Chapter 7
Support Vector Machines

Support vector machines (SVM) was developed as a type of classifiers, largely in computer science, with its own set of research questions, conceptual frameworks, technical language, and culture. A substantial amount of the initial interest in support vector machines stemmed from the important theoretical work surrounding it (Vapnick 1996). For many, that remains very attractive.

The early applications of SVM were not especially compelling. But, over the past decade, the applications to which support vector machines have been applied have broadened (Christianini and Shawe-Taylor 2000; Moguerza and Munõz 2006; Ma and Gao 2014), available software has responded (Joachims 1998; Chen et al. 2004; Hsu et al. 2010; Karatzoglou et al. 2015), and relationships between support vector machines and other forms of machine learning have become better understood (Bishop 2006: Chaps. 6 and 7; Hastie et al. 2009: 417–437). SVM has joined a mainstream of many machine/statistical learning procedures. It incorporates some unique features to be sure, but many familiar features as well. In practice, SVM can be seen as a worthy competitor to random forests and boosting.

This chapter will draw heavily on material covered in earlier chapters. In particular, regression kernels, discussed in Chap. 2, will make an important encore appearance. Much of the earlier material addressing why boosting works so well also will carry over, at least in broad brush strokes. Support vector machines can be understood in part as a special kind of margin maximizer and in part as a loss function optimizer with an unusual loss function.

Different expositions of support vector machines often use rather different notation. In particular, the notational practices of computer science and statistics will rarely correspond. For example, the excellent treatment of support vector machines by Bishop (2006: Chap. 7) and the equally excellent treatment of support vector machines by Hastie and his colleagues (2009: Chap. 12) are difficult to compare without first being able to map one notional scheme on to the other. In this chapter,

---

The original version of this chapter was revised: See the "Chapter Note" section at the end of this chapter for details. The erratum to this chapter is available at https://doi.org/10.1007/978-3-319-44048-4_10.

R.A. Berk, *Statistical Learning from a Regression Perspective*,
Springer Texts in Statistics, DOI 10.1007/978-3-319-44048-4_7

the notation of Hastie and colleagues will be used by and large because it corresponds better to the notation used in earlier chapters.

7.1 Support Vector Machines in Pictures

Support vector machines has more demanding mathematical underpinnings than boosting or random forests. In some ways, it is another form of penalized regression. But before we get to a technical discussion, let's take a look at several figures that will make the key ideas accessible.

7.1.1 The Support Vector Classifier

Suppose there is a binary response variable coded, as is often done in boosting, as 1 and -1. There is also a $f(x)$, where x is a vector of one or more predictors. The function can be written in a familiar linear manner as

$$f(x) = \beta_0 + x^T\beta. \tag{7.1}$$

Equation 7.1 is essentially a linear regression with a binary outcome of 1 or -1 and no restrictions in practice on what numeric values the function yields. The $f(x)$ might be .6 for one observation, -1.2 for another observations, 2.1 for another observation, and so on. More is needed for classification. If $f(x)$ is a positive number, the label 1 is assigned to an observation. If $f(x)$ is a negative number, the label -1 is assigned to an observation. One can then compare the 1s and -1s from the function to the 1s and -1s of the response variable. The problem to be tackled in the pages ahead is how to make the two sets of 1s and -1s correspond as much as possible, not just in the data on hand, but in new realizations of the data. The task is to produce accurate classifications, which has been a major theme of past chapters. But the way SVM goes about this is novel. We begin with the support vector classifier.[1]

Figure 7.1 shows a three-dimensional scatter plot much like those used in earlier chapters. As before, there are two predictors (X and Z) and a binary response Y, that can take on values of red or blue. Red might represent dropping out of school, and blue might represent graduating. (Blue could be coded as 1, and red could be coded as -1.) The two predictors might be reading grade level and the number of truancies per semester. In this figure, the blue circles and red circles are each located in quite different areas of the two-dimensional space defined by the predictors. In fact, there is lots of daylight between the two groups, and a linear decision boundary easily could be drawn to produce perfect homogeneity. In SVM language, a linear

[1] In the SVM literature, the response variable is often called the "target variable," and the intercept in Eq. 7.1 is often called the "bias." Each observation is sometimes called an "example."

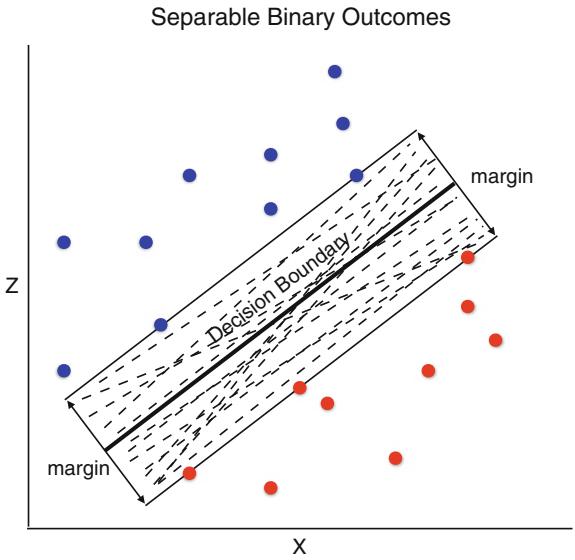

Fig. 7.1 A support vector classifier with two predictors X and Z and two linearly separable classes shown as *Red* or *Blue*

separating hyperplane could be drawn to produce separation between the two classes. More such SVM language will be introduced as we proceed.

In Fig. 7.1, there is a limitless number of linear decision boundaries producing separation. These are represented by the dashed lines in Fig. 7.1. Ideally, there is a way to find the best linear decision boundary.

Enter the support vector classifier. When there is separation, the support vector classifier solves the problem of which line to overlay by constructing two parallel lines on either side of, and the same distance from, the decision boundary. The two lines are placed as far apart as possible without including any observations within the space between them. One can think of the two lines as fences defining a buffer zone. In other words, the support vector classifier seeks two parallel fences that maximize their perpendicular distance from the decision boundary. There can be only one straight line parallel to the fences and midway between them. That decision boundary is shown with the solid black line.

Observations can fall right on either fence but not on their wrong sides. Here, there are no blue circles below the upper fence and no red circles above the lower fence. Observations that fall on top of the fences are called "support vectors" because they directly determine where the fences will be located and hence, the optimal decision boundary. In Fig. 7.1, there are two blue support vectors and three red support vectors.

In Fig. 7.1, the total width of the buffer zone is shown with the two double-headed arrows. The distance between the decision boundary and either fence is called the "margin," although some define the margin as the distance between the two fences (which amounts to the same thing). The wider the margin, the greater the separation between the two classes. Although formally the margin for a support vector classifier differs from the margins used by boosting and random forests, larger margins remain desirable because generalization error will usually be smaller.

Classification follows directly. Cases that fall on one side of the decision boundary are labeled as one class, and cases that fall on the other side of the decision boundary are labeled as the other class. Subsequently, any new cases for which the outcome class is not known will be assigned the class determined by the side of the decision boundary on which they fall. And that location will be a function of X and Z. The classification rule that follows from the decision boundary is called "hard thresholding," and the decision boundary is often called the "separating hyperplane." Sometimes the two fences are called the "margin boundary."

The data shown in Fig. 7.1 are very cooperative, and such cooperation is in practice rare. Figure 7.2 shows a plot that is much like Fig. 7.1, but the two sets of values are no longer linearly separable. Three blue circles and the two red circles violate their margin boundaries. They are on the wrong side of their respective buffer zone fences with each distance represented by an arrow. Moreover, there is no way to relocate and/or narrow the buffer zone so that there is a separating hyperplane able to partition the space into two perfectly homogeneous regions. There is no longer any linear solution to the classification problem.

One possible response is to permit violations of the buffer zone. One can specify some number of the observations that would be allowed to fall on the wrong side of their margin boundary. These are called "slack variables." One can try to live with a result that looks a lot like Fig. 7.2. The idea might be to maximize the width of the buffer zone conditional on the slack variables.

But that is not quite enough. Some slack variables fall just across their margin boundary, and some fall far away. In response, the distance between the relevant fence and the location of the slack variable can be taken into account. The sum of such distances can be viewed as a measure of how permissive one has been when

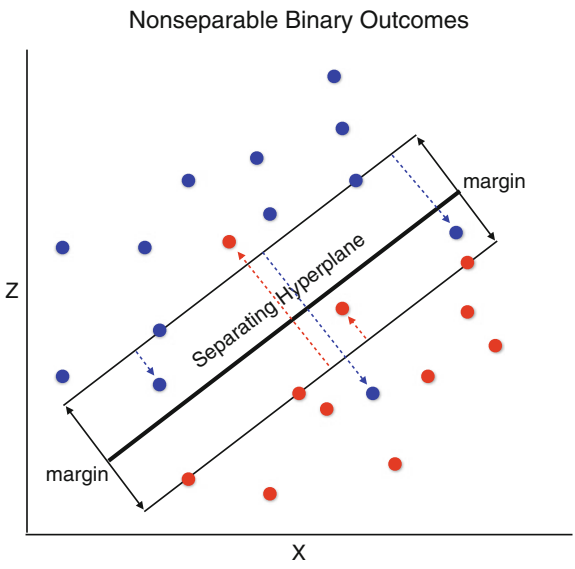

Fig. 7.2 A support vector classifier with predictors X and Z when there are two classes that are not linearly separable

the margin is maximized. If one is more permissive by allowing for a larger sum, it may be possible to locate a separating hyperplane within a larger margin. Again, larger margins are good. More stable classifications can follow. But more permissive solutions imply more bias because misclassifications will be introduced. A form of the bias-variance tradeoff reappears. It follows that the sum of the distances can be a tuning parameter when a support vector classifier is applied to data. Fitting the support vector classifier with slack variables is sometimes called "soft thresholding."

7.1.2 Support Vector Machines

There is a complementary solution to classification problems when the classes are not linearly separable. One can allow for a nonlinear decision boundary in the existing predictor space by fitting a separating hyperplane in higher dimensions. We introduced this idea in Chap. 1 when linear basis expansions were discussed, and we elaborated on it in Chap. 2 when regression kernels were considered in some depth. Support vector classifiers become support vector machines when a kernel replaces a conventional set of predictors. However, the use of kernels is not straightforward. As already noted, there can be several kernel candidates with no formal guidance on which one to choose. In addition, kernels come with tuning parameters whose values usually have to be determined empirically. Finally, recall that kernel results are scale dependent (and normalizing papers over the problem) with categorical predictors a major complication.

In summary, support vector machines estimate the coefficients in Eq. 7.1 by finding a separating hyperplane producing the maximum margin, subject to a constraint on the sum of the slack variable distances. With those estimates in hand, fitted values are produced. Positive fitted values are assigned a class of 1, and negative fitted values are assigned a class of -1.

7.2 Support Vector Machines More Formally

With the main conceptual foundations of support vector machines addressed, we turn briefly to a somewhat more formal approach. To read the literature about support vector machines, some familiarity for the underlying mathematics and notation is essential. What follows draws heavily on Hastie and his colleagues (2009: 417–438) and on Bishop (2007: Chaps. 6 and 7).

7.2.1 The Support Vector Classifier Again: The Separable Case

There are N observations in the training data. Each observation has a value for each of p predictors and a value for the response. A response is coded 1 or -1. The separating hyperplane is defined by a conventional linear combination of predictors as

$$f(x) = \beta_0 + x^T\beta = 0. \tag{7.2}$$

Notice that the value of 0 is half way between -1 and 1. If you know the sign of $f(x)$, you know the class assigned. That is, classification is then undertaken by the following rule,

$$G(x) = \text{sign}\,(\beta_0 + x^T\beta). \tag{7.3}$$

A lot of information can be extracted from the two equations. One can determine for any i whether $y_i f(x_i) > 0$ and, therefore, whether it is correctly classified.[2] $\beta_0 + x^T\beta$ can be used to compute the signed distance of any fitted point in the predictor space from the separating hyperplane. Hence, one can determine whether a fitted point is on the wrong side of its fence and if so, how far.

Putting all this information together, we are ready to take on the margin maximization task. For the separable case, the trick is to find values β and β_0, to maximize the margin.

Let M be the distance from the separating hyperplane to the margin boundary. Then the goal is

$$\max_{\beta,\beta_0,\|\beta\|=1} M, \tag{7.4}$$

subject to

$$y_i(\beta_0 + x_i^T\beta) \geq M, \quad i = 1, \ldots, N, \tag{7.5}$$

where for mathematical convenience the regression coefficients are standardized to have a unit length.[3] In words, our job is to find values for β and β_0 so that M is as large as possible for observations that are correctly classified. Notice that $2M$ is the margin.

The left-hand side of Eq. 7.5 in parentheses is the distance between the separating hyperplane and a fitted point. Because M is a distance centered on the separating hyperplane, Eq. 7.5 identifies correctly classified observations on or beyond their margin boundary. No cases are inside their fences. Thus, M is sometimes characterized as producing a "hard boundary" because it is statistically impermeable. That is basically the whole story for the support vector classifier when the outcomes are linearly separable.

[2] Because y is coded as 1 and -1, products that are positive represent correctly classified cases.

[3] Because there is no intention to interpret the regression coefficients, nothing important is lost.

It can be mathematically easier, if less intuitive, to work with an equivalent formulation:[4]

$$\min_{\beta,\beta_0} \|\beta\| \tag{7.6}$$

subject to

$$y_i(\beta_0 + x_i^T\beta) \geq 1, \quad i = 1, \ldots, N. \tag{7.7}$$

Because $M = 1/\|\beta\|$, Eq. 7.6 now seeks to *minimize the norm* of the coefficients through a proper choice of the coefficient values. (Hastie et al. 2009: Sect. 4.5.2). Equation 7.7 defines a linear constraint and requires that the points closest to the separating hyperplane are at a distance of 1.0, and that all other observations are farther away (i.e., distance > 1). Equations 7.6 and 7.7 do not change the underlying optimization problem and lead to a more direct, easily understood solution (Bishop 2007: 327–328).

7.2.2 The Nonseparable Case

We return for the moment to Eqs. 7.4 and 7.5, but for the nonseparable case, some encroachments of the buffer zone have to be tolerated. Suppose one defines a set of "slack" variables $\xi = (\xi_1, \xi_2, \ldots, \xi_N)$, $\xi_i \geq 0$, that measure how far observations are on the wrong side of their fence. We let $\xi_i = 0$ for observations that are on the proper side of their fence or right on top of it; they are correctly classified and not in the buffer zone. The farther an observation moves across its fence into or through the buffer zone, the larger is the value of the slack variable.

The slack variables lead to a revision of Eq. 7.5 so that

$$y_i(\beta_0 + x_i^T\beta) \geq M(1 - \xi_i) \tag{7.8}$$

for all $\xi_i \geq 0$, and $\sum_{i=1}^{N} \xi_i \leq W$, with W as some constant quantifying how tolerant of misclassifications one is prepared to be.

The right-hand side of Eq. 7.8 equals M when an observation falls on top of its margin. For observations that fall on the wrong side of their margin, ξ_i is positive. As the value of ξ_i becomes larger, the margin-based threshold becomes smaller and more lenient as long as the sum of the ξ_i is less than W (Bishop 2007: 331–332). Equation 7.8, changes a hard thresholding as a function M into a soft thresholding as a function of $M(1 - \xi_i)$. The fence is no longer statistically impermeable.

There is again an equivalent and more mathematically convenient formulation, much like the one provided earlier as Eqs. 7.6 and 7.7 (Hastie et al. 2001: 373):

[4]The mathematics behind this is not deep, but there are several steps that require familiarity with vector algebra. Interested readers should be able to find excellent treatments on the web. See, for example, lectures on support vector machines by Patrick H. Winston of MIT or by Yaser Abu-Mostafa of Caltech.

$$\min_{\beta,\beta_0} \|\beta\| \tag{7.9}$$

subject to

$$y_i(\beta_0 + x_i^T\beta) \geq 1 - \xi_i, \quad i = 1, \ldots, N, \tag{7.10}$$

for all $\xi_i \geq 0$, and $\sum_{i=1}^{N} \xi_i \leq W$, with W as some constant. As before the goal is to minimize the norm of the coefficients but with special allowances for slack variables. For larger ξ_i's, the linear constraint is more lenient. Once again, there is soft thresholding. In expositions coming from computer science traditions, Eqs. 7.9 and 7.10 are considered "canonical."

Figure 7.3 is a small revision of Fig. 7.2 showing some important mathematical expressions. Observations for which $\xi_i > 1$ lie on the wrong side of the separating hyperplane and are misclassified. Observations for which $0 < \xi_i \leq 1$ lie in the buffer zone but on the correct side of the separating hyperplane. Observations for which $\xi_i = 0$ are correctly classified and on the margin boundary. The circles with borders are support vectors that will be discussed momentarily.

Equations 7.9 and 7.10 constitute a quadratic function with linear constraints whose quadratic programming solution can be found using Lagrange multipliers (Hastie et al. 2009: Sect. 12.2.1). Figure 7.4 shows a toy example in which there is single variable (i.e., x), a quadratic function of that variable in blue, and a linear constraint in red. The minimum when the constraint is imposed is larger than the minimum when the linear constraint is not imposed. The quadratic programming challenge presented by the support vector classifier is that the single x is replaced by the coefficients in Eq. 7.9, and the simple linear constraint is replaced by the N linear constraints in Eq. 7.10.

In the notation of Hastie et al. (2009: 421), the solution has

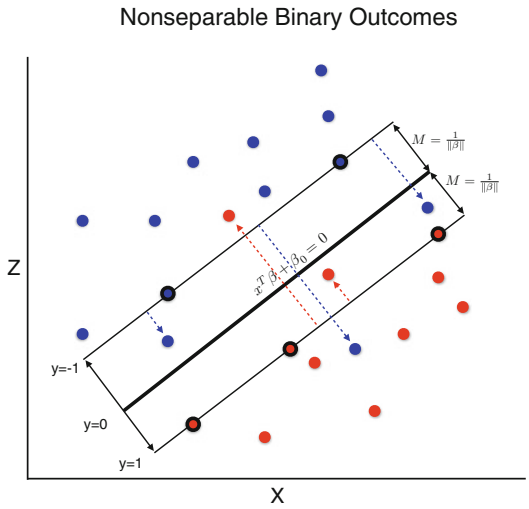

Fig. 7.3 A support vector classifier with some important mathematical expressions for predictors X and Z when there are two classes that are not separable (Support vectors are *circled*.)

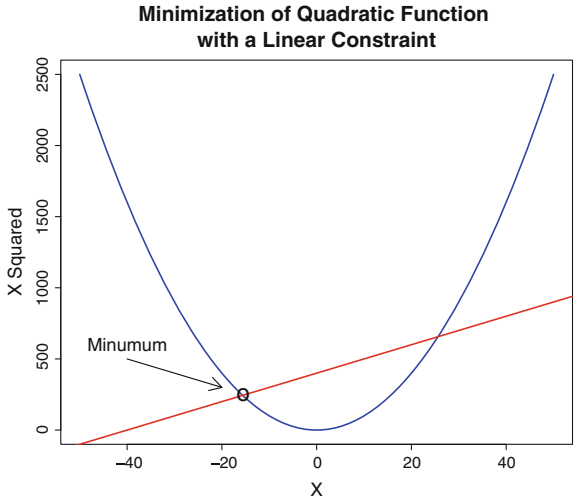

Fig. 7.4 Finding the minimum of a quadratic function with a linear constraint

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i, \tag{7.11}$$

where $\hat{\alpha}_i$ represents a new coefficient for each i whose value needs to be estimated from the data. All of the N values for $\hat{\alpha}_i$ are equal to 0 except for the support vectors that locate the separating hyperplane. The value of $\hat{\beta}_0$ is estimated separately. With all of the coefficients in hand, classification is undertaken with Eq. 7.3: $\hat{G}(x) = \text{sign}(\hat{\beta}_0 + x^T \hat{\beta})$.

7.2.3 Support Vector Machines

We now turn from the support vector classifier to the support vector machine. The transition is relatively simple because support vector machines are essentially support vector classifiers that use kernels as predictors. Kernels were considered at some length in Chap. 2 and will not be reconsidered here. But as we proceed, it is important to recall that (1) the choice of kernel is largely a matter of craft lore and can make a big difference, (2) factors are formally not appropriate when kernels are constructed, and (3) there can be several important tuning parameters.

The Lagrangian is defined as before except that in place of the predictors contained in $\mathbf{X}$, support vector machines work with their linear basis expansions $\Phi(\mathbf{X})$ contained in $\mathbf{K}$. The result is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{i=1}^{N} \hat{\alpha}_i y_i K(x, x_i), \tag{7.12}$$

where $K(x, x_i)$ is the kernel (Hastie et al. 2009: 424; Bishop 2007: 329). All else follows in the same manner as for support vector classifiers.

For $f(x) = h(x)^T\beta + \beta_0$, the optimization undertaken for support vector machines can be written in regularized regression-like form (Hastie et al. 2009: 426; Bishop 2007: 293):

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2, \tag{7.13}$$

where the + next to the right bracket indicates that only the positive values are used. The product $y_i f(x_i)$ is negative when there is a misclassification. Therefore, the term in brackets is positive unless a case is classified correctly and is on the correct side of its fence.[5] The term in brackets is also linear in $y_i f(x_i)$ before becoming 0.0 for values that are not positive. $\|\beta\|^2$ is the squared norm of the regression coefficients, and λ determines how much weight is given to the sum of the slack variables. This is much like the way ridge regression penalizes a fit. A smaller value of λ makes the sum of slack variables less important and moves the optimization closer to the separable case. There will be a smaller margin, but the separating hyperplane can be more complex (Bishop 2007: 332).[6]

Equation 7.13 naturally raises questions about the loss function for support vector machines (Hastie et al. 2001: Sect. 12.3.2; Bishop 2007: 337–338). Figure 7.5 shows with a blue line the "hinge" SVM loss function. The broken magenta line is a binomial deviance loss of the sort used for logistic regression. The binomial deviance has been rescaled to facilitate a comparison.

Some refer to the support vector loss function as a "hockey stick." The thick vertical line in red represents the separating hyperplane. Values of $yf(x)$ to the left indicate observations that are misclassified. Values of $yf(x)$ to the right indicate observations that are properly classified. The product of y and $f(x)$ will be ≥ 1 if a correctly classified observation is on the proper side of its fence.

Consider the region defined by $yf(x) < 1$. Moving from left to right, both loss functions decline. At $yf(x) = 0$, the hinge loss is equal to 1.0, and an observation is a support vector. Moving toward $yf(x) = 1$, both loss functions continue to decline. The hinge loss is equal to 0 at $yf(x) = 1$. The binomial deviance is greater than 0. For $yf(x) > 1$, the hinge loss remains 0, but the binomial deviance continues to decline, with values greater than 0.

One can argue that the two loss functions are not dramatically different. Both can be seen as an approximation of misclassification error. The misclassification loss function would be a step function equal to 1.0 to the left $yf(x) = 0$ and equal to 0.0 at or to the right of $yf(x) = 0$. It is not clear in general when the hinge loss or the

[5] An example of a correct classification on the wrong side of its fence: $1 - (.9) = .1$. An example of a correct classification on the right side of its fence: $1 - (1.1) = -.1$. For a case that is a support vector: $1 - (1) = 0$.

[6] λ is equal to the reciprocal of the weight given to the sum of the slack variables in the usual Lagrange expression (Hastie et al. 2009: 420, 426).

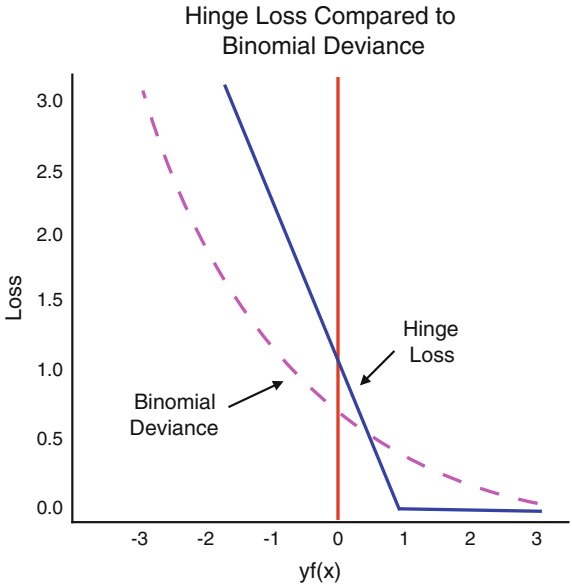

Fig. 7.5 Binomial and hinge loss as a function of the product of the true values and the fitted values

binomial deviance should be preferred although it would seem that the hinge loss would be somewhat less affected by outliers.

7.2.4 SVM for Regression

Support vector machines can be altered to apply to quantitative response variables. One common approach is to ignore in the fitting process residuals smaller in absolute value than some constant (called ϵ-insensitive regression). For the other residuals, a linear loss function is applied. Figure 7.6 provides an illustration.

The result is a robustified kind of regression. Any relative advantage in practice from support vector machine regression compared to any of several forms of robust regression is not clear, especially with what we have called kernelized regression in the mix. But readers interested in regression applications will find what they need in the *kernlab* or *e1071* libraries. For example, *kernlab* has a form of kernelized quantile regression.

7.2.5 Statistical Inference for Support Vector Machines

To this point, the discussion of support vector machines has been presented as a level I problem. But a level II analysis can be on the table. Equation 7.13 makes

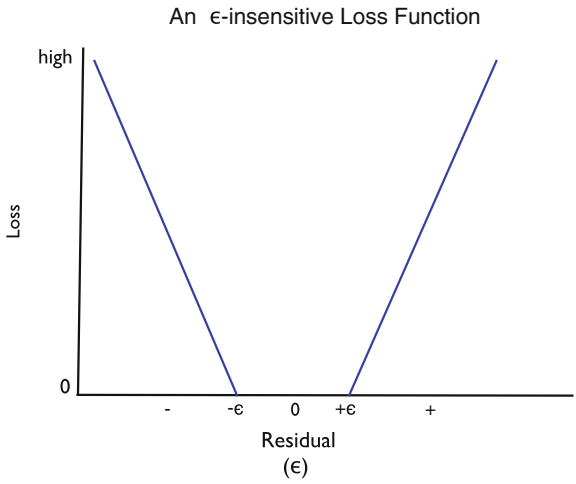

Fig. 7.6 An example of an ϵ-insensitive loss function that ignores small residuals and applies symmetric linear loss to the rest

clear that support vector machines is a form of penalized regression. In particular, it is essentially kernelized ridge regression with a hinge loss function. Therefore, the discussion of statistical inference undertaken in Chap. 2 applies. A clear and credible account of an appropriate data generation process is essential. A proper estimation target must be articulated. And then, having legitimate test data can be very important, or at least an ability to construct sufficiently large split samples. As before, however, the results of support vector machines are sample size dependent because a kernel matrix is $N \times N$. This affects how the estimation target is defined. For example, kernels from split samples will necessarily be smaller than $N \times N$, which alters the estimation target. The estimation target is now a support vector machine for a kernel matrix based on fewer than N observations. In effect, the number of predictors in $\mathbf{K}$ is reduced.

7.3 A Classification Example

Support vector machines perform much like random forests and stochastic gradient boosting. However, there can be much more to tune. We undertake here a relatively simple analysis using the Mroz dataset from the *car* library in R. Getting fitted values with a more extensive set of inputs is not a problem. The problem is linking the inputs to outputs with graphical methods, as we will see soon.

The data come from a sample survey of 753 husband-wife households. The response variable is whether the wife is in the labor force. None of the categorical predictors can be used, which leaves household income exclusive of the wife's income, the age of the wife, and the log of the wife's expected wage. For now, two predictors are selected: age and the log of expected wage. About 60 % of the

wives are employed, so the response variable is reasonably well balanced, and there seems to be nothing else in the data to make an analysis of labor force participation problematic.

Figure 7.7 shows the code to be used. The recoding is undertaken to allow for more understandable variables and to code the response as a factor with values of

```
#### SVM With Mroz Employment Data ####
library(car)
data(Mroz)
attach(Mroz)

# Recodes
Participate<-as.factor(ifelse(lfp=="yes",1,-1)) # For clarity
Age<-age
LogWage<-lwg
Income<-inc
mroz<-data.frame(Participate,Age,LogWage,Income)

### Radial kernel with defaults: kpar="automatic",type="C-svc"
library(kernlab)
svm1<-ksvm(Participate~Age+LogWage,data=mroz,kernel="rbfdot",
           cross=5)
preds1<-predict(svm1,newdata=mroz) # Fitted values
summary(preds1) # Standard output
table(mroz$Participate,preds1) # Confusion table
prop.table(table(mroz$Participate,preds1),1) # Percentage
plot(svm1,data=mroz) # Plot separating hyperplane

### ANOVA kernel
#Define Weights
wts<-table(Participate) #Connects levels to Counts
wts[1]=.47 # Replace count for -1 class
wts[2]=.53 # Replace count for 1 class

library(kernlab)
svm2<-ksvm(Participate~Age+LogWage+Income,data=mroz,
           kernel="anovadot",kpar=list(sigma=1,degree=1),
           C=5,cross=3,type="C-svc",class.weights=wts)
svm2 # Standard output
preds2<-predict(svm2,newdata=mroz) # Fitted classes
table(mroz$Participate,preds2) # Confusion table
prop.table(table(mroz$Participate,preds2),1) # Percentage
plot(svm2,data=mroz,slice=list(Income=17)) # At median income
```

Fig. 7.7 R code for support vector machine analyses of labor force participation

1 and −1. The numerical values make the graphical output examined later easier to interpret.

The first analysis is undertaken with a radial kernel, which has a reputation of working well in a variety of settings. There are two tuning parameters. C is the penalty parameter determining the importance of the sum of the slack variables in the Lagrangian formulation. A larger value forces the fit toward the separable solution. We use the default value of 1. The other tuning parameter is σ, which sits in the denominator of the radial kernel. We let its value be determined by an empirical procedure that "estimates the range of values for the sigma parameter which would return good results when used with a Support Vector Machine (*ksvm()*). The estimation is based upon the 0.1 and 0.9 quantile of $\|x - x'\|^2$. Basically any value in between those two bounds will produce good results" (online documentation for *ksvm()* in the library *kernlab*). A single measure of spread is being provided for the entire set of predictors. The squared norm is larger when cases are more dissimilar over the full set of predictors. Finally, a cross-validation measure of classification error is included to get a more honest measure of performance.

Table 7.1 shows an in-sample confusion table. There are no out-of-bag observations or test data; the table is constructed in-sample. But C was set before the analysis began, and σ was determined with very little data snooping. The proportion misclassified in the training data was .28, and the fivefold cross-validation figure was .29.[7] Because the two proportions are very similar, overfitting apparently is not an important problem for this analysis.

Table 7.1 is interpreted like all of the earlier confusion tables, although the sign of the fitted values determines the class assigned. The empirical cost ratio is little less than two (i.e., 139/72). Incorrectly classifying a wife as in the labor force is about two times more costly than incorrectly classifying a wife as not in the labor force. That cost ratio would need to be adjusted should it be inconsistent with the preferences of stakeholders.

The results look quite good. Overall, the proportion misclassified is .28, although it should be cost weighted to be used properly as a performance measure. When a logistic regression was run on the same data with the same predictors, the proportion misclassified was .45. The large gap is an excellent example of the power of support vector machines compared to more conventional regression approaches. Model error

Table 7.1 SVM confusion table for forecasting labor force participation (radial kernel, default settings)

	Predict not labor force	Predict labor force	Model error
Not labor force	253	72	.22
Labor force	139	289	.32
Use error	.35	.20	Overall error = .28

[7] Both are included as part of the regular *ksvm()* output.

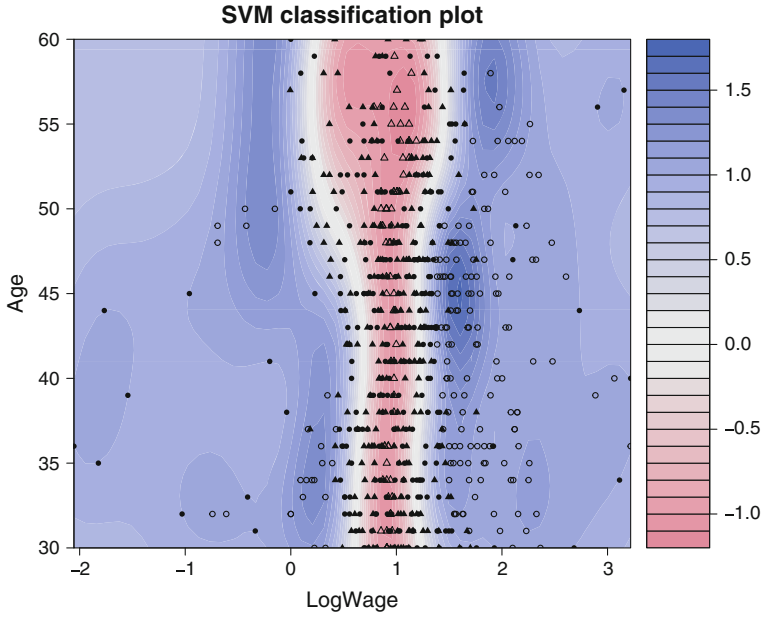

Fig. 7.8 Contour plot of SVM fitted values for labor force participation showing the separating hyperplane, observed values of the response, and support vectors (The *circles* are wives in the labor force. The *triangles* are wives not in the labor force. *Filled circles* or *triangles* are support vectors. A radial kernel with default settings was used.)

and use error also look good. For example, when a wife is predicted to be in the labor force, that classification is correct about 80 % of the time.

There are no variable importance plots or partial dependence plots available in *kernlab()*.[8] However, one can plot the separating hyperplane for two predictors in the units of those predictors. Figure 7.8 is a contour plot showing the separating hyperplane for labor force participation in units of the fitted values. Positive fitted values in shades of blue mean that a wife was classified as in the labor force. Negative fitted values in shades of red mean that a wife was not classified as in the labor force. Age in years is on the vertical axis, and the log of expected wage is on the horizontal axis. Individuals in the labor force are shown with circles. Individuals not in the labor force are shown with triangles. Filled circles or triangles are support vectors.

The colors gradually shift from red to blue as the fitted values gradually shift from less than −1.0 to more than 1.5. The deeper the blue, the larger the positive fitted values. The deeper the red, the smaller the negative (i.e., more negative) fitted values. Deeper blues and deeper reds mean that an observation is farther from the separating hyperplane and more definitively classified. Consequently, the fitted values play a

[8]The other popular support vector machines library in R is *e1071*. It works well, but has fewer kernel options than *kernlab* and many fewer features for working with kernels. It also lacks variable importance plots and partial dependence plots.

Table 7.2 SVM confusion table for forecasting labor force participation (ANOVA kernel, cost weighted, $\sigma = 1$, degree = 1, C = 5)

	Predict not labor force	Predict labor force	Model error
Not labor force	225	100	.30
Labor force	113	315	.26
Use error	.33	.24	Overall error = .29

similar role to the vote proportions in random forests. Bigger is better because bigger implies more stability. If one were doing forecasting, the fitted value for each case could be used as a measure of the assigned class reliability.

The margin around the separating hyperplane is shown in white. Its shape and the location of the support vectors may seem strange. But recall that the separating hyperplane is estimated in a predictor space defined by a kernel. When the results are projected back into a space defined by the predictors, complicated nonlinear transformations have been applied.

But perhaps the story in Fig. 7.8 broadly makes sense. The middle pink area gets wider starting around age 50. At about that age, the number of wives not in the labor force increases over a wider range of expected wages. The larger blue areas on either side indicate that either a low expected or a high expected wage is associated with greater labor force participation. The former may be an indicator of economic need. The latter may be an indicator of good job prospects. There is little evidence of interaction effects because the pink area is effectively perpendicular to the horizontal axis.

For illustrative purposes, the same data can be reanalyzed changing the kernel and empirical cost ratio. An ANOVA kernel is used because in practice, it is often recommended for regression applications. Also, just as in stochastic gradient boosting, one can apply case weights to alter the empirical cost ratio. Here, a weight of .53 is applied to the 1s and a weight of .47 is applied to the −1s so that cases with wives in the labor force are given more relative weight. Finally, a third predictor is added to the mean function to illustrate later an interesting graphics option.

Table 7.2 shows the confusion table that results with the value for σ set to 1.0, the value for degree set to 1, the value for C set to 5.0, and household income as a third predictor. All three tuning values were determined after some trial and error using performance in confusion tables to judge.

The empirical cost ratio is now about 1 to 1 (i.e., 113/100), and classification error for being in the labor force has declined from .32 to .26. In trade, classification error for not being in the labor force has increased from .22 to .30. The overall proportion misclassified when *not weighted* by costs is effectively unchanged (i.e., .28) With so many alterations compared to the previous analysis, it is difficult to isolate the impact of each new feature of the analysis. However, it seems that including household income does not make a large difference.

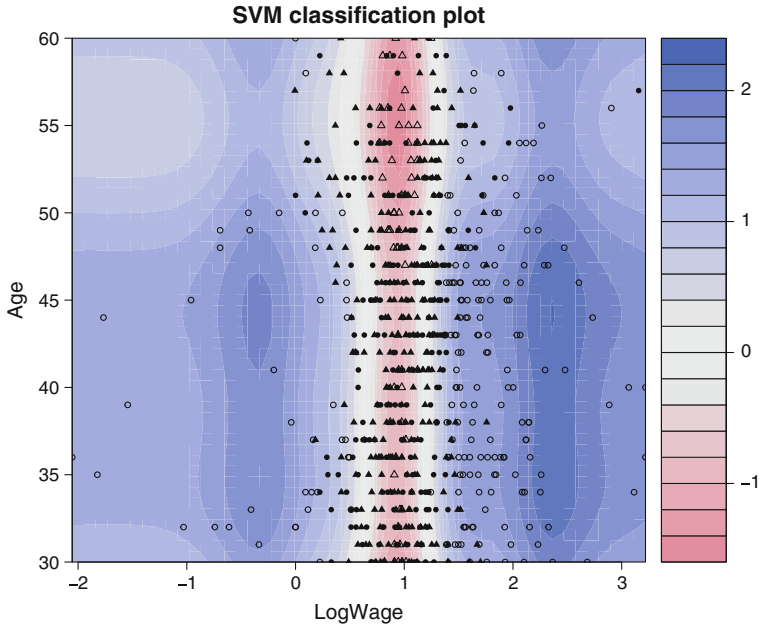

Fig. 7.9 Contour plot of SVM fitted values for labor force participation showing the separating hyperplane, observed values of the response, and support vectors (The circles are wives in the labor force. The triangles are wives not in the labor force. Filled circles or triangles are support vectors. An ANOVA kernel was used, cost weighted, with $\sigma = 1$, Degree = 1, C = 5, and household income values set to its median.)

Figure 7.9 shows the corresponding plot for the separating hyperplane. The layout is the same, but the content is different. Plots with two predictor dimensions means that the roles of only two predictors can be displayed. Here, there are three predictors. In response, the plotting function (*ksvm.plot()*) requires the predictors not displayed in the plot be set to some value. They are, in effect, held constant at that value. If any such predictors are not explicitly fixed at some value, the default has them fixed at 0.0. One can see from the last line of Fig. 7.7 that for all observations, the median of $17,000 is the assigned, fixed value.

This approach is less than ideal. It assumes that there are no interaction effects with the predictors whose values are fixed. An absence of interaction effects seems unlikely, so the issue is whether those interaction effects are large enough to matter. Perhaps the only practical way to get some sense is to examine a substantial number of plots like Fig. 7.9 with fixed values at other than the median. But even an exhaustive set of two-variable plots cannot be definitive unless there are only three predictors overall. That way, there are no interaction effects involving two or more of the fixed predictors.

The substantive message in Fig. 7.9 has not changed much. Because the 1s have been given more weight, more of them are forecast. As a result, the blue area is

larger, and the red area is smaller. But the substantive conclusions about the roles of age and expected wage are about the same. Holding household income constant at its median does not seem to matter much.

All of the results so far have been a form of level I analysis. But for these data, a level II analysis should be seriously considered. The data are from a sample survey with a well-defined, finite population. The data generation process is clear. A reasonable estimation target is the population SVM regression with the same loss function and values for the tuning parameters as specified in the data analysis. The main obstacle is the lack of test data. A split sample approach could be applied if one is prepared to redefine the estimation target so that it has a smaller sample size.

A split sample reanalysis was undertaken in which the sample of 753 observations was randomly split into nearly equal halves (i.e., there is an odd number of observations). The code for the radial kernel analysis was applied to one half of the data. The confusion table code and the code for a separating hyperplane plot were applied to the other half. The results were very similar (within sampling error). An important implication is that in this instance, the results are not affected by cutting the number of observations in half. As in earlier chapters, a nonparametric bootstrap could be applied to the output from the second split of the data to provide useful information on the uncertainty of that output.

7.4 Summary and Conclusions

Support vector machines have some real strengths. SVM was developed initially for classification problems and performs well in a variety of real classification applications. As a form of robust regression, it may also prove to be useful when less weight needs to be given to more extreme residuals. And, the underlying fundamentals of support vector machines rest on well-considered and sensible principles.

Among academics, the adjective "interesting" is to damn with faint praise. But the recent applications discussed in Ma and Guo (2014) are genuinely interesting. They illustrate the rich set of data analysis problems to which support vector machines are being applied. They also document that a large number of talented researchers are working with and extending support vector machines. Its future looks bright.

In general, however, the comparative advantage of support vector machines compared to random forests and stochastic gradient boosting is not apparent. To begin, there is no evidence that it typically leads to smaller generalization error. Problems working with indicator predictor variables will often be a serious constraint. And choosing an appropriate kernel coupled with the required tuning can be a challenge. Finally, SVM may be overmatched by "big data."

Where support vector machines seem to shine is when the number of predictors and number of observations are modest. Then, kernels can have genuine assets that other machine learning procedures may not be able to match. The analysis immediately above is perhaps an example.

Exercises

Problem Set 1

Support vector machines begin with kernels. Review the section on kernels in Chap. 2 looking especially at the material on radial kernels. Load the R dataset *trees* and have a look at the three variables in the file. The code below will allow you to explore how the matrix derived from the radial kernel changes depending on the values assigned to σ. Try values of .01, .05, .1, and 1. Consider how the standard deviation changes, excluding matrix elements equal to 1.0. Also have a look at the three-dimensional histograms depending on the value of σ. Describe what you see. How do the changes you see affect the complexity of the function that can be estimated?

```
library(kernlab) # you may need to install this
library(plot3D) # you may need to install this
X<-as.matrix(trees)
rfb<-rbfdot(sigma=.01) # radial kernel
K<-kernelMatrix(rfb,X)
sd(K[K<1]) # standard deviation with 1's excluded.
hist3D(z=K,ltheta=45,lphi=50,alpha=0.5,opaque.top=T,scale=F)
```

Problem Set 2

Construct a dataset as follows.

```
w<-rnorm(500)
z<-rnorm(500)
w2<-w^2 x<-(-1+3*w2-1*z)
p<-exp(x)/(1 + exp(x))
y<-as.factor(rbinom(500,1,p))
```

1. Regress y on w and z using logistic regression and construct a confusion table with the resubstituted data. You know that the model has been misspecified. Examine the regression output and the confusion table. Now regress y on $w2$ and z using logistic regression and construct a confusion table with the resubstituted data. You know that the model is correct. Compare the two sets of regression coefficients, their hypothesis tests, and two confusion tables. How does the output from the two models differ? Why?
2. Can you do as well with SVM using w and z as when logistic regression is applied to the correct model? With w and z as predictors (not $w2$), use an ANOVA kernel in *ksvm()* from the library *kernlab*. You will need to tune the ANOVA kernel using some trial and error. (Have a look at the material on the ANOVA kernel in Chap. 2.) Start with $sigma = .01$ and $degree = 1$. Increase both in several steps until $sigma = 10$ and $degree = 3$. Chose the values that give you the fewest classification errors. How does this confusion table from the well-tuned SVM

compare to the confusion tables from the correct logistic regression? What is the general lesson?
3. Construct and interpret the SVM classification plot for the best SVM confusion table.
4. Repeat the SVM analysis, but with class weights. To construct the nominal weights with a ratio of, say 3 to 1, use

```
wts<-c(3,1) # specify weights
names(wts)<-c("0","1") #assign weights to classes
```

Insert these two lines of code before the call to *ksvm()* and then include the argument *class.weights=wts* in *ksvm()*. Try several different pairs of nominal weights until you get a cost ratio in the confusion table such that the 1s are three times as costly as the 0s. How do the results differ?

Problem Set 3

1. From the *MASS* library, load the Pima.tr dataset. The variable "type" is the response. All other variables are predictors. Apply $ksvm()$ from the *kernlab* library and again use the ANOVA kernel and weights to address asymmetric costs. The doctor wants to be able to start treatment before the test results are in and thinks it is twice as costly to withhold treatment from a patient who needs it compared to giving treatment to a patient who does not need it. Apply SVM to the data so that the confusion table has a good approximation of the desired cost ratio and about as good performance as can be produced with these data. This will take some tuning of the ANOVA kernel and the tuning parameter C, which determines the weight given to the penalty in the penalized fit. (In Problem Set 2, C was fixed at the default value of 1.0.) A good strategy is to first set $C = 1$ and tune the ANOVA kernel. Then, see if you can do better altering the value of C.
2. Choose two predictors in which you think the doctor might be particularly interested and construct an SVM classification plot. Fix all other predictors at their means. Interpret the plot. Do you think there is any useful information in the plot to aid the physician. Why?