7

Support Vector Machines

Support vector machines (SVM) will seem somewhat far afield from the statistical learning procedures discussed to this point. SVM was developed as a classifier, largely in computer science, with its own set of research questions, conceptual frameworks, technical language, and culture. In addition, a substantial amount of the initial interest in support vector machines stemmed from the important theoretical work surrounding it (Vapnick, 1996). The early applications were not especially compelling.

Over the past few years, the applications to which support vector machines has been applied have broadened (Christianini and Shawe-Taylor, 2000; Moguerza and Munõz, 2006), and the available software has responded (Joachims, 1998; Hsu et al., 2007; Chen et al., 2004). Support vector machines now can have more the look and feel of regression. Formal links to statistical learning in statistics have been made so that there are increasingly shared concepts and language across interested disciplines (Hastie et al., 2001: Sections 12.1–12.3; Bishop, 2006: Chapters 6–7). And although there are still some very important components of a legitimate regression analysis that are a bit beyond SVM's reach, these drawbacks will probably be remedied reasonably soon. It is useful therefore to spend some time providing a brief summary of how support vector machines works. It introduces a number of intriguing ideas and has considerable promise for regressionlike applications.

7.1 A Simple Didactic Illustration

Consider a very simple classification problem. A guidance counsellor in a small high school is trying to determine which students are at risk for dropping out. There are data on students from the past several years from which one can determine which students dropped out. There is also a promising set of covariates. In looking at the data, the guidance counsellor notices that all students who were reading at second-grade level dropped out of school. At the other extreme, none of the students who were reading at a twelfth-grade level

R.A. Berk, *Statistical Learning from a Regression Perspective*,
DOI: 10.1007/978-0-387-77501-2_7,

ever dropped out. For both of these sets of students, the high-risk students and the low-risk students, the classification job is done because this eyeball classifier works perfectly.

What about the rest of the students? The eyeball classifier stumbles because it is not apparent how to subset the students any further so that the two classes of students are homogeneous. For example, among those students reading at tenth-grade level, seven out of every ten graduate. It might make sense, therefore, to employ a more powerful classifier for these students. And the quality of the classifier overall will depend on how accurately the students between the two extremes can be classified. In other words, variation in the loss function will depend only on how accurately the middle group of students is assigned to the binary outcome.

Focusing only on the middle group of students, a useful classifier might try to put the students into one of the two classes so that a pair of objectives are achieved. First, there should be a small number of misclassifications: students incorrectly labeled as dropouts and students incorrectly labeled as graduating high school. There is nothing new in this.

Second, subject to this small number of misclassifications, the students in the two classes should be as different as possible in their reading ability. If there is a continuum of reading ability monotonically related to dropping out of school, greater separation between the two groups can imply more stable classifications. For example, if the two groups can be divided so that the highest reading level for the dropout group is at least one grade level below the lowest reading level for the higher reading group, a relatively clear distinction has been made. In contrast, if that gap is only 5% of a grade level, the distinction is not nearly so clear. One grade level implies approximately nine months of class time, whereas 5% of a grade level may imply only two weeks of class time. The larger distinction is desirable because it implies that the classifications are more stable under random perturbations of the data. Generalization error will be smaller.

Operationally, suppose the guidance counsellor is prepared to live with 20 misclassifications. Given 20 misclassifications, what reading level should be chosen to separate the two classes? A fourth-grade level? A sixth-grade level? An eighth-grade level? It would make sense to choose a threshold so that on either side, the difference in reading ability between the two groups was as large as possible.

This method of partitioning the data has some important similarities to nearest neighbor methods. The distinction between the two groups of students only depends on where students are located on the measure of reading ability. Students who have similar reading abilities will be classified in the same manner. It is the distance between students that matters, not their raw score. Note that there is no concern with how the data were generated and no substantive interest in how predictors might be related to a response. The goal is solely to construct accurate and stable classifications by classifying similar students similarly.

An accurate and stable classification procedure could give the guidance counsellor a helpful forecasting device. In the future, students similar to those who had been classified as dropouts because of their inability to read well would be seen as having a great risk of dropping out. Interventions of various sorts might follow.

This very simple example illustrates several fundamental features of SVM. It is a classifier that partitions the data, but unlike classification trees, does not do so in a stagewise fashion. In addition, only observations near the classification boundary figure directly in the fitting process. Then, the partitioning is accomplished so that the distance in the predictor space between the two groups is as large as possible; separation is maximized conditional on a predetermined and tolerable number of classification errors. Maximizing the separation serves much the same purpose as large margins in bagging and random forests. Finally, the key information extracted from the predictors is, in effect, a matrix of distances. Observations sufficiently near one another in predictor space will be treated alike.

No simple example can be made to map exactly onto support vector machines. Moving back and forth between the guidance counsellor illustration and the diagrams and equations that follow will reveal an imperfect match. But with some central issues now raised, the new material may be somewhat more accessible.

7.2 Support Vector Machines in Pictures

With the simple example behind us, we can turn to a bit more formal exposition of SVM. Despite its roots in computer science and its focus on classification per se, at its core support vector machines can be treated in a manner introduced in Chapter 2. Hastie et al. (2001: 380–381) point out that one can use the familiar formulation of fitting a set of observed values of a response variable subject to a complexity penalty. But if we skip to that punch line, the underlying intuitions will likely be lost.

7.2.1 Support Vector Classifiers

Suppose there is a binary response variable coded, as is often done in boosting, as"1" and "–1." There is a fitting function $f(x)$, where x can be one or more predictors. If the $f(x)$ returns a positive number, the label "1" is assigned to the observation. If the $f(x)$ returns a negative number, the label "–1" is assigned to the observation. A fitting function can be written as

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

where, as before, $h(x)^T$ are basis functions of x.

In the SVM literature, the response variable is often called the "target variable," and the intercept in Equation 7.1 is often called the "bias." Nevertheless, the basic idea is the same as before: there is a series of basis functions $h(x)$, where x can represent more than one predictor. The basis functions are additively combined, with the vector βs as the weights. Then the goal, as usual, is to minimize the loss without making the fitting function unnecessarily complex.

But beneath the surface, the fitting exercise is quite novel. A key feature is that when undertaking a classification task, some observations are treated very differently from others. In much the same spirit as boosting, the observations that are more difficult to classify receive more attention. But unlike boosting, a qualitative distinction is first made between the observations that are difficult to classify and observations that are not. Then, the problematic observations determine the criterion by which classes are to be assigned. This, in turn, implies the use of an unusual loss function. Thinking back to the guidance counsellor illustration, observations that fall in regions where there is a mix of students who drop out and students who graduate are the observations that are difficult to classify and consequently, the observations that determine the precise location of the threshold to be imposed separating high-risk from low-risk students.

Another key feature is that the predictors affect the class assigned by how they locate observations in the space defined by the predictors. What matters is where observations fall with respect to one another. This is a significant difference between the statistical learning procedures in this chapter and those discussed in earlier chapters. Details are provided a little later. But it is a bit like what real estate agents often say: what matters is "location, location, location."

Figure 7.1 shows a partitioning diagram. As before, there are two predictors (x and z) and a binary response y, that can take on values of A or B. B might represent dropping out of school and A might represent graduating. (A could be coded as 1 and B could be coded as –1.) The two predictors might be reading grade level and the number of truancies per semester. In this figure, the As and the Bs 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 groupings.

With data of this sort, one can apply a "support vector classifier." The goal is to locate a "decision boundary," here represented by the dark straight line, using information from the predictors so that the partitions are as homogeneous as possible. In spirit, this is lot like CART. The decision boundary is also called a "separating hyperplane." Observations that fall on one side of the decision boundary are assigned to one class, and observations that fall on the other side of the decision boundary are assigned to the other class. In this instance, each observation above and to the right would be correctly classified as an A. Each observation below and to the left would be correctly

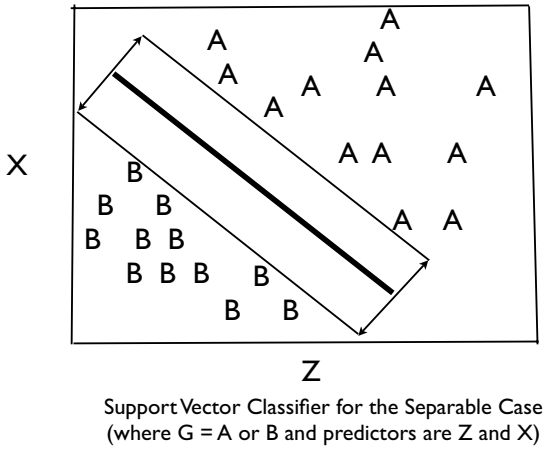

Fig. 7.1. Support vector classifier when there are two separable classes.

classified as a B. Future observations would then be forecasted to an A or a B depending on where in the predictor space they fell.

For Figure 7.1, finding a decision boundary is easy. In fact, it is too easy. There are a limitless number of linear decision boundaries one could draw that would also define two perfectly homogeneous partitions. A way needs to be found to determine the "best" decision boundary among all those that sensibly could be constructed.

For data such as those shown in Figure 7.1, the support vector classifier solves this problem by constructing two parallel lines on either side of, and the same distance from, the decision boundary. The two lines are 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 data "buffer zone." Observations can fall right on either fence but not across them, inside the buffer zone. In Figure 7.1, the total width of the buffer zone is shown with the two double-headed arrows.

The distance between the two fences or the distance between the decision boundary and either fence is called the "margin." The different definitions amount to the same thing in practice, and justification for maximizing the margin has a familiar ring. The wider the margin is, the greater the separation between the two classes. Larger margins are desirable because generalization error will usually be smaller. A more definitive and, therefore, more stable distinction is being made between the two classes. Thus, the buffer zone's margin plays much the same role as the margin used in bagging and random

forests, although it is not defined in the same way. Sometimes the two fences are called the "margin boundary."

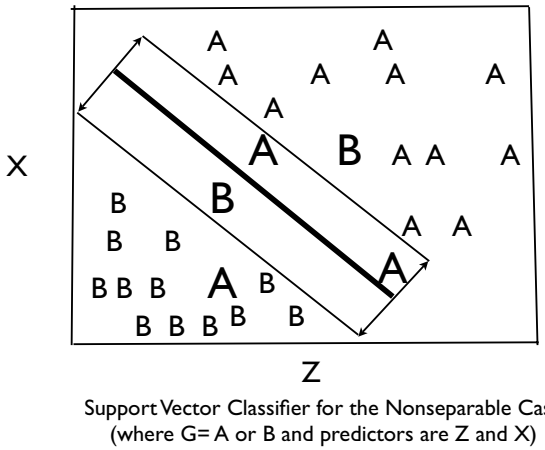

Fig. 7.2. Support vector classifier when there are two classes that are not separable.

The data shown in Figure 7.1 are very cooperative, and such cooperation is in practice rare. Figure 7.2 shows a partitioning diagram that is much like Figure 7.1, but the two sets of values are no longer separable with a straight line. The three larger *A*s and the two larger *B*s violate the margin boundary. They are on the wrong side of their respective buffer zone fences. The large *A*s are too far toward the lower left of the figure, and the large *B*s are too far toward the upper right. Moreover, there is no way to relocate and/or narrow the buffer zone so there is a separating hyperplane able to partition the space into two perfectly homogeneous regions. There is no longer any linear solution to this classification problem.

One possible response is to permit some misclassifications. One could specify some fraction of the observations that could be allowed to fall on the wrong side of the decision boundary. That is, one could try to live with a result that looked a lot like Figure 7.2. The idea would be to maximize the width of the buffer zone subject to some specified number of misclassifications.

But that is not quite enough. Some misclassified observations fall just across the decision boundary and some fall far away. Some correctly classified observations fall on the wrong side of their fence, violating the need for wide separation between the two classes. But these observations can differ on how far on the wrong side of the fence they fall. In response, the distance between

the relevant fence and the location of an observation can be taken into account for both kinds of violations. For example, there are two large As whose distance from their fence is small, and one whose distance is substantial. The first two are correctly classified. The last one is not. There is a large B close to its fence and a large B much farther away. The former is correctly classified. The latter is not.

The sum of such distances can be viewed as a measure of how permissive one has been when the margin is maximized. If one is more permissive by allowing for a larger sum, it usually possible to construct a larger margin. Again, larger margins are good. More stable classifications can follow. But more permissive solutions imply more bias. There will be a greater number of misclassifications and/or the misclassifications will be more in error. 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.

Sometimes, several observations will fall right on the margin boundary. These observations, in addition to the observations on the wrong side of the margin boundary are used to determine, in the space defined by the predictors, the precise location of the decision boundary and the buffer zone. Because such observations "support" the location of the decision boundary, they are called "support vectors." No other observations play that role.

The support vectors represent the observations most difficult to correctly classify. These As and Bs inhabit much the same space that the predictors define. They can differ from each other in their response values even though they are near each other and, therefore, alike on their predictor values. Thus, the special attention given to the support vectors is in the same spirit as the extra weight given to misclassified cases in boosting.

Figure 7.3 is almost the same as Figure 7.2, but the buffer zone has been enlarged and reoriented a bit to represent a solution to the classification problem. There are two kinds of support vectors: those on the wrong side of the margin boundary and those on top of the margin boundary. The support vectors are shown in large letters. Classification is determined by the side of the decision boundary on which an observation falls. In Figure 7.3, one A is misclassified and one B is misclassified.

7.2.2 Support Vector Machines

There is a still better solution to the classification problem when the classes are nonseparable. Suppose one allows for a nonlinear decision boundary. In somewhat the same manner of the smoothers we considered in Chapter 2, the data are used to help determine an appropriate nonlinear function, within a given category of functions, that can separate the two classes of data. The added flexibility makes it much more likely that the decision boundary will classify accurately and with a large margin.

Once again, a basis function approach can be applied. One can enlarge the predictor space in which the observations sit and construct a nonlinear

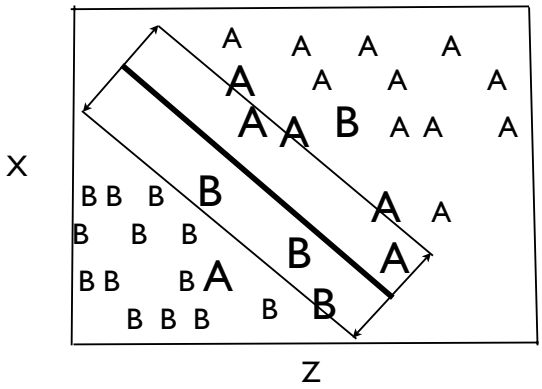

Fig. 7.3. Support vector classifier when there are two classes that are not separable with support vectors shown.

function of the original variables. Support vector machines capitalizes on these ideas by allowing for a very large number of transformations of the predictors, but only for particular classes of transformations. A few of these transformation are considered shortly. We show that with the focus on the location of observations in the predictor space, and especially on their proximity to one another, the transformations are applied not to each predictor by itself, but to the predictor space they define. This is a useful and very clever approach.

There are some of the same tuning issues for support vector machines as there are for support vector classifiers. One tuning parameter is a function of the sum of the distances of the observations on the wrong side of the margin boundary. This sum serves the same purpose as it did for support vector classifiers. One or more other tuning parameters are also common, depending on which predictor transformations are employed. We will see that, somewhat in contrast to random forests and stochastic gradient boosting, variation across a set of reasonable values for the tuning parameters can have a large impact on the results.

In summary, the goal of support vector classifiers is to linearly partition the space defined by the predictors so that two homogeneous regions are defined. The decision boundary for those two regions is determined by making the margin as large as possible. This often leads to a compromise in which some observations are allowed to cross over into, and even beyond, the region inside the margin boundary. Support vector machines takes support vector classifiers

one significant step further by allowing the decision boundary and the margin boundary to be nonlinear. But in both cases, the tradeoff between variance and bias remains.

7.3 Support Vector Machines in Statistical Notation

Support vector classifiers and support vector machines can be considered along with some other procedures as maximum margin classifiers. It is probably fair to say that support vector machines is at this point the most popular. But despite its popularity, or perhaps because of its popularity, there are several somewhat different ways it can be formulated and presented. What follows draws heavily on Hastie and his colleagues (2001) and on Bishop (2006). As before, we start with support vector classifiers to introduce some of the key concepts.

7.3.1 Support Vector Classifiers

There is a set of p predictors and a binary outcome. The goal is to find a linear boundary so that in the space defined by the predictors, the data are partitioned into two regions, both of which are perfectly homogeneous with respect to the response. In addition, the margin is to be as large as possible.

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. The response is coded "1" or "–1." A separating hyperplane of dimension $p - 1$ is defined as

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

Classification is then undertaken by the following rule,

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

The $f(x)$ in Equation 7.2 can be used to compute the signed distance of any point from the separating hyperplane. Thus, one can determine for any i whether $y_i f(x_i) > 0$ and, therefore, is correctly classified. Because y is coded as 1 and –1, products that are positive represent correctly classified cases. One can also determine if that observation is on the wrong side of its fence and if so, how far. Finally, for observations that are misclassified, one can determine how badly.

The ability to generate signed distances greater than 0 between observations and the separating hyperplane, conditional on the values of β and β_0, implies that one can maximize the margin by how β and β_0 are chosen. Values

of β and β_0 are sought so that the distance from the decision boundary to the closest observations is as large as possible. This implies the use of a novel loss function that is discussed later.

More formally, the goal is to

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

subject to

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

The left-hand side of Equation 7.5 is the distance between the decision boundary and an observation. In this formulation, C is the distance from the decision boundary to the margin boundary; $2C$ is the margin. Because C is a distance centered on the decision boundary, Equation 7.5 identifies correctly classified observations on or beyond the margin boundary. Because Equation 7.5 applies to cases $1, \ldots, N$, no cases are inside their fences. Thus, C is sometimes characterized as producing a "hard boundary" that is impermeable. In short, the goal is to find the values of the coefficients that maximize the margin, such that there are no observations inside the fences. This basically characterizes the support vector classifier for the separable case.

It is sometimes useful to work with an equivalent formulation (Hastie et al., 2001: 372):

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

subject to

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

A key change from Equation 7.4 is that the norm constraint on the coefficients has been discarded, so that one can set $C = 1/||\beta||$ (Hastie, 2001: 108–109). Equation 7.7 then can follow. Equation 7.7 requires that the point closest to the decision boundary has a distance 1.0 and that all other observations are farther away (i.e., distance > 1). This particular normalization is arbitrary but does not change the basic problem and can lead to a more direct and easily comprehended solution (Bishop, 2006: 327–328).

Minimizing the norm can be thought of as trying to make the values of the coefficients as small as possible, which forces the margin to be as large as possible. Then, at least one observation must fall on the margin boundary (Bishop, 2006: 328). One can do no better than this and maintain separation.

The Nonseparable Case

To move toward the non separable case, some misclassifications have to be tolerated. Define a set of "slack" variables $\xi = (\xi_1, \xi_2, \ldots, \xi_N)$, $\xi_i \geq 0$, that measure how far incorrectly classified observations are on the wrong side of their fence. We let $\xi_i = 0$ for observations that are correctly classified and on the proper side of their fence or right on top of it; they are not in the buffer

zone. The farther an observation moves across its fence into and even through the buffer zone, the larger is the value of the slack variable. In other words, the value for a slack variable i denotes how inaccurate the classification exercise is for observation i.

The slack variables lead to a revised setting in which the coefficients are chosen so that C is maximized. Specifically,

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

for all $\xi_i \geq 0$, and $\sum_{i=1}^{N} \xi_i \leq K$, with K as some constant.

In Equation 7.8, the "hard boundary" C has been transformed into a "soft" boundary $C(1 - \xi_i)$. The boundary moves from observation to observation depending on the value of ξ_i so that crossing into the buffer zone and even to its other side can be permitted (Bishop, 2006: 331–332). Because slack variables cannot be negative, the constraint becomes smaller as the value of the slack variable increases. It is as if the margin were shrunk proportionally for observations that have a slack variable defined.

For a response variable coded "1" or "–1",

$$\xi_i = |y_i - f(x_i)|. \tag{7.9}$$

Equation 7.9 then implies that

1. $0 < \xi_i \leq 1$ for observations that violate the buffer zone but are correctly classified.
2. $\xi_i = 1$ for observations that are correctly classified and right on top of the decision boundary.
3. $\xi_i > 1$ for misclassified observations, implying that the constraint in Equation 7.8 is negative.

These are the values whose sum cannot exceed K. Because for all misclassified observations $\xi_i > 1$, K can be interpreted as the upper bound on the number of misclassifications. As a result, K can become a useful tuning parameter. With a larger K, there are more support vectors. The decision boundary that follows is more stable; there is less variance from sample to sample. But the price is more misclassifications. There is greater bias. There is no "right" value for K beyond how the classifier performs, and as we show later, choosing the "best" value for K depends heavily on craft lore.

For completeness, we again offer an equivalent formulation much like the one provided earlier (Hastie et al., 2001: 373).

$$\min_{\beta,\beta_0} ||\beta|| \tag{7.10}$$

subject to

$$y_i(\beta_0 + x_i^T\beta) \geq 1 - \xi_i, \quad i = 1, \ldots, N, \tag{7.11}$$

for all $\xi_i \geq 0$, and $\sum_{i=1}^{N} \xi_i \leq K$, with K as some constant. In exposition coming from computer science traditions, Equations 7.10 and 7.11 are considered "canonical." But canonical does not mean trouble free. Bishop (2006:

322) observes that the minimization process can be distorted by outlier values of ξ_i. A very small number of badly misclassified observations can drive the solution.

7.3.2 Support Vector Machines

We now turn from the support vector classifier to the support vector machine. A key difference is the use of nonlinear transformations of the predictors. Thus, beginning with Equation 7.2, x is replaced by $h(x)$. However, the nature of the transformations is unlike any of those considered in earlier chapters.

Suppose one has an $N \times M$ data matrix D of data, which includes a response variable and a set of predictors. N is the number of observations and M is the number of variables. A conventional scatterplot of the data will locate each observation in a space defined by the variables. A cross-product matrix of the data will be $M \times M$ and symmetric with each off-diagonal entry a measure of how one variable is related to another. With proper scaling the cross-product matrix can become a conventional correlation matrix. One can learn which variables are strongly associated with other variables.

One can alternatively proceed with a transpose of D. Now rows are variables and the columns are observations. A different kind of scatterplot might follow in which variables are located in a space defined by observations. Compared to the usual scatterplot, the roles of variables and observations are reversed. A cross-product matrix is now $N \times N$ and symmetric with each off-diagonal entry a measure of how one observation is related to another. Again, with proper scaling the cross-products can be turned into correlations. One can learn which observations are strongly associated with other observations.

Figure 7.4 shows the two different scatterplots. To keep the plots manageable, there are for both just two variables and two observations. The plot on the left shows a conventional scatterplot with variables defining the space. The plot on the right shows an alternative scatterplot with observations defining the space.

This dual representation of data can be applied to any dataset or a subset thereof. In SVM, the predictors are arrayed in observation space so that the relationships between observations can be exploited. The key to all this is a set of inner products. For example, if a is an $N \times 1$ vector, the inner product is $a^T a$, a scalar that is the sum of the squared values of a. If there are two $N \times 1$ vectors a and b, the inner product is $a^T b = ba^T$, which is the sum of their cross-products.

For every pair of cases i and j, one computes $x_i^T x_j$. If there are two predictors, for example, there will be two predictor values for case i and two predictor values for case j. The inner product would be the sum of two cross-products. This sum can be viewed as the distance between case i and case j, or their similarity. And again, with proper scaling the sum of the cross-products can be turned into a correlation.

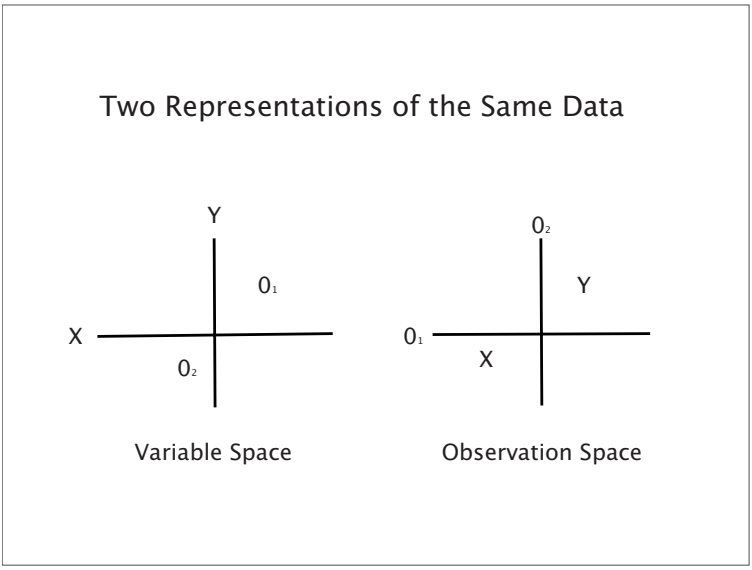

Fig. 7.4. Variable space and observation space.

In order to allow for more flexible fitting, the cross-products can be transformed using a "kernel function" $K(x_i, x_j)$. For example, a relatively simple kernel is the polynomial, which can be represented as

$$K(x_i, x_j) = (1 + x_i^T x_j)^d, \tag{7.12}$$

where d is the order of the polynomial.

The term "kernel" used in SVM is different from the term "kernel" used in smoothing. In smoothing, a kernel refers to a region of the predictor space in which some calculations are to be undertaken. In SVM, a kernel represents a transformation applied to the transposed cross-product matrix of predictors.

Consider a simple numerical example. For observation i, the value for the first variable is 2, and the value for the second variable is 3. For observation j the value for the first variable is 1, and the value for the second variable is 3. So the inner product is $(2 \times 1) + (3 \times 3)$. With a second-order polynomial, the result is

$$K(x_i, x_j) = [1 + (2 \times 1) + (3 \times 3)]^2 = [1 + 2 + 9]^2 = 144. \tag{7.13}$$

For observation i and observation j, 144 is the value of the transformed predictors. The goal of the transformation is to alter how the two observations are related to each other. More generally, it is as if observations are moved around so that separation is more effectively achieved.

There are many different kinds of kernels currently in use or recently proposed (Bishop, 2006: 295–297), but at this point, there is no universal set nor

much consensus about which work best for which kinds of data. In each case, the result must be an $N \times N$ symmetric, positive definite matrix K. This is the input into the optimization procedure whose solution has the following form (Hastie et al., 2001: 378; Bishop, 2006: 329),

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{i=1}^{N} \hat{\alpha}_i y_i K(x, x_i), \tag{7.14}$$

where $\hat{\alpha}_i$ is a positive weight given to each observation estimated from the data (Hastie et al., 2001: Section 12.3.3).

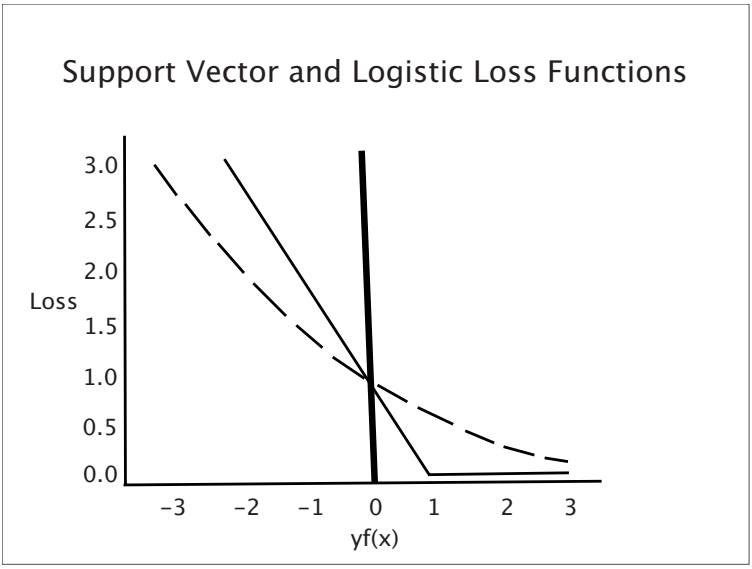

Fig. 7.5. A comparison of classification loss functions.

Interestingly, the minimization exercise for support vector classifiers can be written in regularized sum of squares form (Hastie et al., 2001: 380; Bishop, 2006: 293):

$$\min_{\beta_0, \beta} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \lambda ||\beta||^2, \tag{7.15}$$

where the $+$ indicates that only the positive values are used, and λ plays the same role as K. Note that these residuals are treated in a linear fashion, implying less sensitivity to extreme values than, for instance, a quadratic function.

The use of a regularized sum of squares naturally raises questions about characteristics of the loss function for support vector classifiers and support

vector machines (Hastie et al., 2001: 380–381; Bishop, 2006: 337–338). Figure 7.5 shows the "hinge" loss function (the thin solid line) used by both and for purposes of comparison, the logistic loss function (the thin broken line). The logistic loss function has been rescaled to facilitate a comparison.

Some speak of the support vector loss function as a "hockey stick." The thick vertical line represents the separating hyperplane. Values of $yf(x)$ to the left denote observations that are misclassified. Values of $yf(x)$ to the right denote 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 $yf(x) = 0$, an observation is correctly classified, but in the buffer zone. The loss is equal to 1.0. Moving toward $yf(x) = 1$, both loss functions continue to decline. The support vector loss function is equal to 0 at $yf(x) = 1$. The logistic loss function is greater than 0. For $yf(x) > 1$, the support vector loss function has a loss of 0. The logistic loss function continues to decline, but still has a loss greater than 0.

One could 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 support vector loss function or the logistic loss function should be preferred although it would seem that the support vector loss function would be somewhat less affected by outliers.

7.3.3 SVM for Regression

Support vector machines can be altered to apply to quantitative response variables. One common approach is in the fitting process to only use residuals smaller in absolute value than some constant (called "e-insensitive" regression). These are somewhat analogous to observations on the incorrect side of the fence. For the other residuals, one has a linear loss function. The result is a robustified kind of regression. The relative advantage in practice of support vector machine regression compared to any of several forms of robust regression is not clear.

7.4 A Classification Example

Support vector machines is sufficiently different from the procedures discussed earlier that a graduated illustration may be helpful. We begin with a relatively simple analysis using the Mroz dataset from the *car* library.

The data come from a sample survey of 783 husband–wife households. The response variable will be whether the wife is employed. For now, two predictors are selected: household income exclusive of the wife's earnings in

tens of thousands of dollars, and the log of the wife's expected weekly wage. Expected wage is the earnings anticipated if the wife finds a job. The data are divided randomly into a training dataset of 400 observations and a test data set of 383 observations. About 60% of the wives are employed, so the response variable is reasonably well balanced, and there is nothing else in the data to make an analysis of labor force participation especially problematic.

7.4.1 SVM Analysis with a Linear Kernel

	Predict Unemployed	Predict Employed	Model Error
Unemployed	38	136	.78
Employed	38	188	.17
Use Error	.50	.41	Overall Error = .44

Table 7.1. SVM confusion table for forecasting employment: linear kernel, $cost = 1$, $\gamma = NULL$, 339 support vectors.

Table 7.1 shows the SVM confusion table with a linear kernel, and tuning parameters set at their default values. The linear kernel is used for simplicity. For pairs of observations, the transformation is of the form $(x_i^T x_j)$. The tuning parameter that svm() calls γ is not relevant to the linear kernel and is set to "NULL". The tuning parameter that svm() calls "cost" plays the same role as λ in Equation 7.15 or K in Equation 7.8. It determines how much weight is given to the penalty function or alternatively, how tolerant one is prepared to be about observations that are not fit well. It is through this tuning parameter that one trades bias against variance. The default sets cost to 1.0. Resubstituted data are used to build the table, but with the linear kernel, two predictors, and a sample of 400, overfitting is not likely to be problematic.

The overall proportion of cases misclassified is .44. The SVM model misclassifies unemployed wives 78% of the time and employed wives 17% of the time. Clearly, it is more difficult in this case to accurately classify employed women.

Figure 7.6 shows a classification plot with the linear boundary implied by the linear kernel. The darker area to the lower right represents that part of the predictor space in which all cases are classified as unemployed. The lighter area to the upper left represents that part of the predictor space in which all cases are classified as employed. The "x" symbols denote support vectors. The "o" symbols denote vectors that are not used to locate the decision boundary. The plotting procedure in svm() also has the ability to show which observations represent employed wives and which observations represent unemployed wives, but one has to be able to plot in color. R does, but there are no color reproductions in this book.

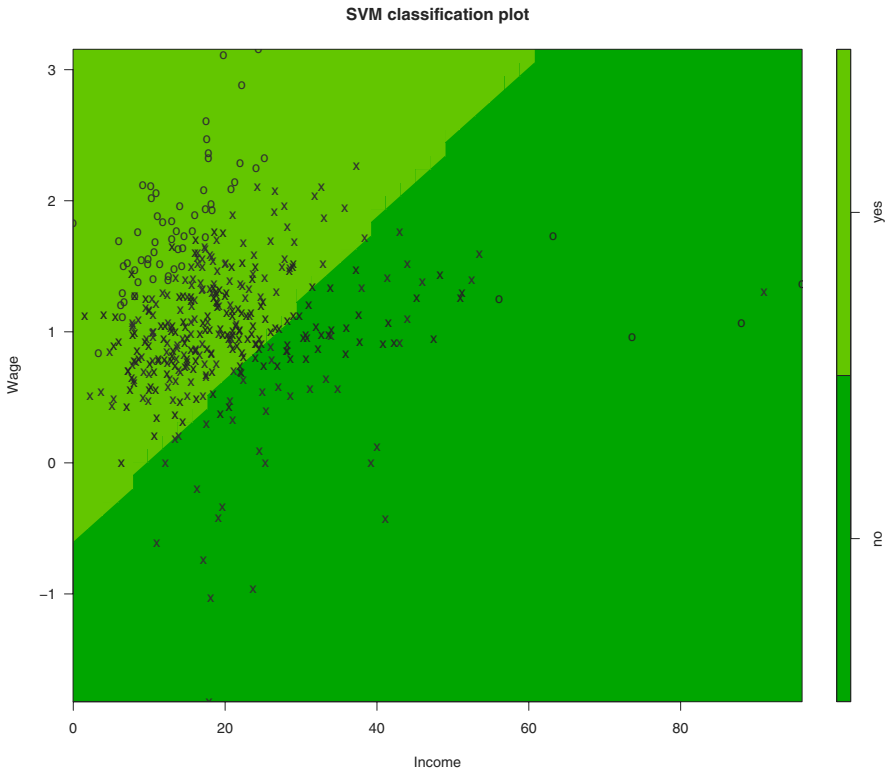

Fig. 7.6. Plot of observations, support vectors, and classifications for linear kernel.

Employed wives are identified as coming from households with lower incomes but who have higher expected wages (in log units). The diagonal decision boundary implies that an additive model has been applied. Overall, the story seems sensible enough, but perhaps a nonlinear kernel can do better.

	Predict Unemployed	Predict Employed	Model Error
Unemployed	110	64	.37
Employed	52	174	.23
Use Error	.32	.19	Overall Error = .29

Table 7.2. SVM Confusion table for employment: radial kernel, $cost = 1$, $\gamma = .50$, 275 support vectors.

7.4.2 SVM Analysis with a Radial Kernel

Table 7.2 shows how well SVM classifies when a nonlinear kernel is used, in this case the radial kernel. The radial kernal is popular because it is relatively easy to understand and seems to perform well. Specifically, we use

$$K(x_i, x_j) = e^{-\gamma|x_i - x_j|^2}. \tag{7.16}$$

The distance between a pair of observations is squared, multiplied by $-\gamma$, and exponentiated. Because of the negative sign, for a given value of γ, larger distances between observations have smaller kernel values. It also follows that larger values of γ for a given distance have smaller kernel values. In other words, smaller values of γ make differences in the distances between observations in the predictor space more important; observations are spread out more with respect to one another. Therefore, γ can be a useful tuning parameter.

For Table 7.2, the same tuning parameters are as before, but γ is set to the default value of .5. Somewhat fewer support vectors are found implying that fewer observations are difficult to classify. Overall, there is a dramatic improvement in fitting skill, implying the linear kernel was missing important relationships between the two predictors and the response that were uncovered by the radial kernel.

The overall proportion of cases misclassified is .29. The SVM model misclassifies unemployed wives 37% of the time and employed wives 23% of the time. The better performance, compared to the linear kernel comes from a substantial improvement in the ability to correctly classify unemployed wives.

But the improved classification skill comes at a high price. The classification plot shown in Figure 7.7 is challenging to interpret. Wives who fall in the middle ranges of the log of expected wages tend to be unemployed almost regardless of household income. In other words, a U-shaped relationship between the log of expected wages and employment surfaces that holds across most income levels. Wives with relatively high or relatively low expected wages are more likely to be employed than wives with middling expected wages. Why wives with relatively low expected wages behave as do wives with relatively high expected wages is not apparent and perhaps reflects the role of predictors not included in the model. The exception to this pattern is found when household income is very low. Then, wages do not seem to matter at all. Wives from very low income households are classified as employed.

7.4.3 Varying Tuning Parameters

It usually important to vary the SVM tuning parameters to see which give the best results. Hsu and his colleagues (2007) suggest trying a range of cost values between 2^{-5} and 2^{15}, and γ values between 2^{-15} and 2^{3}, ideally in a highly systematic manner. Over such a wide range of values, there will likely be dramatic changes in performance.

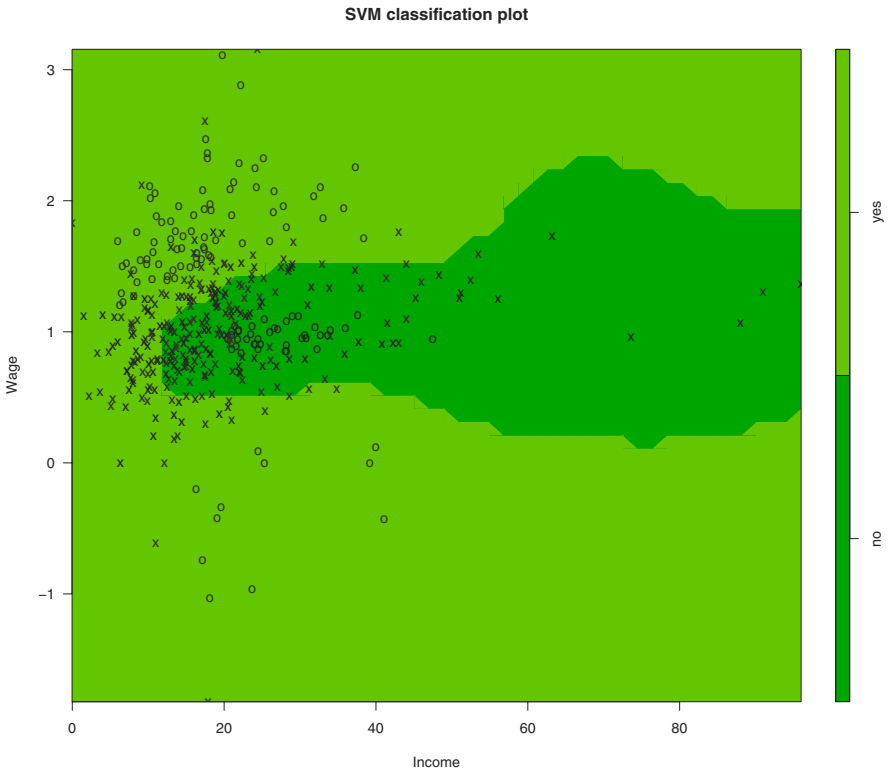

Fig. 7.7. Plot of Observations, Support Vectors, and Classifications for Radial Kernel.

The overall classification skill shown in Table 7.2 is somewhat robust to variation in the values for cost. Essentially, the same overall classification skill materializes with values for cost between .1 and 10,000. However, the ratio of false positives to false negatives can change enough to matter and the classification plots can change substantially. In particular, the unemployment "hole" changes shape and size as the cost parameter increases. And with costs much above 10, the classification plot begins to fragment in a manner that makes little substantial sense. Figure 7.8 shows one example when costs are set at 10. Overall classification is not materially affected because the same region, located to the center left of the plot, contains largely the same observations classified the same way regardless. Put another way, the shape and size change most where there are very few observations. One important lesson is much like that learned for smoothers. Great caution is required if one tries to draw substantive conclusions in regions where there are few data.

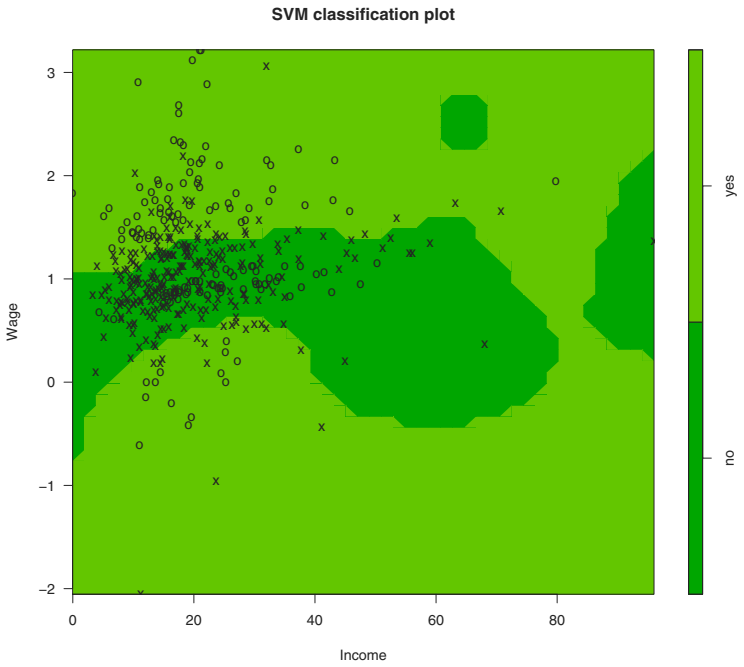

Fig. 7.8. Plot of Observations, Support Vectors, and Classifications for Radial Kernel with cost=10

Varying γ between .0001 and 10 can have very important effects. Until a value for γ of about .005 is reached, there is virtually no classification skill beyond the marginal distribution. Useful classification skill can be obtained thereafter, but γ values larger than about 5 lead again to fragmentation of the classification plots that is very difficult to interpret. Figure 7.9 is a representative illustration.

There is no formal justification for the ranges of cost and γ values tried and unfortunately, there seems to be no principled way to choose among the results that can follow by varying the values of the SVM tuning parameters. One has the option of trial and error with some measure of classification skill as the criterion. Indeed, the procedure tune.svm() will undertake a grid search over specified values of several tuning parameters with some overall measure of performance the arbiter. A key problem with such trial-and-error approaches is that substantive concerns have no direct way to contribute. One risks results that are substantive nonsense. Another concern is that one cannot easily take into account differential classification skill for the true positives and true negatives. Finally, overtuning is encouraged. Even with test data, repeated trials over a large grid will soon lead to a model tuned to the test

Fig. 7.9. Plot of observations, support vectors, and classifications for radial kernel with γ=5.

data. The same dangers exist with cross-validiation or bootstrap approaches.

7.4.4 Taking the Costs of Classification Errors into Account

We have not yet addressed the issue of how to value false positives and false negatives. In part because of the concern with overall classification accuracy, there is little discussion in the SVM literature about taking the costs of false positives and false negatives into account. But in svm(), one can manipulate the number of false negatives and false positives in a confusion table by weighting the response categories. In the spirit of CART and random forests, one can implicitly alter the prior distribution of the response by weighting the data.

The prior distribution of the response variable has 43% of the wives unemployed and 57% of the wives employed. Suppose one gives unemployed wives a weight of .60 and unemployed wives a weight of .40. The intent is to change

the 43–57 marginal distribution into something closer to 60–40. One might employ weights because of what the proper marginal distribution should be (e.g., to correct for oversampling employed wives) or to alter the relative number of false positives and false negatives so that their ratio is more responsive to policy considerations.

	Predict Unemployed	Predict Employed	Model Error
Unemployed	154	17	.10
Employed	98	131	.43
Use Error	.39	.11	Overall Error = .29

Table 7.3. SVM confusion table for employment with weighting: radial kernel, $cost = 1$, $\gamma = .50$, 268 support vectors.

Table 7.3 shows the result. The overall amount of forecasting error has not changed materially, but predictive skill has shifted toward unemployed wives. We are doing better with unemployed wives and worse with employed wives. Consistent with this, the implicit relative costs of false negatives and false positives has changed substantially. Without weighting, the costs of false negatives and false positives were about the same. There were 64 incorrectly classified unemployed wives and 52 incorrectly classified employed wives. Now, there are 17 incorrectly classified unemployed wives and 98 incorrectly classified employed wives. In short, weighting may in practice be a useful approach for altering the prior distribution of the response and the relative costs of false positives and false negatives. For this simple analysis, however, there seems to be no clear rationale for choosing a particular prior other than the empirical prior, which is the implicit default.

7.4.5 Comparisons to Logistic Regression

One might wonder how SVM compares to a logistic regression analysis of the same data. If a radial SVM kernel is used with the default values of the tuning parameters, logistic regression's overall forecasting accuracy is about 50% worse than SVM's when both SVM and logistic regression are evaluated with the test data ($N = 353$). One explanation might be that there are important nonlinear relationships between the two predictors and the response, and SVM is able to find them. Logistic regression cannot find them because logistic regression is committed to whatever functional forms one builds in. Here a simple additive model is used. Such an interpretation is consistent with the earlier comparisons between the SVM results with a linear kernel and a radial kernel.

Alternatively, the nonlinear relationships that SVM finds might be papering over the impact of omitted variables. In other words, although SVM is

better able to link the two predictors to the response, the true roles of the two predictors may be misrepresented in the classification plots. Recall that just this concern was raised earlier. Given the two predictors, SVM is able to show in the classification plots how those predictors are related to the decision boundary. But one must be cautious about attributing to these two variables substantive relationships that may really be in part a function of predictors not included in the analysis. If a richer set of predictors were available, the performance differences between SVM and logistic regression might be reduced. Put another way, a two-predictor description may look very different from a description using more than two predictors.

Consider a somewhat more complicated analysis. The response variable is the same as before but there are now five additional predictors: (1) the number of children in the household 5 years of age or younger, (2) the number of children in the household 6 to 18 years old, (3) whether the wife attended college, (4) whether the husband attended college, and (5) the wife's age.

	Predict Unemployed	Predict Employed	Model Error
Unemployed	89	65	.42
Employed	51	148	.26
Use Error	.36	.30	Overall Error = .32

Table 7.4. Logistic regression confusion table for forecasting employment.

Table 7.4 shows the confusion table for the logistic regression analysis. The table is constructed from the test data. Table 7.5 shows the confusion table for the SVM analysis. It too is constructed from the test data. The overall forecasting error is the same. There are modest differences in forecasting skill for employed wives and unemployed separately. Logistic regression does somewhat better with employed wives and SVM does somewhat better with the unemployed wives. There are related differences in the ratio of false positives to false negatives. But on its face, it is not apparent which model is preferable. One possible implication is that once a richer set of predictors is used, the linear relationships imposed by logistic regression are sufficient. There are no important nonlinearities remaining, at least as a function of the predictors available.

It would be useful if one could examine a classification plot for the SVM analysis. Unfortunately, on this svm() stumbles. Because there are seven predictors, the space defined by the predictors is in seven dimensions. What svm() offers is to plot any plane in that space. There are in practice a very large number of such planes (in theory a limitless number) and no principled way to determine which should be examined. Moreover, if the goal is to understand how predictors are related to the response, a plane through a high-dimensional space is not likely to be helpful unless the decision boundary is very simple.

	Predict Unemployed	Predict Employed	Model Error
Unemployed	111	43	.28
Employed	71	128	.36
Use Error	.39	.25	Overall Error = .32

Table 7.5. SVM Confusion Table for Forecasting Employment: $cost = 1$, $\gamma = .125$, 278 Support Vectors

More likely, the decision boundary will change in important ways from plane to plane in ways that will defy any simple explanation.

One possible solution would be to construct marginal plots, in the spirit of partial dependence plots, by integrating out all but one predictor dimension at a time. Such plots would be far more interpretable but at the price of throwing out a lot of information. In particular, interaction effects would be lost unless interaction variables are constructed and included among the predictors.

In summary, SVM has its roots in classification problems so that classification accuracy has been a driving concern. Less attention has been paid to representing how inputs are related to outputs so that from a regression perspective, SVM has some serious limitations. However, there seems to be no reason in principle why analogues to importance plots and partial dependence plots could not be developed, and it is likely that they will be introduced in the near future.

7.5 Software Considerations

There are several free software packages available to download that have good implementions of SVM. There are also lots of free modules that can be downloaded and cobbled together (see http//www.kernel-machines.org/). In R, the procedure SVM (in the R library *e1071*) implements four different kernel functions, two kinds of classification procedures and two kinds of regression procedures. Initial experience with the software is good. It runs well and seems to be largely bug-free. Two other SVM programs in R that seem to work satisfactorily are in the library *svmpath* and *ksvm*. The discussion that follows concentrates on the SVM implementation in *e1071*, but the comments apply more generally.

For reasons discussed earlier, it is not easy to learn from SVM how inputs are related to outputs. The SVM software in R relies on plots much the same as those used in this chapter. Multivariate relationships are, therefore, very difficult to visualize and in addition, one is effectively limited to quantitative predictors. There is also no direct help in determining predictor importance.

A second difficulty with the software is that the user must specify the kernel function. As noted earlier, guidance is pretty thin on which kinds of kernel functions work better for which kinds of data. Typically, there is a bit

of theory and some simulations suggesting that if the data are generated by a certain kind of stochastic process, one kind of kernel may perform better than another kind of kernel. However, in practice one usually has little idea what stochastic process has generated the data, so such information is not very helpful. In effect, the kind of kernel becomes a tuning parameter.

A third difficulty is that the software requires at least one tuning parameter, and usually two. One must always specify the value of the tuning parameter that determines the weight given to the slack observations. However, there is usually no clear way to link that value to subject matter or decision-making concerns. For most kernels, a second tuning parameter is needed, whose relationships to the data and the goals of the analysis are even more distant.

The current advice, at least for "beginners," is to proceed by trial and error using a cross-validation measure of generalization error (Hsu et al., 2007). For each combination of tuning parameters, a cross-validation measure of performance is computed with the training data. Eventually, the set of values with the smallest generalization error is selected. These are then applied when SVM is used on the full set of training data to arrive at final results. Alternatively, the trial-and-error process can be automated with tune.svm(), as noted earlier

This strategy has several related problems. To begin, the informal grid search strategy recommended has a large number of grid points. The strategy is time consuming. An automated approach can help a lot in cutting the amount of human labor required, but substantive considerations get short shrift.

In addition, the application of a cross-validation measure of performance is usually often a good idea and in principle, overfitting can be usefully addressed. However, the grid search procedure goes back to the same well many times, so the assets of cross-validation are gradually dissipated. The same data are being recycled over and over. Similar problems affect the use of test data and the bootstrap. And overtuning leads to overfitting.

Also, the benchmark for the grid search procedure is some estimate of overall forecasting error. This ignores potentially important differences between the errors that result from trying to forecast "successes" and the errors that result from trying to forecast the "failures." Misleading results can follow. At the very least, important information is not taken into account.

A final difficulty with the software is that there is no direct way to address the relative costs of false negatives and false positives. The software provides a method to reweight the data to adjust for unbalanced response variable distributions and it seems that the weights can be used much as a prior distribution. In effect, the weighting becomes another tuning parameter. But, the formal justification for this approach is thin and one could in principle try to address differential costs with combinations of other tuning parameters.

Perhaps the best advice at this point is to undertake the data analysis with statistical learning procedures that have the requisite tools for examining the output and that seem to be easier to implement in an informed

manner. Then, SVM can be applied to the same data. In the comparisons across statistical learning tools, the performance of SVM may be better understood and appreciated. Perhaps then it can be used in a more informed manner.

7.6 Summary and Conclusions

Support vector machines has some real strengths. SVM was developed initially for classification problems and seems to perform 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.

However, SVM was not developed to capture the $f(X)$ and therefore, does not fit well within a regression framework in which one cares about how predictors are related to a response. It also stumbles badly as an exploratory tool to inform future work on some $f(X)$. Finally, although SVM can often fit the data at least as well as random forests and stochastic gradient boosting, it is rare to find applications for which it performs dramatically better.

At the same time, it is difficult to arrive at an overall assessment. Support vector machines is still evolving rapidly with new kernel functions and computer algorithms appearing on a regular basis. The tradeoffs between the various kernels and between them and alternative statistical learning procedures is not at all clear. In short, it is difficult at this time to make a strong case one way or another for support vector machines compared to procedures such as random forests and boosting even if the primary goal of the primary data analysis is to fit the response as well as possible.

Exercises

Problem Set 1

Construct a dataset as follows. You will need to load the *Rlab*.

```
w<-rnorm(500)
z<-rnorm(500)
w2<-w^2
x<-(-1+3*w2-1*z)
p<-exp(x)/(1 + exp(x))
library(Rlab)
y<-rbern(500,p)
```

As you proceed, you will need to read carefully the help commands for the various procedures to determine which require that y be a factor, and the form in which the data are expected.

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. Repeat the two analyses using svm() from the library *e1071*. Use the default settings. First regress y on w and z and then regress regress y on $w2$ and z. Compare the confusion tables and classification plots from the two analyses. How does the output from the two models differ? Why?

3. What do you answers to the first two question tell you how about how SVM responds to nonlinear relationships that are not introduced explicitly in the predictors used?

Problem Set 2

From the *MASS* library, load the Pima.tr dataset. The variable "type" is the response. All other variables are predictors. Apply svm() to the data (in library *e1071*). Use all of the predictors and the defaults. Study the output by constructing a confusion table and some classification plots. There are a very large number of possible classification plots. To get a feel for what they can reveal, use"glu" and "bp" to define the two axes and set the other predictors at their means. So, you get to see the decision boundary for "glu" and "bp" with all other variables fixed at their average values. Now see what happens when the predictors are set at their first quartile and then their third quartile. There is no "right" classification plot. Each shows a different part of the decision boundary in different locations of the predictor space.

1. Apply svm() to the data (in library *e1071*). Use all of the predictors and the defaults. Study the output by constructing a confusion table and a classification plot. Use the first plot constructed above. Now change the default radial kernel to a linear kernel. Study the output by constructing a confusion table and a classification plot. Compare the output from the radial kernel to the output from the linear kernel. How are they much the same and how are they somewhat different. Why?

2. For the response variable type, about a third of the observations are "yes" and about two-thirds are "no." Try imposing a prior with the proportions reversed. For example, the weights can be constructed with

```
wts=table(type)
wts[1]=.3
wts[2]=.7
```

Repeat the earlier analysis with the radial kernel. Also consider the ratio of false positives to false negatives. How have things changed? Are the changes consistent with the alternation in the attempt to alter the prior? Look at both a confusion table and a classification plot. For the latter, use"glu" and "bp" to define the two axes and set the other predictors at their means.

3. With the weighting as before (the default), repeat the analysis with cost parameter values of .1 and 10. How have things changed? Consider both a confusion table and a classification plot. For example, what does the classification plot convey when the cost is set at 10?

4. With the weighting as before and the cost parameter set to 1, try a values of γ of .01, .5, 1, and 5. How have things changed? Consider both a confusion table and a classification plot as before. How does the output change? From the confusion table alone, which value of γ produces the most accurate classifications? Is this the best model? Why or why not?