \lambda\|\mathbf{w}\|^2$, under the constraints

$$(\mathbf{x}_i \cdot \mathbf{w}) + b \geq +1 - \xi_i \quad \text{for } y_i = +1; \tag{10}$$

$$(\mathbf{x}_i \cdot \mathbf{w}) + b \leq -1 + \xi_i \quad \text{for } y_i = -1; \tag{11}$$

$$\xi_i \geq 0, \quad \forall i.$$

Here λ is a parameter to be chosen by the user, and q is a positive integer. In the following we will concentrate on the case $q = 1$, since this is the most common situation. Notice (10) and (11) can be combined as

$$\xi_i \geq 1 - y_i[(\mathbf{x}_i \cdot \mathbf{w}) + b].$$

The nonlinear support vector machine maps the input variable into a high dimensional (often infinite dimensional) feature space, and applies the linear support vector machine in the feature space. Computationally, this can be achieved by the application of a (reproducing) kernel. For an introduction to reproducing kernels and reproducing kernel Hilbert spaces, see Wahba (1990). Also see Evgeniou, Pontil, and Poggio (1999) for an overview of the relation between support vector machines and the regularization problem. The nonlinear support vector machine with kernel K is equivalent to a regularization problem in the reproducing kernel Hilbert space (RKHS) H_K: Find $f(\mathbf{x}) = h(\mathbf{x}) + b$ with $h \in H_K$, $b \in R$, and ξ_i, $i = 1, 2, ..., \ell$, to minimize

$$\frac{1}{\ell}\left(\sum_{i=1}^{\ell} \xi_i\right) + \lambda\|h\|_{H_K}^2, \tag{12}$$

under the constraints

$$\xi_i \geq 1 - y_i f(\mathbf{x}_i), \tag{13}$$

$$\xi_i \geq 0, \ \forall i. \tag{14}$$

Once the solution $\hat{f}$ is found, we classify any new subject according to the sign of $\hat{f}$.

It is shown in Lin (1999) that, if the reproducing kernel Hilbert space is rich enough, (for example, when we use the Gaussian kernel or the spline kernel), the solution of the nonlinear support vector machine $\hat{f}$ approaches $sign(p_s - 1/2)$ as $\ell \to \infty$. This provides a reason why the nonlinear support vector machine works well in the standard situation, since $sign(p_s - 1/2)$ coincides with the Bayes rule of classification in the standard situation. In the more general nonstandard situation, however, the Bayes rule of classification for minimizing the expected cost in the target population is $sign[p_s - \frac{L(-1)}{L(-1)+L(1)}]$, which is in general different from $sign(p_s - 1/2)$. Hence the support vector machine introduced above will not perform optimally in the nonstandard situation. This is not clear in the current literature. In the nonstandard situation, $sign(p_s - 1/2)$ is actually the classification rule that minimizes the expected misclassification rate in the sampling population, and we will call it the "naive" Bayes rule.

3. Support vector machines for the nonstandard situation

In order to extend the support vector machine methodology to the nonstandard situation, we modify the (nonlinear) support vector machine (12) to minimizing

$$\frac{1}{\ell}\left(\sum_{i=1}^{\ell} L(y_i)\xi_i\right) + \lambda\|h\|^2_{H_K}, \tag{15}$$

under the constraints (13) and (14), where $L(y)$ is given by (6). That is, the slack variables are penalized according to different weights in the objective function. We remark that we do not need the exact values for $L(1)$ and $L(-1)$. What we really need is the ratio of the two.

Note the whole problem can be shown to be equivalent to the regularization problem of minimizing

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L(y_i)[(1 - y_i f(\mathbf{x}_i))_+] + \lambda\|h\|^2_{H_K} \tag{16}$$

over all the functions of the form $f(\mathbf{x}) = h(\mathbf{x}) + b$, with $h \in H_K$. Here $(\cdot)_+$ is a function such that τ_+ is τ, if $\tau > 0$; and is 0, otherwise. Regularization problems similar to this have long been studied in the statistics literature. Examples include penalized least square regression, penalized logistic regression, penalized density estimation, and regularization procedures used in more general nonlinear inverse problems. Cox and O'Sullivan (1990) provided a general framework for studying regularization methods. The method of regularization has two components: a data fit functional component and a regularization penalty component. The data fit functional component dictates that the estimate should follow the pattern in the data, whereas the regularization penalty component imposes smoothness conditions. The data fit component usually approaches a limiting functional as $\ell \to \infty$. In general the limiting functional can be used to identify the target function: the target function is the minimizer of the limiting functional. Under the assumption that the target function can be approximated by the elements in the RKHS under consideration and certain other general regularity conditions, the solution of the regularization problem approaches the target function as $\ell \to \infty$. For more discussion on this, see Lin (1999).

In our situation the limiting functional of the data fit component is $E[L(Y_s)(1 - Y_s f(\mathbf{X}_s))_+]$. The following lemma suggests that the solution of problem (16) approaches $sign[p_s - \frac{L(-1)}{L(-1)+L(1)}]$, the Bayes rule of classification in the nonstandard situation.

Lemma 3.1. *The minimizer of* $E[L(Y_s)(1 - Y_s f(\mathbf{X}_s))_+]$ *is* $sign[p_s - \frac{L(-1)}{L(-1)+L(1)}]$.

Proof: Notice

$$E\,[L(Y_s)(1 - Y_s f(\mathbf{X}_s))_+] = E\,\{E\,\{[L(Y_s)(1 - Y_s f(\mathbf{X}_s))_+] \,|\, \mathbf{X}_s\}\}$$

We can minimize $E\,[L(Y_s)(1 - Y_s f(\mathbf{X}_s))_+]$ by minimizing

$$E\,\{[L(Y_s)(1 - Y_s f(\mathbf{X}_s))_+] \,|\, \mathbf{X}_s = \mathbf{x}\}$$

for every fixed $\mathbf{x}$.

For any fixed $\mathbf{x}$, we have $E\{[L(Y_s)(1 - Y_s f(\mathbf{X}_s))_+] \,|\, \mathbf{X}_s = \mathbf{x}\} = p_s(\mathbf{x})L(1)[(1 - f(\mathbf{x}))_+] + (1 - p_s(\mathbf{x}))L(-1)[(1 + f(\mathbf{x}))_+]$. Let us search for $\bar{z}$ that minimizes $A(z) = p_s(\mathbf{x})L(1)\,[(1 - z)_+] + (1 - p_s(\mathbf{x}))L(-1)[(1 + z)_+]$.

First notice that the minimizer of $A(z)$ must be in $[-1, 1]$. For any z outside $[-1, 1]$, let $z' = sign(z)$, then z' is in $[-1, 1]$ and it is easy to check $A(z') < A(z)$. So we can restrict our search in $[-1, 1]$.

For $z \in [-1, 1]$, $A(z) = p_s(\mathbf{x})L(1)(1 - z) + [1 - p_s(\mathbf{x})]L(-1)(1 + z) = \{[1 - p_s(\mathbf{x})]L(-1) - p_s(\mathbf{x})L(1)\}z + p_s(\mathbf{x})L(1) + [1 - p_s(\mathbf{x})]L(-1)$. Therefore $A(z)$ is minimized at $z = 1$ when $[1 - p_s(\mathbf{x})]L(-1) - p_s(\mathbf{x})L(1) < 0$ and at $z = -1$ otherwise. That is, the minimizer is 1 if $p_s(\mathbf{x}) > L(-1)/[L(-1) + L(1)]$, and -1, otherwise. Thus the lemma is proved. □

The theory of reproducing kernel Hilbert space guarantees that the solution of (16) has the form $h(\cdot) = \sum_{i=1}^{\ell} c_i K(\mathbf{x}_i, \cdot)$. Letting $e = (1, \dots, 1)'$, $y = (y_1, y_2, \dots, y_\ell)'$, $c = (c_1, c_2, \dots, c_\ell)'$, and with some abuse of notation, letting $f = (f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_\ell))'$ and K now be the $\ell \times \ell$ matrix with ijth entry $K(\mathbf{x}_i, \mathbf{x}_j)$, we have

$$f = Kc + eb$$

We plug this into (15) and convert to the dual problem by introducing multipliers α_i and β_i for the constraints (13) and (14). Let Y be the $\ell \times \ell$ diagonal matrix with y_i in the iith position, let $H = \frac{1}{2\ell\lambda} YKY$, and let $L(y) = (L(y_1), L(y_2), \dots, L(y_\ell))'$. The Wolfe dual of (15) is equivalent to minimizing

$$\frac{1}{2}\alpha' H\alpha - e'\alpha \tag{17}$$

subject to

$$0 \le \alpha \le L(y), \quad \text{and} \quad \alpha' y = 0.$$

Once (17) is solved, we obtain c from the relation $c = \frac{1}{2\ell\lambda} Y\alpha$. Also, b can be found from any of the support vectors for which $0 < \alpha_i < L(y_i)$. In fact, we can derive from the Karush-Kuhn-Tucker condition that $b = y_i - \sum_{j=1}^{\ell} c_j K(\mathbf{x}_i, \mathbf{x}_j)$ for any i satisfying $0 < \alpha_i < L(y_i)$. The equation given below is a robust alternative to get a solution of b.

$$b = \frac{\sum_{i=1}^{\ell} \alpha_i (L(y_i) - \alpha_i)\big(y_i - \sum_{j=1}^{\ell} c_j K(\mathbf{x}_i, \mathbf{x}_j)\big)}{\sum_{i=1}^{\ell} \alpha_i (L(y_i) - \alpha_i)}$$

Note the whole computation procedure is very similar to that of the standard support vector machine: in that case $L(y)$ is replaced by e. As a result, any algorithm for implementing the standard support vector machine can be easily modified to carry out this new procedure.

4. Simulation

We use a simple simulation to illustrate the effectiveness of the modified support vector machine in a nonstandard situation. Consider a population consisting of two subpopulations. The positive subpopulation follows a bivariate normal distribution with mean $(0, 0)'$ and covariance matrix $diag(1, 1)$, whereas the negative subpopulation follows a bivariate normal with mean $(2, 2)'$ with covariance $diag(2, 1)$. The population is unbalanced: The positive and negative subpopulations account for 10% and 90% of the total population, respectively. Assume the cost of false negative is twice the cost of false positive. Notice in this simulation the Bayes rule that minimizes the expected cost in the target population is $sign[16 - 8x_2 - (x_1 + 2)^2 + 2 \ln \frac{8}{81}]$.

Suppose the distributions of the subpopulations are unknown to us, and suppose we have 400 examples available, 40% of which are randomly chosen from the positive subpopulation, and 60% of which are randomly chosen from the negative subpopulation. We apply the standard and modified support vector machine to derive classification rules based on the sample. We use Gaussian kernel $K(s, t) = \exp(-\frac{\|s-t\|^2}{2\sigma^2})$ in our simulation.

In order to be able to choose the smoothing parameters λ and σ, we randomly split the examples into a training set and a tuning set, each with 200 examples.

We first apply the standard support vector machine to derive a classification rule. The estimation is done with the training set and the smoothing parameters λ and σ are chosen by minimizing the misclassification rate in the tuning set. This is typical for the standard support vector machine.

We then apply the modified support vector machine. In this case $L(-1) = 0.36$, and $L(1) = 0.12$. We tune the smoothing parameters by minimizing

$$\frac{1}{200} \sum_{i=1}^{200} L(y_i') 1(y_i' f_{\lambda,\sigma}(\mathbf{x}_i') < 0),$$

where $(\mathbf{x}_i', y_i')$, $i = 1, 2, \ldots, 200$, are the examples in the tuning set, and $f_{\lambda,\sigma}$ is the solution of (15) with smoothing parameters λ and σ. Since the tuning set comes from the same population as the training set, the above is the empirical estimate based on the tuning set of

$$E[L(Y_s)1(Y_s f_{\lambda,\sigma}(\mathbf{X}_s) < 0)], \tag{18}$$

where $(\mathbf{X}_s, Y_s)$ is a random vector from the sampling distribution (the distribution of the population where the examples are drawn, which, as explained earlier, is different from the target population). It can be shown by direct calculation that minimizing (18) is equivalent to minimizing the expected cost in the target population given by

$$E\{c^+[1 - p(\mathbf{X})]1(f_{\lambda,\sigma}(\mathbf{X}) > 0) + c^- p(\mathbf{X})1(f_{\lambda,\sigma}(\mathbf{X}) < 0)\}, \tag{19}$$

where $(\mathbf{X}, Y)$ is a random vector from the target population.

We ran the simulation a number of times. The results are similar to the plots in figure 1. We can see in figure 1 the decision surface of our modified support vector machine is close to that of the Bayes rule minimizing the expected cost in the target population, whereas the

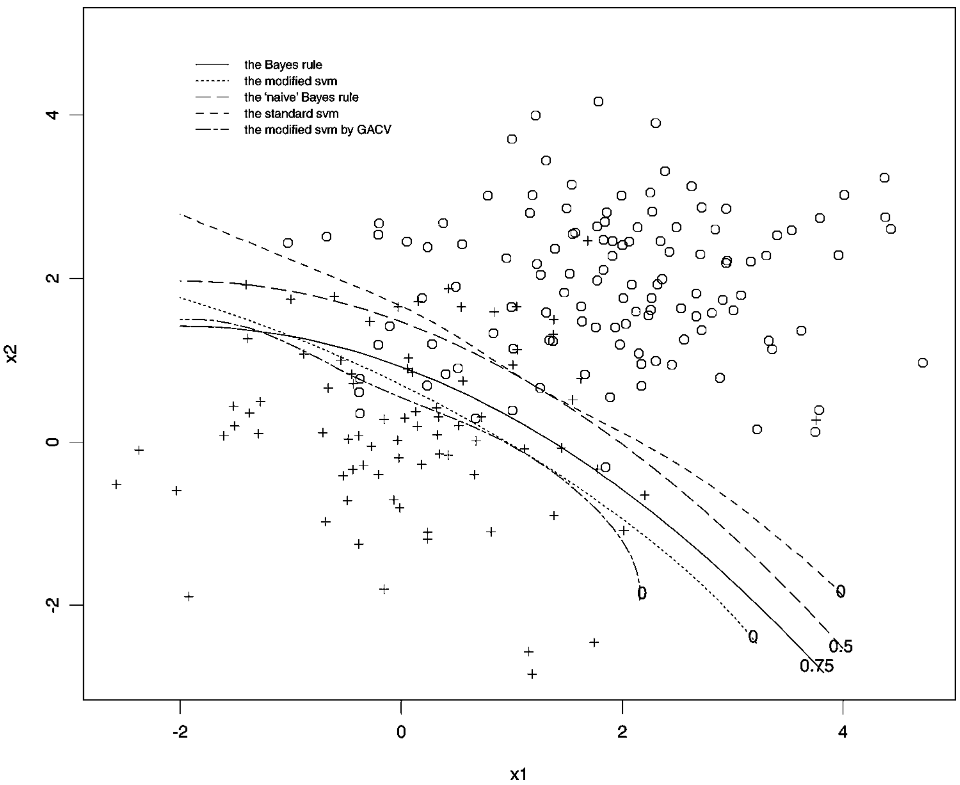

Figure 1. Decision surfaces given by the modified and standard support vector machines, the Bayes rule, and the “naive” Bayes rule. The modified support vector machine is implemented both with a tuning set and with the GACV.

decision surface of the standard support vector machine follows closely that of the "naive" Bayes rule minimizing the expected misclassification rate in the sampling population. The points in figure 1 are the examples in the training set.

In many situation it is desirable to be able to choose the smoothing parameters without a tuning set. Automated choice of the smoothing parameters in the standard support vector machine case has been considered by several authors, see, for example, Cristianini, Campbell, and Shawe-Taylor (1998), and Wahba, Lin, and Zhang (1999). Wahba, Lin, and Zhang (1999) considered the standard support vector machine and proposed choosing the smoothing parameters by minimizing the generalized approximate cross validation (*GACV*), which is a calculable proxy of the generalized comparative Kullback Leibler distance (*GCKL*). These quantities can be easily extended. In the nonstandard situation, we can define

$$GCKL(\lambda,\sigma) = E\{c^{+}[1-p(\mathbf{X})](1+f_{\lambda,\sigma}(\mathbf{X}))_{+} + c^{-}p(\mathbf{X})(1-f_{\lambda,\sigma}(\mathbf{X}))_{+}\},$$

where $(\mathbf{X}, Y)$ is a random pair from the target population.

It is easy to see the *GCKL* is an upper bound of the expected cost in the target population given by (19). But, to the extent that $f_{\lambda,\sigma}$ tends to $sign[p_s - \frac{L(-1)}{L(-1)+L(1)}]$, the *GCKL* tends to two times (19). Thus, when $f_{\lambda,\sigma}$ is close to $sign[p_s - \frac{L(-1)}{L(-1)+L(1)}]$, minimizing the *GCKL* is similar to minimizing the expected cost (19). In general, the global minimum of *GCKL* is easier to identify than the global minimum of (19), since the dependence of *GCKL* on $f_{\lambda,\sigma}$ is continuous and convex, while the dependence of (19) on $f_{\lambda,\sigma}$ is discontinuous and not convex. Our experience is that the *GCKL* usually has unique minimum, whereas (19) can have several local minima.

Direct calculation shows the *GCKL* has an equivalent form: Let $(\mathbf{X}_s, Y_s)$ be a random vector from the sampling distribution, then

$$GCKL(\lambda,\sigma) = \frac{1}{\pi_s^{+}\pi_s^{-}} E[L(Y_s)(1 - Y_s f_{\lambda,\sigma}(\mathbf{X}_s))_{+}]$$

When we choose the smoothing parameter with the *GCKL*, all we care about is the minimizer of the *GCKL*. Hence we can also take the *GCKL* as $E[L(Y_s)(1 - Y_s f_{\lambda,\sigma}(\mathbf{X}_s))_{+}]$.

A computable proxy of the *GCKL* is the generalized approximate cross validation (*GACV*), given by

$$GACV(\lambda,\sigma) = \frac{1}{\ell}\sum_{i=1}^{\ell} L(y_i)(1 - y_i f_{\lambda,\sigma}(\mathbf{x}_i))_{+} + \hat{D}(\lambda,\sigma)$$

where

$$\hat{D}(\lambda,\sigma) = \frac{1}{\ell}\left[2\sum_{y_i f_{\lambda,\sigma}(\mathbf{x}_i)<-1} L(y_i)\frac{\alpha_i}{2\ell\lambda}K(\mathbf{x}_i,\mathbf{x}_i) + \sum_{y_i f_{\lambda,\sigma}(\mathbf{x}_i)\in[-1,1]} L(y_i)\frac{\alpha_i}{2\ell\lambda}K(\mathbf{x}_i,\mathbf{x}_i)\right],$$

where α_i's are defined as in (17). The dependence of α_i's on the smoothing parameters are suppressed here to ease notation. For Gaussian kernel, we have $K(s, s) = 1$ for any s, and

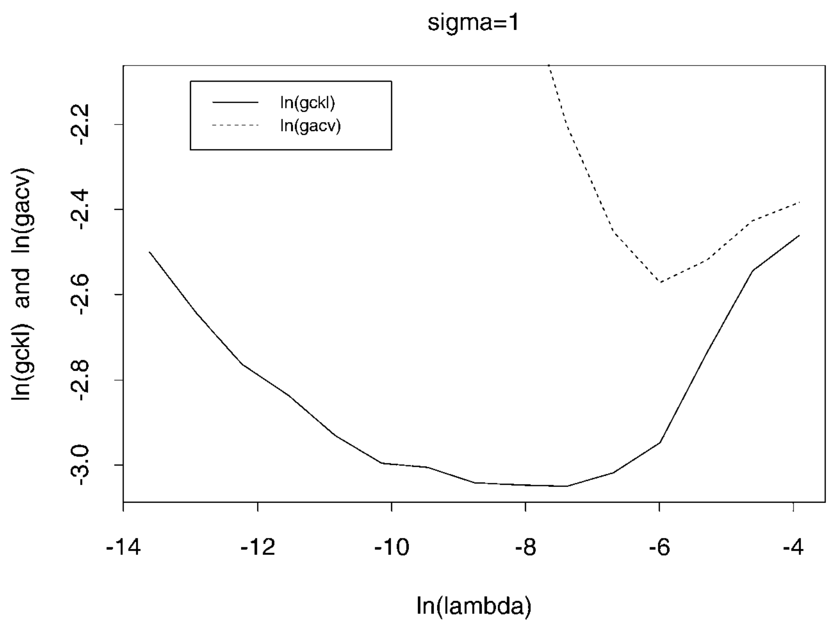

Figure 2. *GCKL* and *GACV* plot as a function of λ when σ is fixed at 1. We can see the minimizer of *GACV* is a decent estimate of the minimizer of *GCKL*.

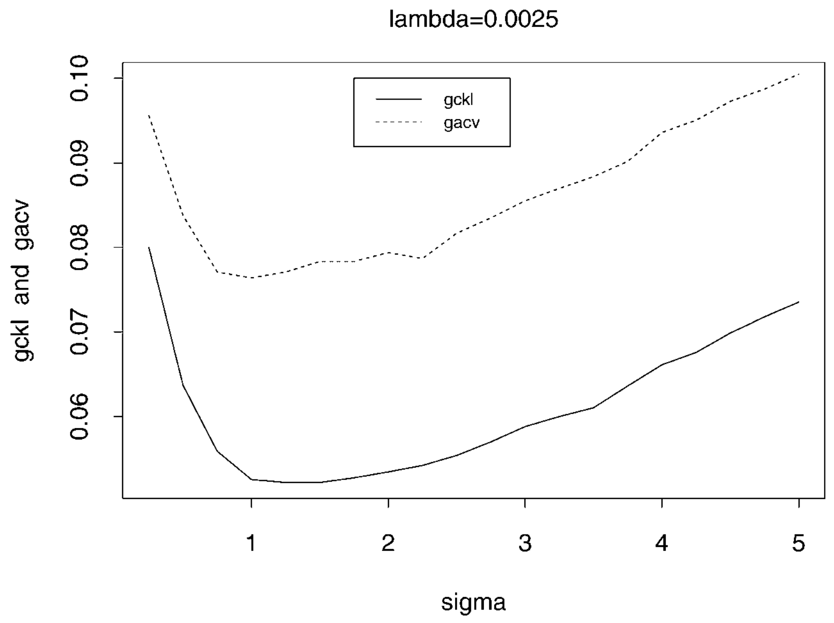

Figure 3. *GCKL* and *GACV* plot as a function of σ when λ is fixed at 0.0025. We can see the minimizer of *GACV* is a decent estimate of the minimizer of *GCKL*.

the expression of *GACV* can be simplified a bit. The derivation of the *GACV* as a proxy of the *GCKL* is parallel to that in Wahba, Lin, and Zhang (1999).

We use *GACV* to choose λ and σ jointly. For the training set in figure 1, the decision surface obtained by using *GACV* is also plotted in figure 1. We can see in this example the *GACV* does a decent job. Note we only need the training set for the whole estimation. Figures 2 and 3 show how *GACV* and *GCKL* vary with the λ and σ around the minimizing λ and σ. The λ and σ chosen by the *GACV* in this example are 0.0025 and 1 respectively. Limited experience with the *GACV* shows that it is a promising technique for picking out good choices of smoothing parameters. However, it seems to be quite variable and further investigation for possible improvement is in order.

Acknowledgment

This work was partly supported by Wisconsin Alumni Research Foundation, NSF under Grant DMS-9704758 and NIH under Grant R01 EY09946.

References

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In D. Haussler (Ed.), *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. Pittsburgh, PA: ACM Press.

Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., Ares, M., Jr., & Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences, 97:1*, 262–267.

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

Cox, D. D. & O'Sullivan, F. (1990). Asymptotic analysis of penalized likelihood and related estimates. *The Annals of Statistics, 18:4*, 1676–1695.

Cristianini, N., Campbell, C., & Shawe-Taylor, J. (1998). Dynamically adapting kernels in support vector machines. NeuroCOLT2 Technical Report Series, NC2-TR-1998-017.

Evgeniou, T., Pontil, M., & Poggio, T. (1999). A unified framework for regularization networks and support vector machines. Technical report, M.I.T. Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences.

Hand, D. J. (1997). *Construction and assessment of classification rules*. Chichester, England: John Wiley & Sons.

Karakoulas, G. & Shawe-Taylor, J. (1999). Optimizing classifiers for imbalanced training sets. In M. S. Kearns, S. A. Solla, & D. A. Cohn (Eds.), *Advances in neural information processing systems* (Vol. 11). Cambridge, MA: MIT Press.

Lin, Y. (1999). Support vector machines and the Bayes rule in classification. *Data Mining and Knowledge Discovery*, To appear.

Wahba, G. (1990). *Spline models for observational data*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Wahba, G., Lin, Y., & Zhang, H. (2000). *GACV* for support vector machines, or , another way to look at margin-like quantities. In A. J. Smola, P. Bartlett, B. Scholkopf, & D. Schurmans (Eds.), *Advances in large margin classifiers*. Cambridge, MA & London, England: MIT Press.

Received March 14, 2000
Revised March 7, 2001
Accepted March 8, 2001
Final manuscript July 20, 2001