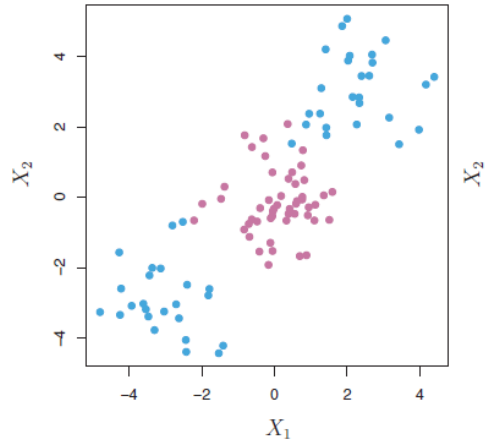
BUAD5082
Machine Learning II

Support Vector Machines
(ISLR, Chapter 9)

Classification with Non-linear Decision Boundaries



- The support vector classifier is a natural approach for classification in the two-class setting, if the boundary between the two classes is linear.
- But in practice we are often faced with non-linear class boundaries. Consider the data in the figure above.
  - A support vector classifier or any linear classifier will perform poorly with this data set.

Classification with Non-linear Decision Boundaries

- When discussing linear regression, we accommodated non-linear decision boundaries by enlarging the feature space using functions of the predictors, such as quadratic and cubic terms, interactions, etc.
- In the case of the support vector classifier, we could address the problem of possibly non-linear boundaries between classes in a similar way.

Classification with Non-linear Decision Boundaries

- For instance, rather than fitting a support vector classifier using p features $X_1, \ldots, X_p$, we could instead fit a support vector classifier using $2p$ features $X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$.
- Then our optimization problem would become

$$\underset{\beta_0, \beta_{11}, \beta_{12} \ldots, \beta_{p1}, \beta_{p2}, \epsilon_1, \ldots, \epsilon_n}{\text{maximize}} \quad M$$

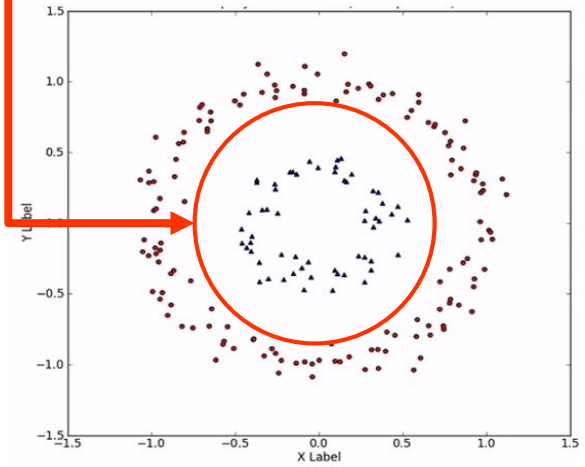
$$\text{subject to } y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i),$$

$$\sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1.$$

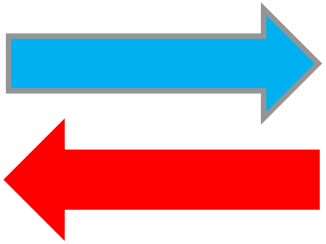
Classification with Non-linear Decision Boundaries

- Why does this lead to a non-linear decision boundary?
- In the enlarged feature space, the decision boundary that results from the optimization problem is in fact linear.
- But in the original feature space, the decision boundary is non-linear.

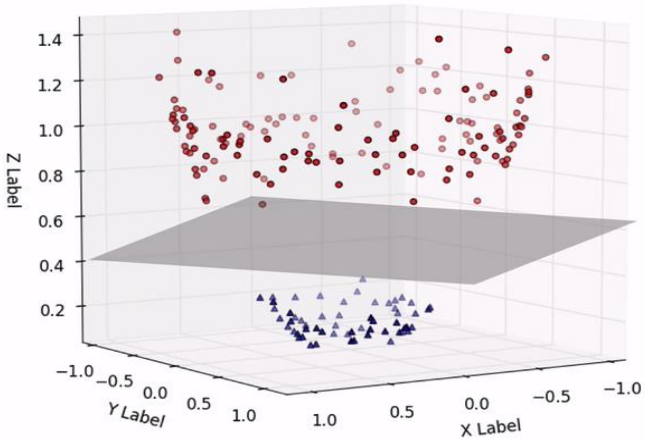
When dealing with a non-linear solution in the original space spanned by the original set of features…



…we can often enlarge the feature space to transform it into a higher dimensional space where there may be a satisfactory linear solution.



…which provides a non-linear solution in the original space..



The Support Vector Machine

- The *support vector machine* (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using ***kernels***.
- The ***kernel approach*** that we discuss next is simply an efficient computational approach for enacting this idea.

The Support Vector Machine

- ***The Linear Kernel***: Recall that the inner product (or "dot product", which are equivalent in $\mathbb{R}^n$) of two vectors with *r* elements *a* and *b* is defined as
$$\langle a, b \rangle = \sum_1^r a_i b_i$$
(the "sumproduct" of the pairwise elements)
  - So the inner product of two observations $x_i, x_{i'}$ is given by $\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j}$
- In our discussion of the Maximal Margin Classifier, we showed that the calculation of the solution involves only the *inner products* of the observations (as opposed to some more complex expression involving the observations), and the representation of the decision boundary involves only the *inner products* of the observations that are the support vectors.

The Support Vector Machine

- Recall also that if x_i and $x_{i'}$ are standardized ($\mu = 0$, $\sigma = 1$) then the inner product $\langle x_i, x_{i'} \rangle$ is the correlation coefficient divided by n - 1.
  - Note here that we are standardizing the observations, not the variables (see sheet InnerProduct in MaximalMarginAndSupportVectorClassifiers.xlsm).
- That is, the inner product is just a scaler multiple of the standardized correlation coefficient, and is therefore a measure of similarity.
- Si in our current context, the inner product of two observations is just a measure of their similarity.

The Support Vector Machine

- With only minor changes to that argument, we can show that the linear Support Vector Classifier can be represented as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

where there are n parameters α_i, $i = 1, \ldots, n$, one per training observation.

Recall our earlier discussion indicating that the Maximal Margin classifier optimization problem can we re-written as

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0.$$

- To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and β_0, all we need are the $\binom{n}{2}$ inner products $x_i, x_{i'}$ between all pairs of training observations.
  - And in order to evaluate the function $f(x)$, we need only compute the inner product between the new point x and each of the training points x_i.
- However, it turns out that α_i's are nonzero only for the support vectors in the solution.
- So if S is the collection of indices of these support vector points, we can rewrite any solution function as

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

Only the observations that are support vectors are required to specify the decision boundary.

The Support Vector Machine

- Suppose that every time the inner product $\sum_{j=1}^{p} x_{ij} x_{i'j}$ appears in the representation of our linear classifier or in a calculation of the solution for the support vector classifier, we replace it with a *generalization* of the inner product of the form

$$K(x_i, x_{i'}),$$

where K is some function that we will refer to as a ***kernel***.

  - A kernel is a function that quantifies the similarity of two observations (just as the inner product does).

- For instance, we could simply take

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j},$$

which would just give us back the support vector classifier.

  - This expression is known as a ***linear kernel*** because the support vector classifier produces a linear decision boundary; the linear kernel essentially quantifies the similarity of a pair of observations using the Pearson (standard) correlation coefficient.

The Support Vector Machine

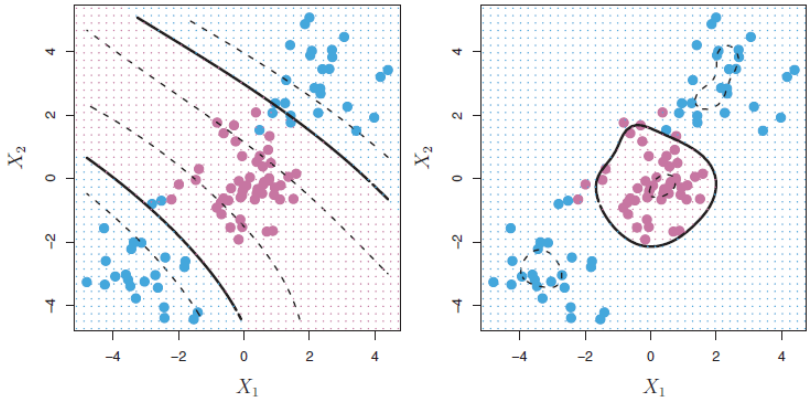
- But one could instead choose another form. For instance, one could replace every instance of $\sum_{j=1}^{p} x_{ij} x_{i'j}$ with

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d,$$

  This is known as a ***polynomial kernel*** of degree d, where d is a positive integer.
- Using such a kernel with $d > 1$, instead of the standard linear kernel, in the support vector classifier algorithm leads to a much more flexible decision boundary.
- When the support vector classifier is combined with a non-linear kernel such as the one above, the resulting classifier is known as a ***Support Vector Machine***.
- In this case the (non-linear) function has the form

$$f(x) = \beta_0 + \sum_{i \in S} \beta_i K(x, x_i)$$

The Support Vector Machine



- The left-hand panel above shows an example of an SVM with a polynomial kernel applied to the non-linear data from an earlier example.
  - The fit is a substantial improvement over the linear support vector classifier.
- Another popular kernel choice is the ***radial kernel***, which takes the form

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right) = e^{-\gamma \|x_i - x_{i'}\|^2}$$

Note that $\|x_i - x_{i'}\|^2$ is the squared Euclidean distance between the two observations x_i and $x_{i'}$

where γ is a training constant that controls the bias-variance tradeoff.

- The right-hand panel above shows an example of an SVM with a radial kernel on this non-linear data; it also does a good job in separating the two classes.

The Support Vector Machine

- How does the radial kernel actually work? If a given test observation $x^* = (x_1^*, x_2^*, \ldots, x_P^*)^T$ is far from a training observation x_i in terms of Euclidean distance, then $\|x_i - x_{i'}\|$ will be large, and so
$$K(x^*, x_i) = \exp(-\gamma \|x_i - x_{i'}\|^2)$$
will be very tiny.
- This means that x_i will play virtually no role in $f(x^*)$.
  - Recall that the predicted class label for the test observation x^* is based on the sign of $f(x^*)$.
    - In other words, training observations that are far from x^* will play essentially no role in the predicted class label for x^*.
- This tells us that the radial kernel has very *local* behavior, in the sense that only nearby training observations have an effect on the class label of a test observation.

The Role of γ In the Radial Kernel

- The training parameter γ controls the bias-variance trade-off.
  - Recall that the decision rule for a SVM with a radial kernel is determined by the sign of

$$f(x) = \beta_0 + \sum_{i \in S} \beta_i K(x, x_i) = \beta_0 + \sum_{i \in S} \beta_i e^{-\gamma \|x_i - x\|^2}$$

Large values of γ decrease the influence of distant support vectors x_i on the sign of $f(x)$, and conversely.

  - When evaluating the sign of an observation x, the fewer support vectors that exert an influence on $f(x)$ the higher the model variance will be (and the lower the bias will be).
  - For a point x that is distant from a support vector x_i, a small value of γ will make $\gamma \|x_i - x\|^2$ smaller, the therefore make the kernel function larger, meaning that the distant support vector x_i will exert more influence on the classification of the training example than if γ were large.
    - So small γ means more support vectors influence the sign of $f(x)$, and therefore the model will have lower variance but higher bias.
    - Conversely, is γ is large, the model will exhibit higher variance but will introduce less bias.
  - So
    - Small γ means large influence of distant support vectors
      - which means more support vectors participate…
      - which means lower variance and higher bias
    - Large γ means small influence of distant support vectors
      - which means fewer support vectors participate…
      - which means higher variance and lower bias

The Gaussian Kernel

- Recall the definition of the Normal (Gaussian) distribution:

$$f_i(x) = \frac{1}{\sqrt{2\pi\sigma_i}} exp\left(-\frac{1}{2{\sigma_i}^2}(x-\mu_i)^2\right)$$

- The Gaussian kernel is defined as follows:

$$K(x^*, x_i) = exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{p}\left(x_{ij}^* - x_{ij}\right)^2\right)$$

- Note the similarity to the radial kernel – we have simply replaced the training constant γ with $\frac{1}{2\sigma^2}$, both of which must be set by the modeler.
- Thus, while a large γ in the radial kernel risks high variance, the opposite is true for the role of σ in the Gaussian kernel:
  - So
    - Small σ means small influence of distant support vectors
      - which means fewer support vectors participate…
      - which means higher variance and lower bias
    - Large σ means large influence of distant support vectors
      - which means more support vectors participate…
      - which means lower variance and higher bias

The Support Vector Machine

- What is the advantage of using a kernel rather than simply enlarging the feature space using functions of the original features, as we did with linear regression?
- One advantage is computational, and it amounts to the fact that to solve the optimization problem using kernels, one need only compute $K(x_i, x_{i'})$ for all $\binom{n}{2}$ distinct pairs, i, i', which can be done without explicitly working in the enlarged feature space.
- This is important because in many applications of SVMs, the enlarged feature space is so large that computations would otherwise be intractable.

Which Kernel Should One Use?

- When to use a linear kernel (a.k.a. no kernel)
  - Works best when p is relatively large and n is relatively small
  - In such cases, variance is the major enemy, so a high-bias approach is usually required.
  - Example: text analytics (say spam prediction), where features correspond to the individual words in a corpus (p = 10,000?) and an observation is a document in the corpus. The predictors are y/n for each word, which can be very sparse in the yes's.
  - And there may only be n = 1,000 documents in the training set.

Which Kernel Should One Use?

- When to use a Gaussian kernel (one of several radial basis functions, or RBFs)
  - When *n* is relatively large (50,000 to millions) and *p* is small relative to *n*.
  - In such cases, variance is not major concern, so a low-bias (and therefore highly flexible) approach usually produces better outcomes.
  - However, computational tractability is a major concern here. As *n* increases, computational constraints may force the use of a linear kernel, or the use of Logistic Regression.
    - Note that Neural Nets are also highly applicable here. SVM's tend to be much faster than Neural Nets, and since the optimization problem being solved is convex, we can always find a global maximum. Local maxima are problematic with NN's.

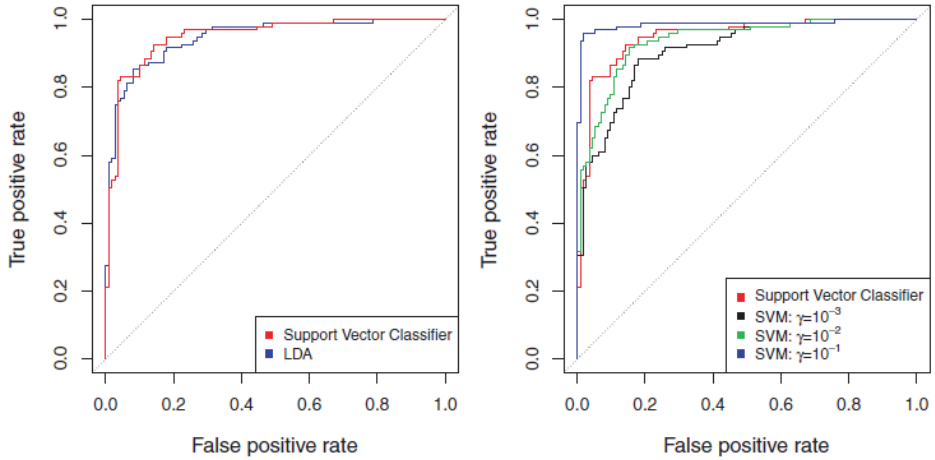
Which Kernel Should One Use?

- Other Kernels
  - In practice, although many other kernels exist, they are very seldom used for SVM's.
  - Examples:
    - String kernel (for text)
    - Circular kernel
    - B-Spline kernel
  - See http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/#kernel_choosing for an extensive list of kernels.

Example: Heart Disease Data

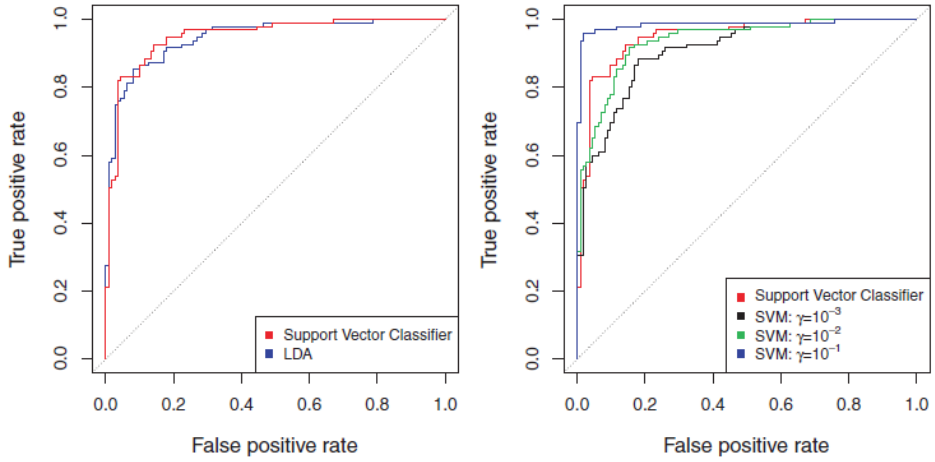
- Consider the following situation involving Heart data.
- The aim is to use 13 predictors such as Age, Sex, and Chol in order to predict whether an individual has heart disease.
- We now investigate how an SVM compares to LDA on this data.
- After removing 6 missing observations, the data consist of 297 subjects, which we randomly split into 207 training and 90 test observations.
- We first fit LDA and the support vector classifier to the training data.
  - Note that the support vector classifier is equivalent to a SVM using a polynomial kernel of degree $d = 1$.

Example: Heart Disease Data



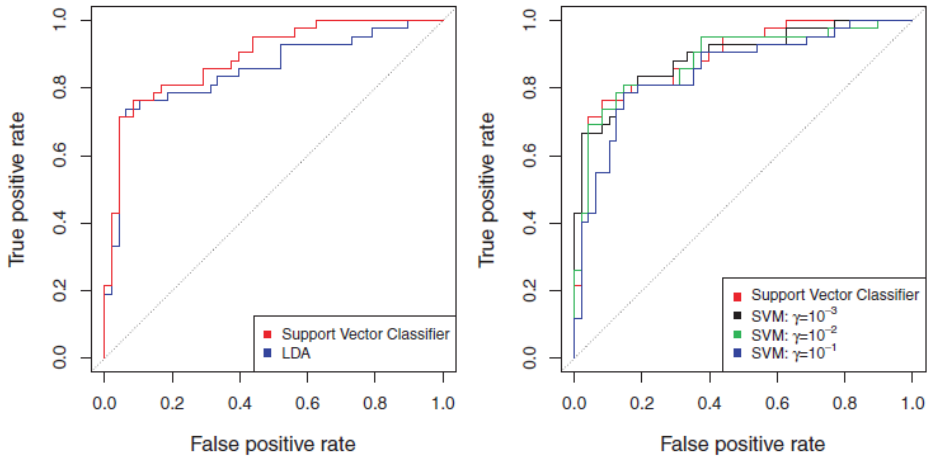
- The left-hand panel displays ROC curves for the training set predictions for both LDA and the support vector classifier.
- Both classifiers compute scores of the form
  $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p$ for each observation.
- For any given cutoff *t*, we classify observations into the *heart disease* or *no heart disease* categories depending on whether $\hat{f}(X) < t$ or $\hat{f}(X) > t$.
- The ROC curve is obtained by forming these predictions and computing the false positive and true positive rates for a range of values of *t*.
- An optimal classifier will hug the top left corner of the ROC plot. In this instance LDA and the support vector classifier both perform well, though there is a suggestion that the support vector classifier may be slightly superior.

Example: Heart Disease Data



- The right-hand panel displays ROC curves for SVMs using a radial kernel, with various values of γ.
- As γ increases and the fit becomes more non-linear, the ROC curves improve. Using $\gamma = 10^{-1}$ appears to give an almost perfect ROC curve.
- However, these curves represent training error rates, which can be misleading in terms of performance on new test data.

Example: Heart Disease Data



- These plots display ROC curves computed on the 90 test observations.
- We observe some differences from the training ROC curves.
  - In the left-hand panel, the support vector classifier appears to have a small advantage over LDA (although these differences are not statistically significant).
  - In the right-hand panel, the SVM using $\gamma = 10^{-1}$, which showed the best results on the training data, produces the worst estimates on the test data.
- This is once again evidence that while a more flexible method will often produce lower training error rates, this does not necessarily lead to improved performance on test data.
- The SVMs with $\gamma = 10^{-2}$ and $\gamma = 10^{-3}$ perform comparably to the support vector classifier, and all three outperform the SVM with $\gamma = 10^{-1}$.

SVMs with More than Two Classes

- So far, our discussion has been limited to the case of binary classification.
- The concept of separating hyperplanes upon which SVMs are based does not lend itself naturally to more than two classes.
- A number of proposals for extending SVMs to the *K*-class case have been made. The two most popular are the *one-versus-one* and *one-versus-all* approaches, which are described briefly here.

SVMs with More than Two Classes

- One-Versus-One Classification when $K > 2$.
  - This approach constructs $\binom{K}{2}$ SVMs, each of which compares a pair of classes.
    - For example, one such SVM might compare the kth class, coded as +1, to the k'th class, coded as -1.
  - We classify a test observation using each of the $\binom{K}{2}$ classifiers, and we tally the number of times that the test observation is assigned to each of the K classes.
  - The final classification is performed by assigning the test observation to the class to which it was most frequently assigned in these $\binom{K}{2}$ pairwise classifications.

SVMs with More than Two Classes

- One-Versus-All Classification
  - This approach fits *K* SVMs, each time comparing one of the *K* classes to the remaining $K - 1$ classes.
  - Let $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the *k*th class (coded as +1) to the others (coded as −1).
  - Let x^* denote a test observation.
  - We assign the observation to the class for which $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \ldots + \beta_{pk}x_p^*$ is the largest, as this amounts to a high level of confidence that the test observation belongs to the *k*th class rather than to any of the other classes.

Relationship to Logistic Regression

- When SVMs were first introduced in the mid-1990s, they made quite a splash in the statistical and machine learning communities due in part to their good performance, but also due to the fact that the underlying approach seemed both novel and mysterious.
  - In the case of Support Vector Classifiers, for example, the idea of finding a hyperplane that separates the data as well as possible, while allowing some violations to this separation, seemed distinctly different from classical approaches for classification, such as logistic regression and linear discriminant analysis.
  - In the case of SVMs, the idea of using a kernel to expand the feature space in order to accommodate non-linear class boundaries appeared to be a unique and valuable characteristic.

Relationship to Logistic Regression

- However, since that time, deep connections between SVMs and other more classical statistical methods have emerged.
- It turns out that one can rewrite the criterion for fitting the support vector classifier $f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$ as

$$\underset{\beta_0, \beta_1, \ldots, \beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where λ is a nonnegative tuning parameter.

Relationship to Logistic Regression

- That is, the original formulation…

$$\underset{\beta_0, \ldots, \beta_p, \epsilon_1, \ldots, \epsilon_n}{maximize} \; M$$

$$subject\ to \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \qquad \sum_{i=1}^{n} \epsilon_i \leq C$$

Note that this term is the ridge penalty term from earlier discussions, and plays a similar role in controlling the bias-variance trade-off for the support vector classifier.

is equivalent to the re-formulation

$$\underset{\beta_0, \ldots, \beta_p}{minimize} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where λ is a nonnegative tuning parameter.

Relationship to Logistic Regression

- That is, the original formulation…

$$\underset{\beta_0,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{maximize} \; M$$

$$subject\ to \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \qquad \sum_{i=1}^{n} \epsilon_i \leq C$$

Small values for *C* in the original formulation, and λ in the re-formulation, result in few violations to the margin; this amounts to a high-variance but low-bias classifier.

is equivalent to the re-formulation

$$\underset{\beta_0,\ldots,\beta_p}{minimize} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}\right)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

where λ is a nonnegative tuning parameter.

Relationship to Logistic Regression

- Now this expression takes a "Loss + Penalty" form that appears repeatedly in machine learning methods:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \{L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta)\}$$

- Here $L(\mathbf{X}, \mathbf{y}, \beta)$ is some loss function quantifying the extent to which the model, parametrized by β, fits the data $(\mathbf{X}, \mathbf{y})$, and $P(\beta)$ is a penalty function on the parameter vector β whose effect is controlled by a nonnegative tuning parameter λ.

Relationship to Logistic Regression

- For instance, Ridge Regression and the Lasso both take this form with

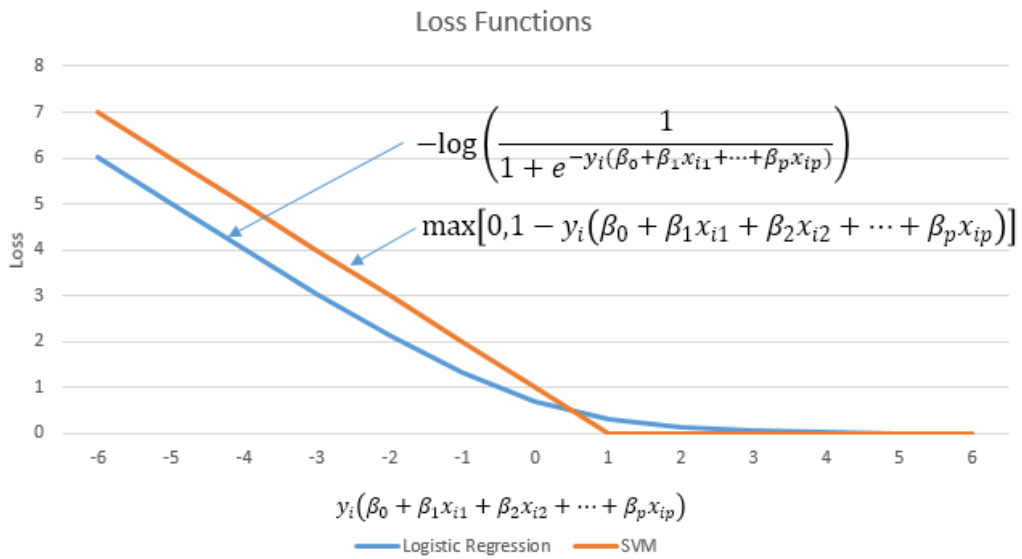
$$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

  with $P(\beta) = \sum_{j=1}^{p} \beta_j^2$ for ridge regression and $P(\beta) = \sum_{j=1}^{p} |\beta_j|$ for the lasso.

- In the re-formulation, the loss function instead takes the form

$$L(X, y, \beta) = \sum_{i=1}^{n} \max[0, 1 - y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip})]$$

Relationship to Logistic Regression



When $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 1$, the SVM loss is zero since the observation is on the correct side of the margin.

- This is known as ***hinge loss***, and is depicted in orange in the figure above.
- This hinge loss function is closely related to the loss function used in logistic regression, which is also shown above.
- Observe that the two loss functions have very similar behavior.

Relationship to Logistic Regression

- When the support vector classifier and SVM were first introduced, it was thought that the tuning parameter C was an unimportant "nuisance" parameter that could be set to some default value, say 1.
- However, the "Loss + Penalty" formulation for the support vector classifier indicates that this is not the case. The choice of tuning parameter is very important and determines the extent to which the model underfits or overfits the data, as illustrated earlier.

Relationship to Logistic Regression

- We have established that the support vector classifier is closely related to logistic regression and other preexisting statistical methods.
- Is the SVM unique in its use of kernels to enlarge the feature space to accommodate non-linear class boundaries?
- The answer to this question is “no”. We could just as well perform logistic regression or many of the other classification methods seen in this course using non-linear kernels; this will be closely related to some of the non-linear approaches we will see in our discussions about moving beyond linearity.
- However, for historical reasons, the use of non-linear kernels is much more widespread in the context of SVMs than in the context of logistic regression or other methods.

SVMs for Regression

- Although we have not addressed it here, there is in fact an extension of the SVM for regression (i.e. for a quantitative rather than a qualitative response), called *support vector regression*.
- In our discussion of regression, we saw that least squares regression seeks coefficients $\beta_0, \beta_1, \ldots, \beta_p$ such that the sum of squared residuals is as small as possible. (Recall that residuals are defined as $y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip}$.)
- Support vector regression instead seeks coefficients that minimize a different type of loss, where only residuals larger in absolute value than some positive constant contribute to the loss function.