12

Support Vector Machines and Flexible Discriminants

12.1 Introduction

In this chapter we describe generalizations of linear decision boundaries for classification. Optimal separating hyperplanes are introduced in Chapter 4 for the case when two classes are linearly separable. Here we cover extensions to the nonseparable case, where the classes overlap. These techniques are then generalized to what is known as the *support vector machine*, which produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space. The second set of methods generalize Fisher's linear discriminant analysis (LDA). The generalizations include *flexible discriminant analysis* which facilitates construction of nonlinear boundaries in a manner very similar to the support vector machines, *penalized discriminant analysis* for problems such as signal and image classification where the large number of features are highly correlated, and *mixture discriminant analysis* for irregularly shaped classes.

12.2 The Support Vector Classifier

In Chapter 4 we discussed a technique for constructing an *optimal* separating hyperplane between two perfectly separated classes. We review this and generalize to the nonseparable case, where the classes may not be separable by a linear boundary.

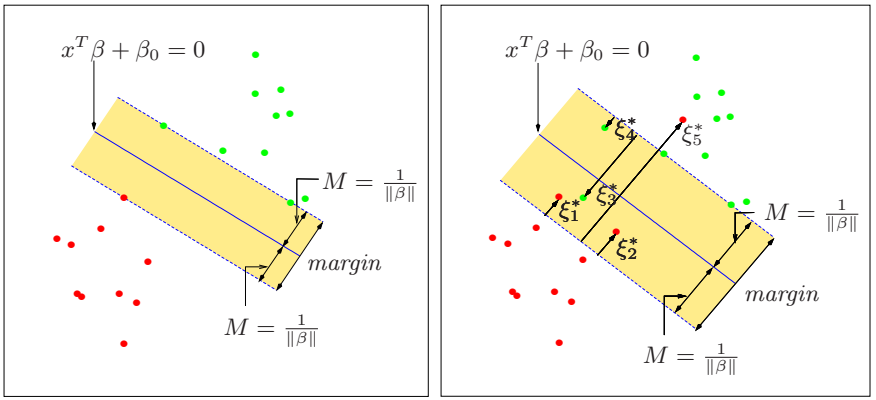


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

Our training data consists of N pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. Define a hyperplane by

$$\{x : \ f(x) = x^T\beta + \beta_0 = 0\}, \tag{12.1}$$

where β is a unit vector: $\|\beta\| = 1$. A classification rule induced by $f(x)$ is

$$G(x) = \text{sign}[x^T\beta + \beta_0]. \tag{12.2}$$

The geometry of hyperplanes is reviewed in Section 4.5, where we show that $f(x)$ in (12.1) gives the signed distance from a point x to the hyperplane $f(x) = x^T\beta + \beta_0 = 0$. Since the classes are separable, we can find a function $f(x) = x^T\beta + \beta_0$ with $y_i f(x_i) > 0 \ \forall i$. Hence we are able to find the hyperplane that creates the biggest *margin* between the training points for class 1 and -1 (see Figure 12.1). The optimization problem

$$\begin{aligned} &\max_{\beta, \beta_0, \|\beta\|=1} M \\ &\text{subject to } y_i(x_i^T\beta + \beta_0) \geq M, \ i = 1, \ldots, N, \end{aligned} \tag{12.3}$$

captures this concept. The band in the figure is M units away from the hyperplane on either side, and hence $2M$ units wide. It is called the *margin*.

We showed that this problem can be more conveniently rephrased as

$$\begin{aligned} &\min_{\beta, \beta_0} \|\beta\| \\ &\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1, \ i = 1, \ldots, N, \end{aligned} \tag{12.4}$$

where we have dropped the norm constraint on β. Note that $M = 1/\|\beta\|$. Expression (12.4) is the usual way of writing the support vector criterion for separated data. This is a convex optimization problem (quadratic criterion, linear inequality constraints), and the solution is characterized in Section 4.5.2.

Suppose now that the classes overlap in feature space. One way to deal with the overlap is to still maximize M, but allow for some points to be on the wrong side of the margin. Define the slack variables $\xi = (\xi_1, \xi_2, \ldots, \xi_N)$. There are two natural ways to modify the constraint in (12.3):

$$y_i(x_i^T\beta + \beta_0) \geq M - \xi_i, \tag{12.5}$$

or

$$y_i(x_i^T\beta + \beta_0) \geq M(1 - \xi_i), \tag{12.6}$$

$\forall i,\ \xi_i \geq 0,\ \sum_{i=1}^N \xi_i \leq$ constant. The two choices lead to different solutions. The first choice seems more natural, since it measures overlap in actual distance from the margin; the second choice measures the overlap in relative distance, which changes with the width of the margin M. However, the first choice results in a nonconvex optimization problem, while the second is convex; thus (12.6) leads to the "standard" support vector classifier, which we use from here on.

Here is the idea of the formulation. The value ξ_i in the constraint $y_i(x_i^T\beta + \beta_0) \geq M(1 - \xi_i)$ is the proportional amount by which the prediction $f(x_i) = x_i^T\beta + \beta_0$ is on the wrong side of its margin. Hence by bounding the sum $\sum \xi_i$, we bound the total proportional amount by which predictions fall on the wrong side of their margin. Misclassifications occur when $\xi_i > 1$, so bounding $\sum \xi_i$ at a value K say, bounds the total number of training misclassifications at K.

As in (4.48) in Section 4.5.2, we can drop the norm constraint on β, define $M = 1/\|\beta\|$, and write (12.4) in the equivalent form

$$\min \|\beta\| \quad \text{subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i\ \forall i, \\ \xi_i \geq 0,\ \sum \xi_i \leq \text{constant}. \end{cases} \tag{12.7}$$

This is the usual way the support vector classifier is defined for the nonseparable case. However we find confusing the presence of the fixed scale "1" in the constraint $y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i$, and prefer to start with (12.6). The right panel of Figure 12.1 illustrates this overlapping case.

By the nature of the criterion (12.7), we see that points well inside their class boundary do not play a big role in shaping the boundary. This seems like an attractive property, and one that differentiates it from linear discriminant analysis (Section 4.3). In LDA, the decision boundary is determined by the covariance of the class distributions and the positions of the class centroids. We will see in Section 12.3.3 that logistic regression is more similar to the support vector classifier in this regard.

12.2.1 Computing the Support Vector Classifier

The problem (12.7) is quadratic with linear inequality constraints, hence it is a convex optimization problem. We describe a quadratic programming solution using Lagrange multipliers. Computationally it is convenient to re-express (12.7) in the equivalent form

$$\begin{aligned} &\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{subject to} \quad \xi_i \geq 0,\ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i\ \forall i, \end{aligned} \tag{12.8}$$

where the "cost" parameter C replaces the constant in (12.7); the separable case corresponds to $C = \infty$.

The Lagrange (primal) function is

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i, \tag{12.9}$$

which we minimize w.r.t β, β_0 and ξ_i. Setting the respective derivatives to zero, we get

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i, \tag{12.10}$$

$$0 = \sum_{i=1}^{N}\alpha_i y_i, \tag{12.11}$$

$$\alpha_i = C - \mu_i,\ \forall i, \tag{12.12}$$

as well as the positivity constraints $\alpha_i,\ \mu_i,\ \xi_i \geq 0\ \forall i$. By substituting (12.10)–(12.12) into (12.9), we obtain the Lagrangian (Wolfe) dual objective function

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\alpha_i\alpha_{i'}y_i y_{i'}x_i^T x_{i'}, \tag{12.13}$$

which gives a lower bound on the objective function (12.8) for any feasible point. We maximize L_D subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N}\alpha_i y_i = 0$. In addition to (12.10)–(12.12), the Karush–Kuhn–Tucker conditions include the constraints

$$\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] = 0, \tag{12.14}$$

$$\mu_i\xi_i = 0, \tag{12.15}$$

$$y_i(x_i^T\beta + \beta_0) - (1-\xi_i) \geq 0, \tag{12.16}$$

for $i = 1, \ldots, N$. Together these equations (12.10)–(12.16) uniquely characterize the solution to the primal and dual problem.

From (12.10) we see that the solution for β has the form

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i, \tag{12.17}$$

with nonzero coefficients $\hat{\alpha}_i$ only for those observations i for which the constraints in (12.16) are exactly met (due to (12.14)). These observations are called the *support vectors*, since $\hat{\beta}$ is represented in terms of them alone. Among these support points, some will lie on the edge of the margin ($\hat{\xi}_i = 0$), and hence from (12.15) and (12.12) will be characterized by $0 < \hat{\alpha}_i < C$; the remainder ($\hat{\xi}_i > 0$) have $\hat{\alpha}_i = C$. From (12.14) we can see that any of these margin points ($0 < \hat{\alpha}_i$, $\hat{\xi}_i = 0$) can be used to solve for β_0, and we typically use an average of all the solutions for numerical stability.

Maximizing the dual (12.13) is a simpler convex quadratic programming problem than the primal (12.9), and can be solved with standard techniques (Murray et al., 1981, for example).

Given the solutions $\hat{\beta}_0$ and $\hat{\beta}$, the decision function can be written as

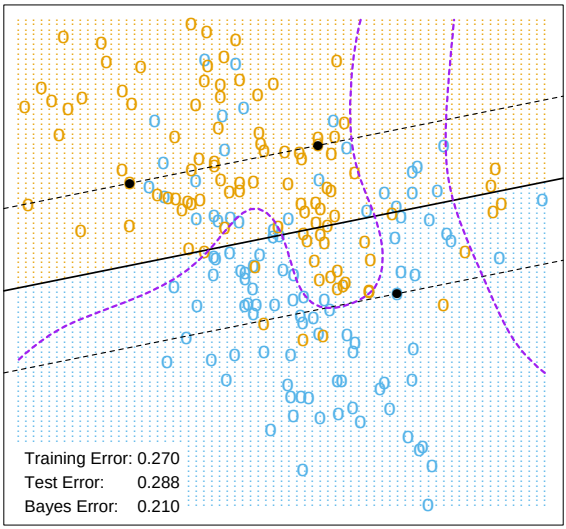
$$\begin{aligned} \hat{G}(x) &= \text{sign}[\hat{f}(x)] \\ &= \text{sign}[x^T\hat{\beta} + \hat{\beta}_0]. \end{aligned} \tag{12.18}$$

The tuning parameter of this procedure is the cost parameter C.

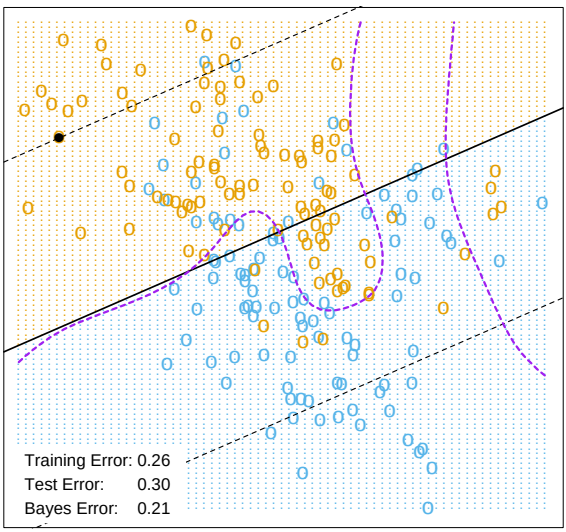
12.2.2 Mixture Example (Continued)

Figure 12.2 shows the support vector boundary for the mixture example of Figure 2.5 on page 21, with two overlapping classes, for two different values of the cost parameter C. The classifiers are rather similar in their performance. Points on the wrong side of the boundary are support vectors. In addition, points on the correct side of the boundary but close to it (in the margin), are also support vectors. The margin is larger for $C = 0.01$ than it is for $C = 10,000$. Hence larger values of C focus attention more on (correctly classified) points near the decision boundary, while smaller values involve data further away. Either way, misclassified points are given weight, no matter how far away. In this example the procedure is not very sensitive to choices of C, because of the rigidity of a linear boundary.

The optimal value for C can be estimated by cross-validation, as discussed in Chapter 7. Interestingly, the leave-one-out cross-validation error can be bounded above by the proportion of support points in the data. The reason is that leaving out an observation that is not a support vector will not change the solution. Hence these observations, being classified correctly by the original boundary, will be classified correctly in the cross-validation process. However this bound tends to be too high, and not generally useful for choosing C (62% and 85%, respectively, in our examples).



$C = 10000$



$C = 0.01$

FIGURE 12.2. *The linear support vector boundary for the mixture data example with two overlapping classes, for two different values of C. The broken lines indicate the margins, where $f(x) = \pm 1$. The support points ($\alpha_i > 0$) are all the points on the wrong side of their margin. The black solid dots are those support points falling exactly on the margin ($\xi_i = 0$, $\alpha_i > 0$). In the upper panel 62% of the observations are support points, while in the lower panel 85% are. The broken purple curve in the background is the Bayes decision boundary.*

12.3 Support Vector Machines and Kernels

The support vector classifier described so far finds linear boundaries in the input feature space. As with other linear methods, we can make the procedure more flexible by enlarging the feature space using basis expansions such as polynomials or splines (Chapter 5). Generally linear boundaries in the enlarged space achieve better training-class separation, and translate to nonlinear boundaries in the original space. Once the basis functions $h_m(x),\ m = 1, \ldots, M$ are selected, the procedure is the same as before. We fit the SV classifier using input features $h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i))$, $i = 1, \ldots, N$, and produce the (nonlinear) function $\hat{f}(x) = h(x)^T\hat{\beta} + \hat{\beta}_0$. The classifier is $\hat{G}(x) = \text{sign}(\hat{f}(x))$ as before.

The *support vector machine* classifier is an extension of this idea, where the dimension of the enlarged space is allowed to get very large, infinite in some cases. It might seem that the computations would become prohibitive. It would also seem that with sufficient basis functions, the data would be separable, and overfitting would occur. We first show how the SVM technology deals with these issues. We then see that in fact the SVM classifier is solving a function-fitting problem using a particular criterion and form of regularization, and is part of a much bigger class of problems that includes the smoothing splines of Chapter 5. The reader may wish to consult Section 5.8, which provides background material and overlaps somewhat with the next two sections.

12.3.1 Computing the SVM for Classification

We can represent the optimization problem (12.9) and its solution in a special way that only involves the input features via inner products. We do this directly for the transformed feature vectors $h(x_i)$. We then see that for particular choices of h, these inner products can be computed very cheaply.

The Lagrange dual function (12.13) has the form

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} \alpha_i\alpha_{i'}y_iy_{i'}\langle h(x_i), h(x_{i'})\rangle. \tag{12.19}$$

From (12.10) we see that the solution function $f(x)$ can be written

$$\begin{aligned} f(x) &= h(x)^T\beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i)\rangle + \beta_0. \end{aligned} \tag{12.20}$$

As before, given α_i, β_0 can be determined by solving $y_i f(x_i) = 1$ in (12.20) for any (or all) x_i for which $0 < \alpha_i < C$.

So both (12.19) and (12.20) involve $h(x)$ only through inner products. In fact, we need not specify the transformation $h(x)$ at all, but require only knowledge of the kernel function

$$K(x, x') = \langle h(x), h(x') \rangle \tag{12.21}$$

that computes inner products in the transformed space. K should be a symmetric positive (semi-) definite function; see Section 5.8.1.

Three popular choices for K in the SVM literature are

$$\begin{aligned} d\text{th-Degree polynomial: } & K(x, x') = (1 + \langle x, x' \rangle)^d, \\ \text{Radial basis: } & K(x, x') = \exp(-\gamma \|x - x'\|^2), \\ \text{Neural network: } & K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2). \end{aligned} \tag{12.22}$$

Consider for example a feature space with two inputs X_1 and X_2, and a polynomial kernel of degree 2. Then

$$\begin{aligned} K(X, X') &= (1 + \langle X, X' \rangle)^2 \\ &= (1 + X_1 X_1' + X_2 X_2')^2 \\ &= 1 + 2X_1 X_1' + 2X_2 X_2' + (X_1 X_1')^2 + (X_2 X_2')^2 + 2X_1 X_1' X_2 X_2'. \end{aligned} \tag{12.23}$$

Then $M = 6$, and if we choose $h_1(X) = 1$, $h_2(X) = \sqrt{2}X_1$, $h_3(X) = \sqrt{2}X_2$, $h_4(X) = X_1^2$, $h_5(X) = X_2^2$, and $h_6(X) = \sqrt{2}X_1 X_2$, then $K(X, X') = \langle h(X), h(X') \rangle$. From (12.20) we see that the solution can be written

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0. \tag{12.24}$$

The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there. A large value of C will discourage any positive ξ_i, and lead to an overfit wiggly boundary in the original feature space; a small value of C will encourage a small value of $\|\beta\|$, which in turn causes $f(x)$ and hence the boundary to be smoother. Figure 12.3 show two nonlinear support vector machines applied to the mixture example of Chapter 2. The regularization parameter was chosen in both cases to achieve good test error. The radial basis kernel produces a boundary quite similar to the Bayes optimal boundary for this example; compare Figure 2.5.

In the early literature on support vectors, there were claims that the kernel property of the support vector machine is unique to it and allows one to finesse the curse of dimensionality. Neither of these claims is true, and we go into both of these issues in the next three subsections.

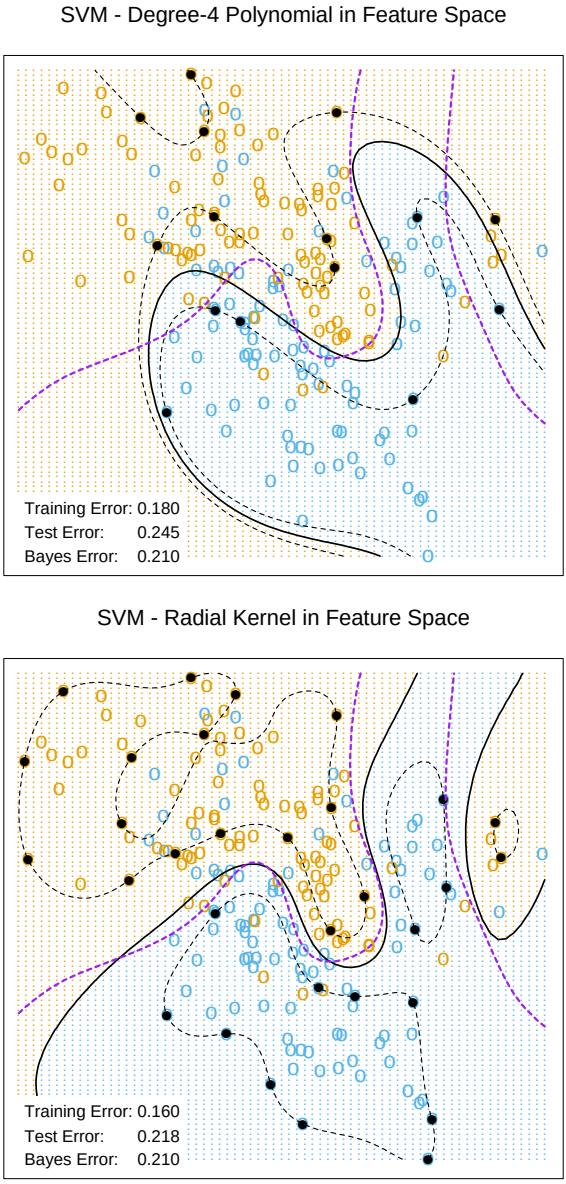


FIGURE 12.3. *Two nonlinear SVMs for the mixture data. The upper plot uses a 4th degree polynomial kernel, the lower a radial basis kernel (with $\gamma = 1$). In each case C was tuned to approximately achieve the best test error performance, and $C = 1$ worked well in both cases. The radial basis kernel performs the best (close to Bayes optimal), as might be expected given the data arise from mixtures of Gaussians. The broken purple curve in the background is the Bayes decision boundary.*

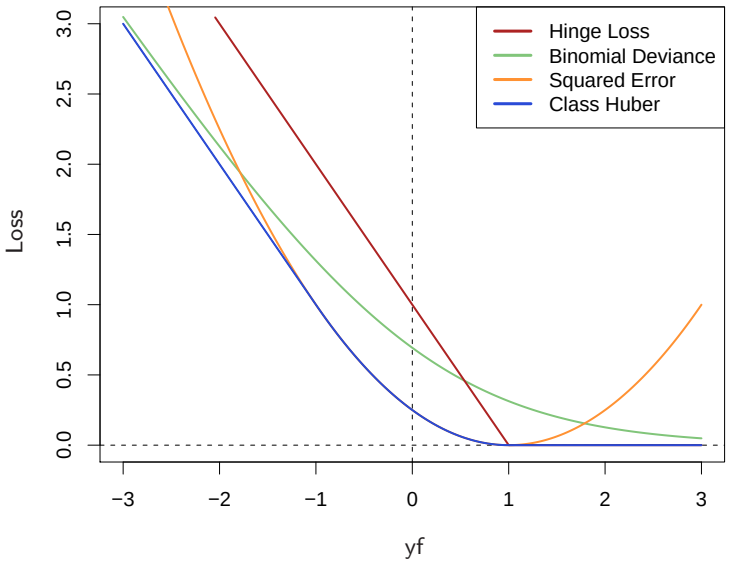


FIGURE 12.4. *The support vector loss function (hinge loss), compared to the negative log-likelihood loss (binomial deviance) for logistic regression, squared-error loss, and a "Huberized" version of the squared hinge loss. All are shown as a function of yf rather than f, because of the symmetry between the $y = +1$ and $y = -1$ case. The deviance and Huber have the same asymptotes as the SVM loss, but are rounded in the interior. All are scaled to have the limiting left-tail slope of -1.*

12.3.2 The SVM as a Penalization Method

With $f(x) = h(x)^T\beta + \beta_0$, consider the optimization problem

$$\min_{\beta_0,\,\beta} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2 \tag{12.25}$$

where the subscript "+" indicates positive part. This has the form *loss* + *penalty*, which is a familiar paradigm in function estimation. It is easy to show (Exercise 12.1) that the solution to (12.25), with $\lambda = 1/C$, is the same as that for (12.8).

Examination of the "hinge" loss function $L(y, f) = [1 - yf]_+$ shows that it is reasonable for two-class classification, when compared to other more traditional loss functions. Figure 12.4 compares it to the log-likelihood loss for logistic regression, as well as squared-error loss and a variant thereof. The (negative) log-likelihood or binomial deviance has similar tails as the SVM loss, giving zero penalty to points well inside their margin, and a

TABLE 12.1. *The population minimizers for the different loss functions in Figure 12.4. Logistic regression uses the binomial log-likelihood or deviance. Linear discriminant analysis (Exercise 4.2) uses squared-error loss. The SVM hinge loss estimates the mode of the posterior class probabilities, whereas the others estimate a linear transformation of these probabilities.*

Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log \frac{\Pr(Y = +1\|x)}{\Pr(Y = -1\|x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y = +1\|x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y = +1\|x) - 1$
"Huberised" Square Hinge Loss	$-4yf(x)$, $yf(x) < -1$ $[1 - yf(x)]_+^2$ otherwise	$f(x) = 2\Pr(Y = +1\|x) - 1$

linear penalty to points on the wrong side and far away. Squared-error, on the other hand gives a quadratic penalty, and points well inside their own margin have a strong influence on the model as well. The squared hinge loss $L(y, f) = [1 - yf]_+^2$ is like the quadratic, except it is zero for points inside their margin. It still rises quadratically in the left tail, and will be less robust than hinge or deviance to misclassified observations. Recently Rosset and Zhu (2007) proposed a "Huberized" version of the squared hinge loss, which converts smoothly to a linear loss at $yf = -1$.

We can characterize these loss functions in terms of what they are estimating at the population level. We consider minimizing $EL(Y, f(X))$. Table 12.1 summarizes the results. Whereas the hinge loss estimates the classifier $G(x)$ itself, all the others estimate a transformation of the class posterior probabilities. The "Huberized" square hinge loss shares attractive properties of logistic regression (smooth loss function, estimates probabilities), as well as the SVM hinge loss (support points).

Formulation (12.25) casts the SVM as a regularized function estimation problem, where the coefficients of the linear expansion $f(x) = \beta_0 + h(x)^T\beta$ are shrunk toward zero (excluding the constant). If $h(x)$ represents a hierarchical basis having some ordered structure (such as ordered in roughness),

then the uniform shrinkage makes more sense if the rougher elements h_j in the vector h have smaller norm.

All the loss-functions in Table 12.1 except squared-error are so called "margin maximizing loss-functions" (Rosset et al., 2004b). This means that if the data are separable, then the limit of $\hat{\beta}_\lambda$ in (12.25) as $\lambda \to 0$ defines the optimal separating hyperplane[1].

12.3.3 Function Estimation and Reproducing Kernels

Here we describe SVMs in terms of function estimation in reproducing kernel Hilbert spaces, where the kernel property abounds. This material is discussed in some detail in Section 5.8. This provides another view of the support vector classifier, and helps to clarify how it works.

Suppose the basis h arises from the (possibly finite) eigen-expansion of a positive definite kernel K,

$$K(x, x') = \sum_{m=1}^{\infty} \phi_m(x)\phi_m(x')\delta_m \tag{12.26}$$

and $h_m(x) = \sqrt{\delta_m}\phi_m(x)$. Then with $\theta_m = \sqrt{\delta_m}\beta_m$, we can write (12.25) as

$$\min_{\beta_0,\,\theta} \sum_{i=1}^{N} \left[1 - y_i(\beta_0 + \sum_{m=1}^{\infty} \theta_m \phi_m(x_i))\right]_+ + \frac{\lambda}{2} \sum_{m=1}^{\infty} \frac{\theta_m^2}{\delta_m}. \tag{12.27}$$

Now (12.27) is identical in form to (5.49) on page 169 in Section 5.8, and the theory of reproducing kernel Hilbert spaces described there guarantees a finite-dimensional solution of the form

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i K(x, x_i). \tag{12.28}$$

In particular we see there an equivalent version of the optimization criterion (12.19) [Equation (5.67) in Section 5.8.2; see also Wahba et al. (2000)],

$$\min_{\beta_0,\alpha} \sum_{i=1}^{N} (1 - y_i f(x_i))_+ + \frac{\lambda}{2}\alpha^T \mathbf{K}\alpha, \tag{12.29}$$

where $\mathbf{K}$ is the $N \times N$ matrix of kernel evaluations for all pairs of training features (Exercise 12.2).

These models are quite general, and include, for example, the entire family of smoothing splines, additive and interaction spline models discussed

[1]For logistic regression with separable data, $\hat{\beta}_\lambda$ diverges, but $\hat{\beta}_\lambda/\|\hat{\beta}_\lambda\|$ converges to the optimal separating direction.

in Chapters 5 and 9, and in more detail in Wahba (1990) and Hastie and Tibshirani (1990). They can be expressed more generally as

$$\min_{f\in\mathcal{H}} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \lambda J(f), \tag{12.30}$$

where $\mathcal{H}$ is the structured space of functions, and $J(f)$ an appropriate regularizer on that space. For example, suppose $\mathcal{H}$ is the space of additive functions $f(x) = \sum_{j=1}^{p} f_j(x_j)$, and $J(f) = \sum_j \int \{f''_j(x_j)\}^2 dx_j$. Then the solution to (12.30) is an additive cubic spline, and has a kernel representation (12.28) with $K(x, x') = \sum_{j=1}^{p} K_j(x_j, x'_j)$. Each of the K_j is the kernel appropriate for the univariate smoothing spline in x_j (Wahba, 1990).

Conversely this discussion also shows that, for example, *any* of the kernels described in (12.22) above can be used with *any* convex loss function, and will also lead to a finite-dimensional representation of the form (12.28). Figure 12.5 uses the same kernel functions as in Figure 12.3, except using the binomial log-likelihood as a loss function[2]. The fitted function is hence an estimate of the log-odds,

$$\begin{aligned} \hat{f}(x) &= \log \frac{\hat{\Pr}(Y = +1|x)}{\hat{\Pr}(Y = -1|x)} \\ &= \hat{\beta}_0 + \sum_{i=1}^{N} \hat{\alpha}_i K(x, x_i), \end{aligned} \tag{12.31}$$

or conversely we get an estimate of the class probabilities

$$\hat{\Pr}(Y = +1|x) = \frac{1}{1 + e^{-\hat{\beta}_0 - \sum_{i=1}^{N} \hat{\alpha}_i K(x, x_i)}}. \tag{12.32}$$

The fitted models are quite similar in shape and performance. Examples and more details are given in Section 5.8.

It does happen that for SVMs, a sizable fraction of the N values of α_i can be zero (the nonsupport points). In the two examples in Figure 12.3, these fractions are 42% and 45%, respectively. This is a consequence of the piecewise linear nature of the first part of the criterion (12.25). The lower the class overlap (on the training data), the greater this fraction will be. Reducing λ will generally reduce the overlap (allowing a more flexible f). A small number of support points means that $\hat{f}(x)$ can be evaluated more quickly, which is important at lookup time. Of course, reducing the overlap too much can lead to poor generalization.

[2] Ji Zhu assisted in the preparation of these examples.

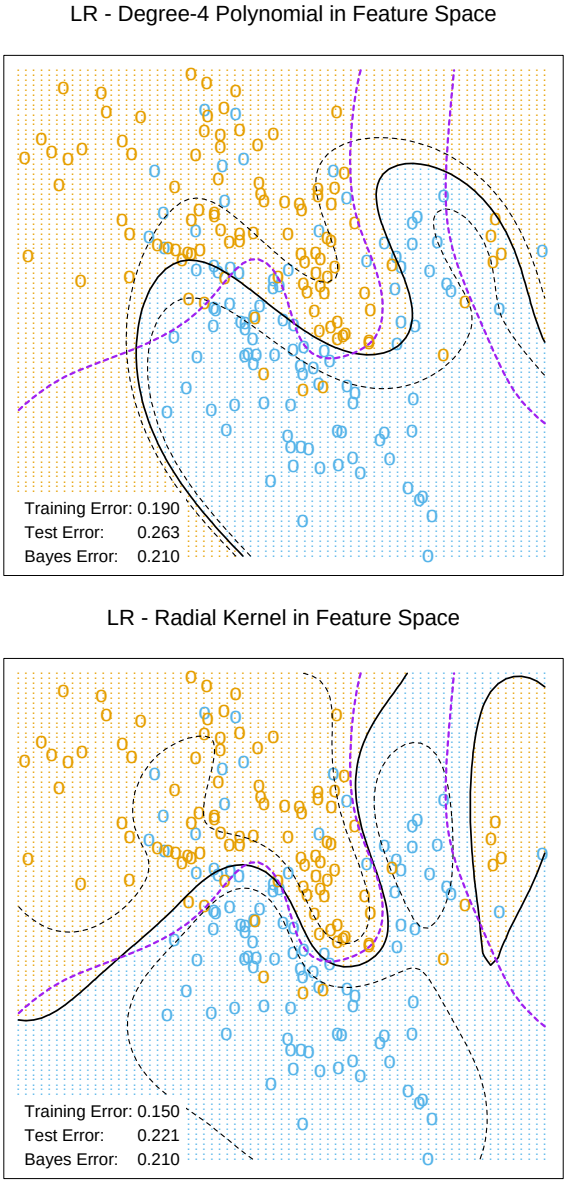


FIGURE 12.5. *The logistic regression versions of the SVM models in Figure 12.3, using the identical kernels and hence penalties, but the log-likelihood loss instead of the SVM loss function. The two broken contours correspond to posterior probabilities of* 0.75 *and* 0.25 *for the +1 class (or vice versa). The broken purple curve in the background is the Bayes decision boundary.*

TABLE 12.2. *Skin of the orange: Shown are mean (standard error of the mean) of the test error over 50 simulations. BRUTO fits an additive spline model adaptively, while MARS fits a low-order interaction model adaptively.*

	Method	Test Error (SE)	
		No Noise Features	Six Noise Features
1	SV Classifier	0.450 (0.003)	0.472 (0.003)
2	SVM/poly 2	0.078 (0.003)	0.152 (0.004)
3	SVM/poly 5	0.180 (0.004)	0.370 (0.004)
4	SVM/poly 10	0.230 (0.003)	0.434 (0.002)
5	BRUTO	0.084 (0.003)	0.090 (0.003)
6	MARS	0.156 (0.004)	0.173 (0.005)
	Bayes	0.029	0.029

12.3.4 SVMs and the Curse of Dimensionality

In this section, we address the question of whether SVMs have some edge on the curse of dimensionality. Notice that in expression (12.23) we are not allowed a fully general inner product in the space of powers and products. For example, all terms of the form $2X_jX'_j$ are given equal weight, and the kernel cannot adapt itself to concentrate on subspaces. If the number of features p were large, but the class separation occurred only in the linear subspace spanned by say X_1 and X_2, this kernel would not easily find the structure and would suffer from having many dimensions to search over. One would have to build knowledge about the subspace into the kernel; that is, tell it to ignore all but the first two inputs. If such knowledge were available a priori, much of statistical learning would be made much easier. A major goal of adaptive methods is to discover such structure.

We support these statements with an illustrative example. We generated 100 observations in each of two classes. The first class has four standard normal independent features X_1, X_2, X_3, X_4. The second class also has four standard normal independent features, but conditioned on $9 \leq \sum X_j^2 \leq 16$. This is a relatively easy problem. As a second harder problem, we augmented the features with an additional six standard Gaussian noise features. Hence the second class almost completely surrounds the first, like the skin surrounding the orange, in a four-dimensional subspace. The Bayes error rate for this problem is 0.029 (irrespective of dimension). We generated 1000 test observations to compare different procedures. The average test errors over 50 simulations, with and without noise features, are shown in Table 12.2.

Line 1 uses the support vector classifier in the original feature space. Lines 2–4 refer to the support vector machine with a 2-, 5- and 10-dimensional polynomial kernel. For all support vector procedures, we chose the cost parameter C to minimize the test error, to be as fair as possible to the

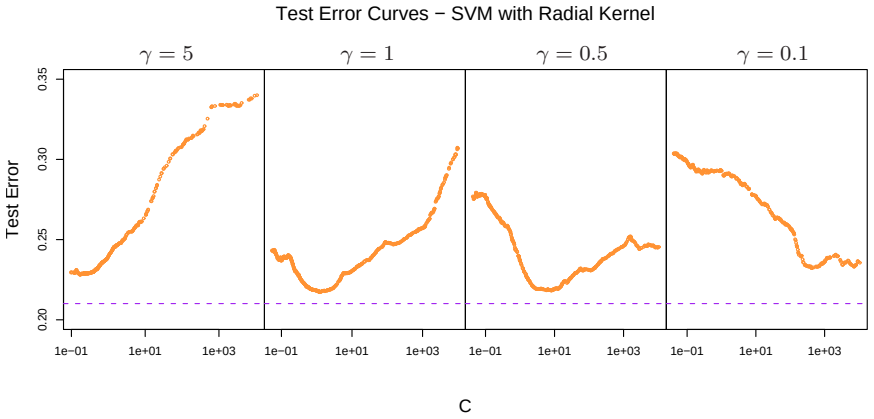


FIGURE 12.6. *Test-error curves as a function of the cost parameter C for the radial-kernel SVM classifier on the mixture data. At the top of each plot is the scale parameter γ for the radial kernel: $K_\gamma(x, y) = \exp(-\gamma||x - y||^2)$. The optimal value for C depends quite strongly on the scale of the kernel. The Bayes error rate is indicated by the broken horizontal lines.*

method. Line 5 fits an additive spline model to the $(-1, +1)$ response by least squares, using the BRUTO algorithm for additive models, described in Hastie and Tibshirani (1990). Line 6 uses MARS (multivariate adaptive regression splines) allowing interaction of all orders, as described in Chapter 9; as such it is comparable with the SVM/poly 10. Both BRUTO and MARS have the ability to ignore redundant variables. Test error was not used to choose the smoothing parameters in either of lines 5 or 6.

In the original feature space, a hyperplane cannot separate the classes, and the support vector classifier (line 1) does poorly. The polynomial support vector machine makes a substantial improvement in test error rate, but is adversely affected by the six noise features. It is also very sensitive to the choice of kernel: the second degree polynomial kernel (line 2) does best, since the true decision boundary is a second-degree polynomial. However, higher-degree polynomial kernels (lines 3 and 4) do much worse. BRUTO performs well, since the boundary is additive. BRUTO and MARS adapt well: their performance does not deteriorate much in the presence of noise.

12.3.5 A Path Algorithm for the SVM Classifier

The regularization parameter for the SVM classifier is the cost parameter C, or its inverse λ in (12.25). Common usage is to set C high, leading often to somewhat overfit classifiers.

Figure 12.6 shows the test error on the mixture data as a function of C, using different radial-kernel parameters γ. When $\gamma = 5$ (narrow peaked kernels), the heaviest regularization (small C) is called for. With $\gamma = 1$

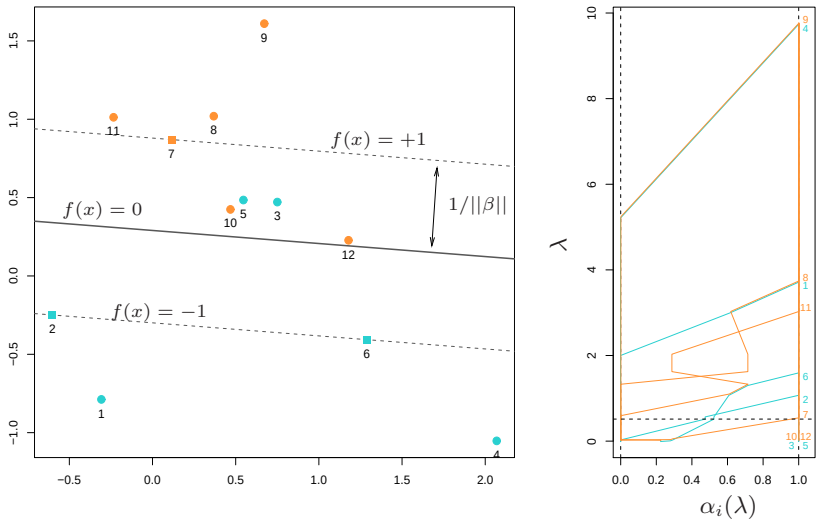


FIGURE 12.7. *A simple example illustrates the SVM path algorithm. (left panel:) This plot illustrates the state of the model at* $\lambda = 1/2$. *The "* $+1$*" points are orange, the "*-1*" blue. The width of the soft margin is* $2/||\beta|| = 2 \times 0.587$. *Two blue points* $\{3, 5\}$ *are misclassified, while the two orange points* $\{10, 12\}$ *are correctly classified, but on the wrong side of their margin* $f(x) = +1$*; each of these has* $y_i f(x_i) < 1$. *The three square shaped points* $\{2, 6, 7\}$ *are exactly on their margins. (right panel:) This plot shows the piecewise linear profiles* $\alpha_i(\lambda)$. *The horizontal broken line at* $\lambda = 1/2$ *indicates the state of the* α_i *for the model in the left plot.*

(the value used in Figure 12.3), an intermediate value of C is required. Clearly in situations such as these, we need to determine a good choice for C, perhaps by cross-validation. Here we describe a path algorithm (in the spirit of Section 3.8) for efficiently fitting the entire sequence of SVM models obtained by varying C.

It is convenient to use the loss+penalty formulation (12.25), along with Figure 12.4. This leads to a solution for β at a given value of λ:

$$\beta_\lambda = \frac{1}{\lambda} \sum_{i=1}^{N} \alpha_i y_i x_i. \tag{12.33}$$

The α_i are again Lagrange multipliers, but in this case they all lie in $[0, 1]$.

Figure 12.7 illustrates the setup. It can be shown that the KKT optimality conditions imply that the labeled points (x_i, y_i) fall into three distinct groups:

- Observations correctly classified and outside their margins. They have $y_i f(x_i) > 1$, and Lagrange multipliers $\alpha_i = 0$. Examples are the orange points 8, 9 and 11, and the blue points 1 and 4.
- Observations sitting on their margins with $y_i f(x_i) = 1$, with Lagrange multipliers $\alpha_i \in [0, 1]$. Examples are the orange 7 and the blue 2 and 6.
- Observations inside their margins have $y_i f(x_i) < 1$, with $\alpha_i = 1$. Examples are the blue 3 and 5, and the orange 10 and 12.

The idea for the path algorithm is as follows. Initially λ is large, the margin $1/||\beta_\lambda||$ is wide, and all points are inside their margin and have $\alpha_i = 1$. As λ decreases, $1/||\beta_\lambda||$ decreases, and the margin gets narrower. Some points will move from inside their margins to outside their margins, and their α_i will change from 1 to 0. By continuity of the $\alpha_i(\lambda)$, these points will *linger* on the margin during this transition. From (12.33) we see that the points with $\alpha_i = 1$ make fixed contributions to $\beta(\lambda)$, and those with $\alpha_i = 0$ make no contribution. So all that changes as λ decreases are the $\alpha_i \in [0, 1]$ of those (small number) of points on the margin. Since all these points have $y_i f(x_i) = 1$, this results in a small set of linear equations that prescribe how $\alpha_i(\lambda)$ and hence β_λ changes during these transitions. This results in piecewise linear paths for each of the $\alpha_i(\lambda)$. The breaks occur when points cross the margin. Figure 12.7 (right panel) shows the $\alpha_i(\lambda)$ profiles for the small example in the left panel.

Although we have described this for linear SVMs, exactly the same idea works for nonlinear models, in which (12.33) is replaced by

$$f_\lambda(x) = \frac{1}{\lambda} \sum_{i=1}^{N} \alpha_i y_i K(x, x_i). \tag{12.34}$$

Details can be found in Hastie et al. (2004). An `R` package `svmpath` is available on CRAN for fitting these models.

12.3.6 Support Vector Machines for Regression

In this section we show how SVMs can be adapted for regression with a quantitative response, in ways that inherit some of the properties of the SVM classifier. We first discuss the linear regression model

$$f(x) = x^T \beta + \beta_0, \tag{12.35}$$

and then handle nonlinear generalizations. To estimate β, we consider minimization of

$$H(\beta, \beta_0) = \sum_{i=1}^{N} V(y_i - f(x_i)) + \frac{\lambda}{2} \|\beta\|^2, \tag{12.36}$$

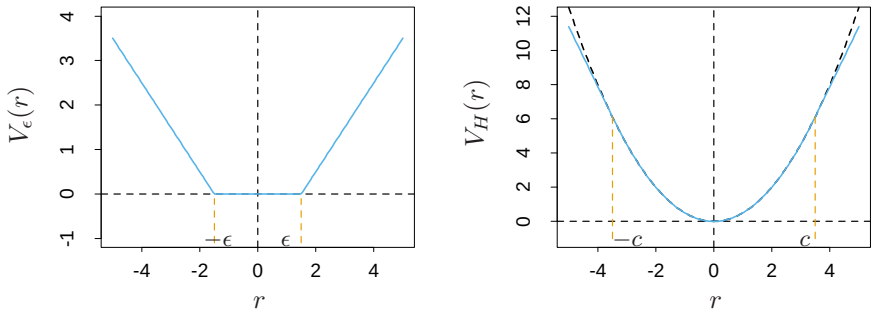


FIGURE 12.8. *The left panel shows the ϵ-insensitive error function used by the support vector regression machine. The right panel shows the error function used in Huber's robust regression (blue curve). Beyond $|c|$, the function changes from quadratic to linear.*

where

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise.} \end{cases} \tag{12.37}$$

This is an "ϵ-insensitive" error measure, ignoring errors of size less than ϵ (left panel of Figure 12.8). There is a rough analogy with the support vector classification setup, where points on the correct side of the decision boundary and far away from it, are ignored in the optimization. In regression, these "low error" points are the ones with small residuals.

It is interesting to contrast this with error measures used in robust regression in statistics. The most popular, due to Huber (1964), has the form

$$V_H(r) = \begin{cases} r^2/2 & \text{if } |r| \le c, \\ c|r| - c^2/2, & |r| > c, \end{cases} \tag{12.38}$$

shown in the right panel of Figure 12.8. This function reduces from quadratic to linear the contributions of observations with absolute residual greater than a prechosen constant c. This makes the fitting less sensitive to outliers. The support vector error measure (12.37) also has linear tails (beyond ϵ), but in addition it flattens the contributions of those cases with small residuals.

If $\hat{\beta}$, $\hat{\beta}_0$ are the minimizers of H, the solution function can be shown to have the form

$$\hat{\beta} = \sum_{i=1}^{N} (\hat{\alpha}_i^* - \hat{\alpha}_i) x_i, \tag{12.39}$$

$$\hat{f}(x) = \sum_{i=1}^{N} (\hat{\alpha}_i^* - \hat{\alpha}_i) \langle x, x_i \rangle + \beta_0, \tag{12.40}$$

where $\hat{\alpha}_i, \hat{\alpha}_i^*$ are positive and solve the quadratic programming problem

$$\min_{\alpha_i,\alpha_i^*} \epsilon \sum_{i=1}^{N}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) + \frac{1}{2}\sum_{i,i'=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_{i'}^* - \alpha_{i'})\langle x_i, x_{i'}\rangle$$

subject to the constraints

$$\begin{aligned} 0 \le \alpha_i,\ \alpha_i^* \le 1/\lambda, \\ \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0, \\ \alpha_i\alpha_i^* = 0. \end{aligned} \tag{12.41}$$

Due to the nature of these constraints, typically only a subset of the solution values $(\hat{\alpha}_i^* - \hat{\alpha}_i)$ are nonzero, and the associated data values are called the support vectors. As was the case in the classification setting, the solution depends on the input values only through the inner products $\langle x_i, x_{i'}\rangle$. Thus we can generalize the methods to richer spaces by defining an appropriate inner product, for example, one of those defined in (12.22).

Note that there are parameters, ϵ and λ, associated with the criterion (12.36). These seem to play different roles. ϵ is a parameter of the loss function V_ϵ, just like c is for V_H. Note that both V_ϵ and V_H depend on the scale of y and hence r. If we scale our response (and hence use $V_H(r/\sigma)$ and $V_\epsilon(r/\sigma)$ instead), then we might consider using preset values for c and ϵ (the value $c = 1.345$ achieves 95% efficiency for the Gaussian). The quantity λ is a more traditional regularization parameter, and can be estimated for example by cross-validation.

12.3.7 Regression and Kernels

As discussed in Section 12.3.3, this kernel property is not unique to support vector machines. Suppose we consider approximation of the regression function in terms of a set of basis functions $\{h_m(x)\}, m = 1, 2, \ldots, M$:

$$f(x) = \sum_{m=1}^{M} \beta_m h_m(x) + \beta_0. \tag{12.42}$$

To estimate β and β_0 we minimize

$$H(\beta, \beta_0) = \sum_{i=1}^{N} V(y_i - f(x_i)) + \frac{\lambda}{2}\sum \beta_m^2 \tag{12.43}$$

for some general error measure $V(r)$. For any choice of $V(r)$, the solution $\hat{f}(x) = \sum \hat{\beta}_m h_m(x) + \hat{\beta}_0$ has the form

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{a}_i K(x, x_i) \tag{12.44}$$

with $K(x,y) = \sum_{m=1}^{M} h_m(x)h_m(y)$. Notice that this has the same form as both the radial basis function expansion and a regularization estimate, discussed in Chapters 5 and 6.

For concreteness, let's work out the case $V(r) = r^2$. Let $\mathbf{H}$ be the $N \times M$ basis matrix with imth element $h_m(x_i)$, and suppose that $M > N$ is large. For simplicity we assume that $\beta_0 = 0$, or that the constant is absorbed in h; see Exercise 12.3 for an alternative.

We estimate β by minimizing the penalized least squares criterion

$$H(\beta) = (\mathbf{y} - \mathbf{H}\beta)^T(\mathbf{y} - \mathbf{H}\beta) + \lambda\|\beta\|^2. \tag{12.45}$$

The solution is

$$\hat{\mathbf{y}} = \mathbf{H}\hat{\beta} \tag{12.46}$$

with $\hat{\beta}$ determined by

$$-\mathbf{H}^T(\mathbf{y} - \mathbf{H}\hat{\beta}) + \lambda\hat{\beta} = 0. \tag{12.47}$$

From this it appears that we need to evaluate the $M \times M$ matrix of inner products in the transformed space. However, we can premultiply by $\mathbf{H}$ to give

$$\mathbf{H}\hat{\beta} = (\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1}\mathbf{H}\mathbf{H}^T\mathbf{y}. \tag{12.48}$$

The $N \times N$ matrix $\mathbf{H}\mathbf{H}^T$ consists of inner products between pairs of observations i, i'; that is, the evaluation of an inner product kernel $\{\mathbf{H}\mathbf{H}^T\}_{i,i'} = K(x_i, x_{i'})$. It is easy to show (12.44) directly in this case, that the predicted values at an arbitrary x satisfy

$$\begin{aligned} \hat{f}(x) &= h(x)^T\hat{\beta} \\ &= \sum_{i=1}^{N} \hat{\alpha}_i K(x, x_i), \end{aligned} \tag{12.49}$$

where $\hat{\alpha} = (\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1}\mathbf{y}$. As in the support vector machine, we need not specify or evaluate the large set of functions $h_1(x), h_2(x), \ldots, h_M(x)$. Only the inner product kernel $K(x_i, x_{i'})$ need be evaluated, at the N training points for each i, i' and at points x for predictions there. Careful choice of h_m (such as the eigenfunctions of particular, easy-to-evaluate kernels K) means, for example, that $\mathbf{H}\mathbf{H}^T$ can be computed at a cost of $N^2/2$ evaluations of K, rather than the direct cost N^2M.

Note, however, that this property depends on the choice of squared norm $\|\beta\|^2$ in the penalty. It does not hold, for example, for the L_1 norm $|\beta|$, which may lead to a superior model.