1 Support Vector Machines

1.1 Maximal Margin Classifier.

In this section, we define a hyperplane and introduce the concept of an optimal separating hyperplane.

1.1. Definition. A **hyperplane** in a p-dimensional space is a flat affine subspace of dimension $p-1$. Mathematically, it is represented by an equation

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0, \tag{1.1}$$

in the sense that *if a point $X = (X_1, \ldots, X_p)^T$ satisfies (1.1), then X lies on the hyperplane*.

1.2. Remark. Observe points that do not lie on the hyperplane must satisfy exactly one of the following:

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p > 0 \quad \text{or} \quad \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p < 0.$$

Thus, we can think of the hyperplane as *dividing p-dimensional space into two halves*. One can easily determine on which side of the hyperplane a point lies by simply calculating the *sign* of the LHS of (1.1).

1.3. Now suppose that we have a $n \times p$ data matrix $\mathbf{X}$ that consists of n training observations in p-dimensional space, i.e., data points $x_1, \ldots, x_n \in \mathbb{R}^p$, and that observations fall into two classes—that is, $y_1, \ldots, y_n \in \{-1, 1\}$ where each represents one of the classes. Suppose further that we also have a test observation, a vector of observed features $x^* = (x_1^*, \ldots, x_p^*)$ of dimensional p. Our goal is to develop a classifier based on the training data that will correctly classify the test observation using its feature measurements. We will present a new approach that is based upon the concept of a *separating hyperplane.*

1.4. Note. Suppose that it is possible to construct a hyperplane that separates the training observations perfectly according to their class labels. Then the hyperplane has the property that for the ith training observation,

$$\beta_0+\beta_1 x_{i1}+\cdots+\beta_p x_{ip} > 0 \text{ if } y_i = 1 \quad \text{and} \quad \beta_0+\beta_1 x_{i1}+\cdots+\beta_p x_{ip} < 0 \text{ if } y_i = -1.$$

Equivalently, the hyperplane has the property that for all $i = 1, \ldots, n$

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) > 0. \tag{1.2}$$

If a separating hyperplane exists, we can classify the test observation x^* based on the sign of $f(x^*) = \beta_0 + \beta_1 x_1^* + \cdots + \beta_p x_p^*$. Moreover, we can make use of the *magnitude* of $f(x^*)$, in the sense that *the further $f(x^*)$ is from zero, the further x^* lies from the separating hyperplane, and thus the more confident about our class assignment for x^* we will be*.

1.5. Remark. If our data is *linearly-separable*, that is, the data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes. To decide which hyperplane to use for the classifier, recall we are more confident about our class assignment for the data points if they lie far away from the hyperplane. This naturally leads to the following definition of a *maximal margin hyperplane*, which is considered to be the *optimal* separating hyperplane.

1.6. Definition. The **margin** of a hyperplane is the minimum distance between the hyperplane and the data points. The **maximal margin hyperplane** is the hyperplane that has the farthest minimum distance to the training observations.

1.7. Remark. An interesting property is that the *maximal margin hyperplane depends only on the vectors that lie closest to it (and not the other points at all)*. Indeed, moving other data points will not affect the separating hyperplane, provided that the data point does not cross the boundary set by the margin. Thus, we would like to give these special vectors a name as well.

1.8. Definition. The p-dimensional vectors that are closest to the separating hyperplane are called **support vectors**, since they "support" the maximal margin hyperplane in the sense that if they were moved slightly then the maximal margin hyperplane would move as well.

1.9. We now see how to construct a maximal margin classifier. Given a set of n training observations $x_1, \ldots, x_n \in \mathbb{R}^p$ and associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$, let $\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$ be the maximum margin hyperplane with margin M. Then for each $i = 1, \ldots, n$, we have $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M$. Using matrix notation, we can express it more compactly as $\mathbf{y}\mathbf{X}\beta \geq M$. If we add one more constraint $\beta^T\beta = 1$, then (one can show) that the perpendicular distance from the hyperplane (to the support vector) is given by $\mathbf{y^T}\mathbf{X}\beta$. Since our goal is to make M as large as possible, the maximal margin hyperplane is thus the solution to the convex optimization problem[*]:

$$\begin{aligned} \max_{\beta_0,\beta_1,\ldots,\beta_p,M} \quad & M \\ s.t. \quad & \beta^T\beta = 1 \\ & \mathbf{y}\mathbf{X}\beta \geq M. \end{aligned}$$

[*]More information can be found in Essentials of Statistical Learning P420. We won't go into much details here.

1.2 Support Vector Classifiers.

1.10. The maximal margin classifier is simple and intuitive, but often times a separating hyperplane does not exist or is not desirable to be used as a classifier. In particular, a classifier based on a separating hyperplane will necessarily perfectly classify all of the training observations, which can lead to sensitivity to individual observations. In this case, we might be willing to consider a classifier based on a hyperplane that does not perfectly separate the two classes, in the interest of

- greater robustness to individual observations, and
- better classification of *most* of the training observations.

For this, we introduce the *support vector classifier* (SVC), also known as a *soft margin classifier*, which is the solution to the following optimization problem:

$$\begin{array}{rl} \max_{\beta_0,\beta_1,\ldots,\beta_p,\varepsilon_1,\ldots,\varepsilon_n,M} & M \\ s.t. & \beta^T\beta = 1 \\ & \mathbf{yX}\beta \geq M(1-\varepsilon) \\ & \varepsilon_1 \geq 0, \sum_{i=1}^n \varepsilon_i \leq C. \end{array}$$

1.11. Let us look at the difference between this optimization problem and the one before. To allow individual observations to be on the wrong side of the margin or the hyperplane, we introduce *slack variables* $\varepsilon_1, \ldots, \varepsilon_n$. These slack variables tells us where the ith observation is located, relative to the hyperplane and relative to the margin:

1. $\varepsilon_i = 0$: the ith observation is on the correct side of the margin;
2. $\varepsilon_i > 0$: the ith observation is on the wrong side of the margin;
3. $\varepsilon_1 > 1$: the ith observation is on the wrong side of the hyperplane.

The *tuning parameter* C bounds the sum of the slack variables. Intuitively, it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate. For example, setting $C = 0$ will force the problem to return a maximal margin hyperplane. In practice, C is generally chosen via cross-validation.

1.12. It turns out that only observations that either lie on the margin or that violate the margin (i.e., be on the wrong side of the margin) will affect the hyperplane, and hence the classifier; we call these observations *support vectors*. In other words, an observation that lies strictly on the correct side of the margin does not affect the SVC. This property of SVC means that it is quite robust to the behavior of observations that are far away from the hyperplane.

1.3 Support Vector Machines.

1.13. So far we have been using linear decision boundaries. In practice, however, we are sometimes faced with non-linear class boundaries. In this section, we discuss how to extend SVC to accommodate this change by enlarging the *feature space* using polynomial functions of the predictors. For instance, rather than fitting a SVC using p features $X_1, \dots, X_p$, we could instead train it using $2p$ features $X_1, \dots, X_p, X_1^2, \dots, X_p^2$.

1.14. The *support vector machine* (SVM) is an extension to the SVC that results from enlarging the feature space in a specific way, using *kernels*. Basically, we want to enlarge our feature space in order to accommodate a non-linear boundary between the classes; the kernel approach is simply an efficient computational approach for enacting this idea.

1.15. Let us revisit the linear case first. It can be shown that [†]:

- The linear SVC can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle , \tag{1.3}$$

  where there are n parameters α_i, $i = 1, \dots, n$, one per training point.
- To estimate the parameters $\alpha_1, \dots, \alpha_n$ and β, all we need are the $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations.
- It turns out that $\alpha_i \neq 0$ only for the support vectors, thus, if $\mathcal{S}$ is the collection of indices of these support vector, we can rewrite (1.3) as

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle . \tag{1.4}$$

  By expanding each of the inner products, it is easy to see that $f(x)$ is a linear function of the coordinates of x. Doing so also establishes the correspondence between the α_i's and the original parameter β_j's.

To summarize, in representing the linear classifier $f(x)$, and in computing its coefficients, all we need are inner products.

1.16. Here comes the extension. Every time the inner product $\langle x_i, x_{i'} \rangle$ appears, we replace it with a *generalization* of the inner product of the form

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

The function K is refer to as a **kernel**, which quantifies the *similarity* of two observations. For instance, if we set K as the standard inner product (also

[†]See ESL P423 for more technical details.

referred to as a *linear* kernel), we will be quantifying the similarity of a pair of observations using Pearson (standard) correlation; the result is simply SVC.

1.17. We could also take

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d, \quad d \in \mathbb{Z}^+. \tag{1.5}$$

This is known as a *polynomial kernel* of degree d. Using a kernel with $d > 1$, we can get a classifier with a more flexible decision boundary. It essentially amounts to fitting a SVC in a higher-dimensional space involving polynomials of degree d, rather than in the original feature space.

1.18. When the SVC is combined with a non-linear kernel such as (1.5), the resulting classifier is known as a SVM. In this case, the function labelled (1.4) has the form

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

1.19. Another popular choice of non-linear kernels is the *radial kernel*:

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right), \quad \gamma \in \mathbb{R}^+.$$

Observe if a test observation x^* is far away from a training observation x, then $\sum (x_j^* - x_j)^2$ is large and $K(x_i, x_{i'})$ is small. Therefore, this kernel has very *local* behavior, in the sense that only nearby training observations have an effect on the class label of a test observation.

1.20. One advantage of using a kernel rather than simply enlarging the feature space using functions of the original features is that the former only requires the computation of $K(x_i, x_{i'})$ for all $\binom{n}{2}$ distinct pairs, while computations in the enlarged feature space might be intractable.

1.21. There are two common approaches to extend SVM to multi-class classifications:

1. **one-versus-one**: Construct $\binom{K}{2}$ SVMs, each of which compares a pair of classes. We classify a test observation using each of the $\binom{K}{2}$ classifiers and assigned the classes that is most frequently assigned to the observation.
2. **one-versus-all**: Constructs K SVMs each comparing one of the classes to the rest $K - 1$ classes. We classify a test observation by assigning it the class with the highest confidence score (hint: $\operatorname{argmax}_{k \in \{1,\ldots,K\}} f_k(x)$).