Support Vector Machines

Support Vector Machines

Support vector machines are a popular method for classification problems where there are two classes.

Support Vector Machines

Support vector machines are a popular method for classification problems where there are two classes.

They have also been extended to regression problems and multi-way classification problems.

Support Vector Machines

Support vector machines are a popular method for classification problems where there are two classes.

They have also been extended to regression problems and multi-way classification problems.

Recall that a **hyperplane** in two dimensions is defined by the equation

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0. \tag{1}$$

Support Vector Machines

Support vector machines are a popular method for classification problems where there are two classes.

They have also been extended to regression problems and multi-way classification problems.

Recall that a **hyperplane** in two dimensions is defined by the equation

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0. \tag{1}$$

This is just a straight line. Any point that satisfies this equation lies on that line.

Support Vector Machines

Support vector machines are a popular method for classification problems where there are two classes.

They have also been extended to regression problems and multi-way classification problems.

Recall that a **hyperplane** in two dimensions is defined by the equation

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0. \tag{1}$$

This is just a straight line. Any point that satisfies this equation lies on that line.

More generally, when there are p dimensions, we can write

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 \ldots + \beta_p x_p = 0. \tag{2}$$

If we form the x_i into a vector $\boldsymbol{x}$, we can also write

$$\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} = 0. \tag{3}$$

If we form the x_i into a vector $\boldsymbol{x}$, we can also write

$$\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} = 0. \tag{3}$$

Every hyperplane divides the space in which it lives into two parts, depending on whether $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} > 0$ or $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} \leq 0$.

If we form the x_i into a vector $\boldsymbol{x}$, we can also write

$$\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} = 0. \tag{3}$$

Every hyperplane divides the space in which it lives into two parts, depending on whether $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} > 0$ or $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} \leq 0$.

In some cases, when we have data labelled with two classes, we can find a **separating hyperplane** such that all the points in one class lie on one side of it, and all the points in the other class lie on the other side.

If we form the x_i into a vector $\boldsymbol{x}$, we can also write

$$\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} = 0. \tag{3}$$

Every hyperplane divides the space in which it lives into two parts, depending on whether $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} > 0$ or $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} \leq 0$.

In some cases, when we have data labelled with two classes, we can find a **separating hyperplane** such that all the points in one class lie on one side of it, and all the points in the other class lie on the other side.

The idea of a **support vector machine** is to find a separating hyperplane, if one exists.

If we form the x_i into a vector $\boldsymbol{x}$, we can also write

$$\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} = 0. \tag{3}$$

Every hyperplane divides the space in which it lives into two parts, depending on whether $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} > 0$ or $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} \leq 0$.

In some cases, when we have data labelled with two classes, we can find a **separating hyperplane** such that all the points in one class lie on one side of it, and all the points in the other class lie on the other side.

The idea of a **support vector machine** is to find a separating hyperplane, if one exists.

If one separating hyperplane exists, then many of them exist, and we want to choose the best one. See Figure 23.1.

If we form the x_i into a vector $\boldsymbol{x}$, we can also write

$$\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} = 0. \tag{3}$$

Every hyperplane divides the space in which it lives into two parts, depending on whether $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} > 0$ or $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} \leq 0$.

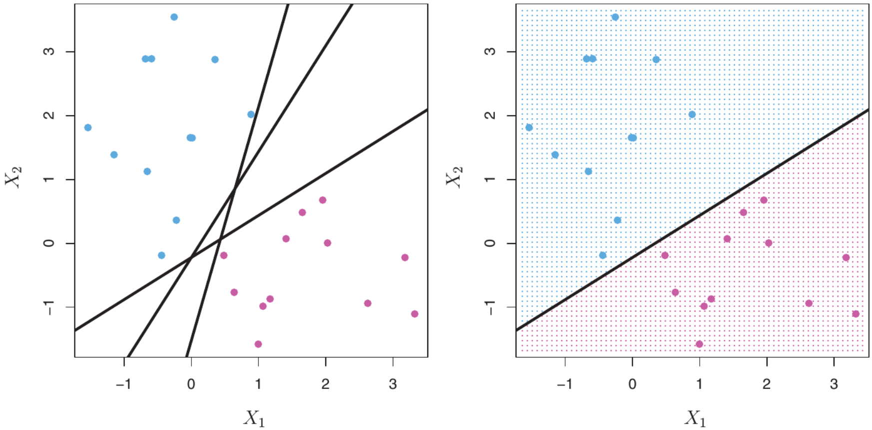
In some cases, when we have data labelled with two classes, we can find a **separating hyperplane** such that all the points in one class lie on one side of it, and all the points in the other class lie on the other side.

The idea of a **support vector machine** is to find a separating hyperplane, if one exists.

If one separating hyperplane exists, then many of them exist, and we want to choose the best one. See Figure 23.1.

More commonly, no separating hyperplane exists. If so, we attempt to find a hyperplane that comes as close as possible.

Figure 23.1 — Separating Hyperplanes



Three separating hyperplanes (left panel) and the decision rule associated with a separating hyperplane (right panel).

Let the training observations be denoted y_i and $\boldsymbol{x}_i$, where y_i contains the class labels, which are -1 and 1.

Let the training observations be denoted y_i and $\boldsymbol{x}_i$, where y_i contains the class labels, which are -1 and 1. Computing science origin!

Let the training observations be denoted y_i and $\boldsymbol{x}_i$, where y_i contains the class labels, which are -1 and 1. Computing science origin!

If a separating hyperplane exists, it must have the property that

$$\begin{aligned} \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} &> 0 \quad \text{if } y_i = 1 \\ \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} &< 0 \quad \text{if } y_i = -1 \end{aligned} \tag{4}$$

for all observations.

Let the training observations be denoted y_i and $\boldsymbol{x}_i$, where y_i contains the class labels, which are -1 and 1. Computing science origin!

If a separating hyperplane exists, it must have the property that

$$\begin{aligned} \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} > 0 \quad \text{if } y_i = 1 \\ \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} < 0 \quad \text{if } y_i = -1 \end{aligned} \tag{4}$$

for all observations.

More compactly, we can write

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) > 0 \quad \text{for all } i = 1, \ldots, n. \tag{5}$$

Let the training observations be denoted y_i and $\boldsymbol{x}_i$, where y_i contains the class labels, which are -1 and 1. Computing science origin!

If a separating hyperplane exists, it must have the property that

$$\begin{aligned} \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} > 0 \quad \text{if } y_i = 1 \\ \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} < 0 \quad \text{if } y_i = -1 \end{aligned} \tag{4}$$

for all observations.

More compactly, we can write

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) > 0 \quad \text{for all } i = 1, \ldots, n. \tag{5}$$

The values of β_0 and $\boldsymbol{\beta}$ are not unique. If (5) is true for any $(\beta_0, \boldsymbol{\beta})$ pair, then it is also true for $(\lambda\beta_0, \lambda\boldsymbol{\beta})$ for any positive λ.

Let the training observations be denoted y_i and $\boldsymbol{x}_i$, where y_i contains the class labels, which are -1 and 1. Computing science origin!

If a separating hyperplane exists, it must have the property that

$$\begin{aligned} \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} > 0 \quad \text{if } y_i = 1 \\ \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} < 0 \quad \text{if } y_i = -1 \end{aligned} \tag{4}$$

for all observations.

More compactly, we can write

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) > 0 \quad \text{for all } i = 1, \ldots, n. \tag{5}$$

The values of β_0 and $\boldsymbol{\beta}$ are not unique. If (5) is true for any $(\beta_0, \boldsymbol{\beta})$ pair, then it is also true for $(\lambda\beta_0, \lambda\boldsymbol{\beta})$ for any positive λ.

When there exists a separating hyperplane, it provides a **perfect classifier**. We simply classify based on the sign of $\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}$.

Maximal Margin Classifiers

Maximal Margin Classifiers

With other methods, such as logit and probit, having a perfect classifier is bad.

Maximal Margin Classifiers

With other methods, such as logit and probit, having a perfect classifier is bad.

- As $\lambda \to \infty$, the value of the loglikelihood function for logit and probit models approaches its maximum value of 0.

Maximal Margin Classifiers

With other methods, such as logit and probit, having a perfect classifier is bad.

- As $\lambda \to \infty$, the value of the loglikelihood function for logit and probit models approaches its maximum value of 0.
- Thus the algorithm tries to make $\lambda \to \infty$.

Maximal Margin Classifiers

With other methods, such as logit and probit, having a perfect classifier is bad.

- As $\lambda \to \infty$, the value of the loglikelihood function for logit and probit models approaches its maximum value of 0.
- Thus the algorithm tries to make $\lambda \to \infty$.
- For support vector machines, this is the ideal situation, albeit one that is rarely achieved with actual data.

Maximal Margin Classifiers

With other methods, such as logit and probit, having a perfect classifier is bad.

- As $\lambda \to \infty$, the value of the loglikelihood function for logit and probit models approaches its maximum value of 0.
- Thus the algorithm tries to make $\lambda \to \infty$.
- For support vector machines, this is the ideal situation, albeit one that is rarely achieved with actual data.

If one separating hyperplane exists, then an infinite number of them exists. In that case, we need to choose among them.

Maximal Margin Classifiers

With other methods, such as logit and probit, having a perfect classifier is bad.

- As $\lambda \to \infty$, the value of the loglikelihood function for logit and probit models approaches its maximum value of 0.
- Thus the algorithm tries to make $\lambda \to \infty$.
- For support vector machines, this is the ideal situation, albeit one that is rarely achieved with actual data.

If one separating hyperplane exists, then an infinite number of them exists. In that case, we need to choose among them.

The **maximal margin hyperplane**, or **optimal separating hyperplane**, is the one that is farthest from the training observations.

Maximal Margin Classifiers

With other methods, such as logit and probit, having a perfect classifier is bad.

- As $\lambda \to \infty$, the value of the loglikelihood function for logit and probit models approaches its maximum value of 0.
- Thus the algorithm tries to make $\lambda \to \infty$.
- For support vector machines, this is the ideal situation, albeit one that is rarely achieved with actual data.

If one separating hyperplane exists, then an infinite number of them exists. In that case, we need to choose among them.

The **maximal margin hyperplane**, or **optimal separating hyperplane**, is the one that is farthest from the training observations.

Intuitively, this seems like the best choice.

The **margin** is simply the smallest perpendicular distance between any of the training observations $\boldsymbol{x}_i$ and the hyperplane.

The **margin** is simply the smallest perpendicular distance between any of the training observations $\boldsymbol{x}_i$ and the hyperplane.

The **maximal margin classifier** classifies each observation based on which side of the maximal margin hyperplane it is.

The **margin** is simply the smallest perpendicular distance between any of the training observations $\boldsymbol{x}_i$ and the hyperplane.

The **maximal margin classifier** classifies each observation based on which side of the maximal margin hyperplane it is.

See Figure 23.2, which is drawn for the same dataset as Figure 23.1.

The **margin** is simply the smallest perpendicular distance between any of the training observations $\boldsymbol{x}_i$ and the hyperplane.

The **maximal margin classifier** classifies each observation based on which side of the maximal margin hyperplane it is.

See Figure 23.2, which is drawn for the same dataset as Figure 23.1.

- In the figure, the maximal margin hyperplane depends on just three points, the three **support vectors**.

The **margin** is simply the smallest perpendicular distance between any of the training observations x_i and the hyperplane.

The **maximal margin classifier** classifies each observation based on which side of the maximal margin hyperplane it is.

See Figure 23.2, which is drawn for the same dataset as Figure 23.1.

- In the figure, the maximal margin hyperplane depends on just three points, the three **support vectors**.
- Small changes in the location of other observations do not affect this hyperplane.

The **margin** is simply the smallest perpendicular distance between any of the training observations $\boldsymbol{x}_i$ and the hyperplane.

The **maximal margin classifier** classifies each observation based on which side of the maximal margin hyperplane it is.

See Figure 23.2, which is drawn for the same dataset as Figure 23.1.

- In the figure, the maximal margin hyperplane depends on just three points, the three **support vectors**.
- Small changes in the location of other observations do not affect this hyperplane.
- On the other hand, even very small changes in the support vectors will change the maximal margin hyperplane, usually affecting both its intercept and its slope.

The **margin** is simply the smallest perpendicular distance between any of the training observations $\boldsymbol{x}_i$ and the hyperplane.

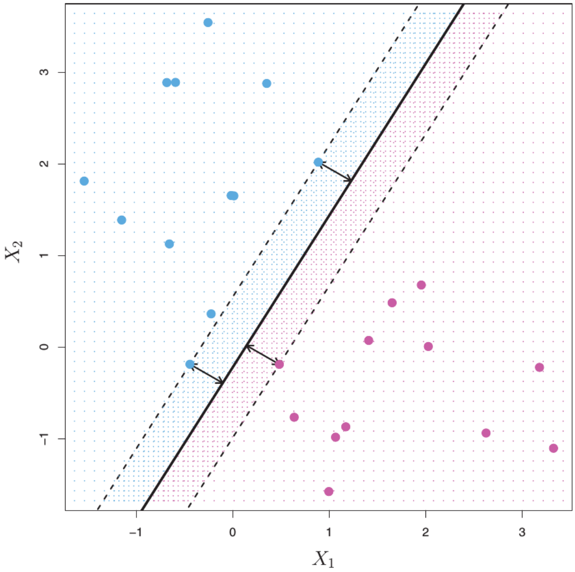
The **maximal margin classifier** classifies each observation based on which side of the maximal margin hyperplane it is.

See Figure 23.2, which is drawn for the same dataset as Figure 23.1.

- In the figure, the maximal margin hyperplane depends on just three points, the three **support vectors**.
- Small changes in the location of other observations do not affect this hyperplane.
- On the other hand, even very small changes in the support vectors will change the maximal margin hyperplane, usually affecting both its intercept and its slope.

The three support vectors (observations) are shown in the figure. Notice what would happen if any of them changed.

Figure 23.2 — Maximal Margin Classifier



The maximal margin hyperplane can be obtained by solving a particular optimization problem.

The maximal margin hyperplane can be obtained by solving a particular optimization problem.

We need to maximize a scalar M with respect to M itself, β_0, and $\boldsymbol{\beta}$ subject to the constraints

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq M, \quad \text{for all } i = 1, \ldots, n. \tag{6}$$

The maximal margin hyperplane can be obtained by solving a particular optimization problem.

We need to maximize a scalar M with respect to M itself, β_0, and $\boldsymbol{\beta}$ subject to the constraints

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq M, \quad \text{for all } i = 1, \ldots, n. \tag{6}$$

and

$$\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta} = 1. \tag{7}$$

The maximal margin hyperplane can be obtained by solving a particular optimization problem.

We need to maximize a scalar M with respect to M itself, β_0, and $\boldsymbol{\beta}$ subject to the constraints

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq M, \quad \text{for all } i = 1, \ldots, n. \tag{6}$$

and

$$\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta} = 1. \tag{7}$$

The first constraint ensures that every point is on the correct side of the maximal margin hyperplane, and indeed that it is distant from it by at least M, the margin.

The maximal margin hyperplane can be obtained by solving a particular optimization problem.

We need to maximize a scalar M with respect to M itself, β_0, and $\boldsymbol{\beta}$ subject to the constraints

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq M, \quad \text{for all } i = 1, \ldots, n. \tag{6}$$

and

$$\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta} = 1. \tag{7}$$

The first constraint ensures that every point is on the correct side of the maximal margin hyperplane, and indeed that it is distant from it by at least M, the margin.

The second constraint is just a normalization. Without such a normalization, there would be infinite number of solutions.

The maximal margin hyperplane can be obtained by solving a particular optimization problem.

We need to maximize a scalar M with respect to M itself, β_0, and $\boldsymbol{\beta}$ subject to the constraints

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq M, \quad \text{for all } i = 1, \ldots, n. \tag{6}$$

and

$$\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta} = 1. \tag{7}$$

The first constraint ensures that every point is on the correct side of the maximal margin hyperplane, and indeed that it is distant from it by at least M, the margin.

The second constraint is just a normalization. Without such a normalization, there would be infinite number of solutions.

For numerical solution, it is easier to rewrite the optimization problem so that M has vanished and there is no constraint on the β coefficients.

The new optimization problem is

$$\min_{\beta_0, \boldsymbol{\beta}} (\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta}) \quad \text{subject to} \tag{8}$$

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq 1, \quad \text{for all } i = 1, \ldots, n. \tag{9}$$

The new optimization problem is

$$\min_{\beta_0,\boldsymbol{\beta}}(\beta_0^2 + \boldsymbol{\beta}^\top\boldsymbol{\beta}) \quad \text{subject to} \tag{8}$$

$$y_i(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}) \geq 1, \quad \text{for all } i = 1, \ldots, n. \tag{9}$$

This is a convex optimization problem that is easily solved. If desired, we can then rescale the coefficients to find M.

The new optimization problem is

$$\min_{\beta_0, \boldsymbol{\beta}} (\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta}) \quad \text{subject to} \tag{8}$$

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq 1, \quad \text{for all } i = 1, \ldots, n. \tag{9}$$

This is a convex optimization problem that is easily solved. If desired, we can then rescale the coefficients to find M.

- The solution to this optimization problem depends on just a few support vectors.

The new optimization problem is

$$\min_{\beta_0, \boldsymbol{\beta}} (\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta}) \quad \text{subject to} \tag{8}$$

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq 1, \quad \text{for all } i = 1, \ldots, n. \tag{9}$$

This is a convex optimization problem that is easily solved. If desired, we can then rescale the coefficients to find M.

- The solution to this optimization problem depends on just a few support vectors.
- Changing the data slightly can greatly change the location of the maximal margin hyperplane, or cause it to vanish.

The new optimization problem is

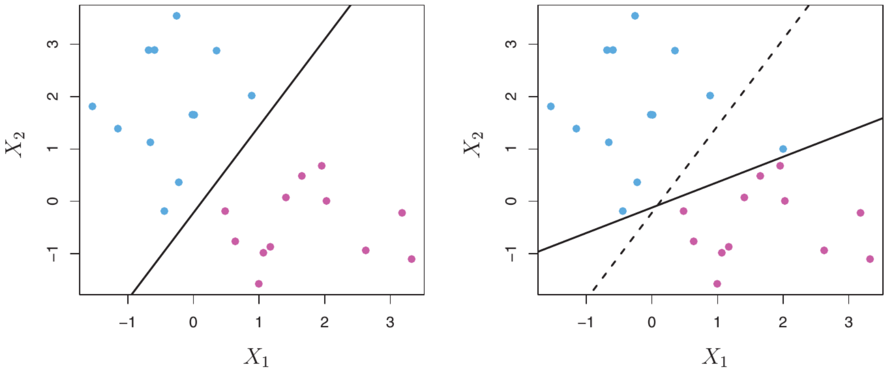
$$\min_{\beta_0, \boldsymbol{\beta}} (\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta}) \quad \text{subject to} \tag{8}$$

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq 1, \quad \text{for all } i = 1, \ldots, n. \tag{9}$$

This is a convex optimization problem that is easily solved. If desired, we can then rescale the coefficients to find M.

- The solution to this optimization problem depends on just a few support vectors.
- Changing the data slightly can greatly change the location of the maximal margin hyperplane, or cause it to vanish.
- In Figure 23.3, adding one observation has changed two of the three support vectors and greatly changed the slope of the maximal margin hyperplane.

Figure 23.3 — Sensitivity of the Maximal Margin Classifier



In the right panel, one observation is added to the ones in the left panel. The maximal margin hyperplane changes greatly.

Support Vector Classifiers

Support Vector Classifiers

In practice, a separating hyperplane rarely exists. For any possible hyperplane, there will be some observations on the wrong side.

Support Vector Classifiers

In practice, a separating hyperplane rarely exists. For any possible hyperplane, there will be some observations on the wrong side.

The **support vector classifier**, or **soft margin classifier**, chooses a hyperplane where some observations are on the wrong side.

Support Vector Classifiers

In practice, a separating hyperplane rarely exists. For any possible hyperplane, there will be some observations on the wrong side.

The **support vector classifier**, or **soft margin classifier**, chooses a hyperplane where some observations are on the wrong side.

In some cases, there may exist a separating hyperplane, but it may still be better to put some observations on the wrong side of the margin.

Support Vector Classifiers

In practice, a separating hyperplane rarely exists. For any possible hyperplane, there will be some observations on the wrong side.

The **support vector classifier**, or **soft margin classifier**, chooses a hyperplane where some observations are on the wrong side.

In some cases, there may exist a separating hyperplane, but it may still be better to put some observations on the wrong side of the margin.

Now we have to maximize M subject to the constraints

$$\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta} = 1, \tag{10}$$

Support Vector Classifiers

In practice, a separating hyperplane rarely exists. For any possible hyperplane, there will be some observations on the wrong side.

The **support vector classifier**, or **soft margin classifier**, chooses a hyperplane where some observations are on the wrong side.

In some cases, there may exist a separating hyperplane, but it may still be better to put some observations on the wrong side of the margin.

Now we have to maximize M subject to the constraints

$$\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta} = 1, \tag{10}$$

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq M(1 - \epsilon_i), \quad \text{for all } i = 1, \ldots, n, \tag{11}$$

Support Vector Classifiers

In practice, a separating hyperplane rarely exists. For any possible hyperplane, there will be some observations on the wrong side.

The **support vector classifier**, or **soft margin classifier**, chooses a hyperplane where some observations are on the wrong side.

In some cases, there may exist a separating hyperplane, but it may still be better to put some observations on the wrong side of the margin.

Now we have to maximize M subject to the constraints

$$\beta_0^2 + \boldsymbol{\beta}^\top \boldsymbol{\beta} = 1, \tag{10}$$

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq M(1 - \epsilon_i), \quad \text{for all } i = 1, \ldots, n, \tag{11}$$

where $\epsilon_i \geq 0$ and $\sum_{i=1}^n \epsilon_i \leq C$, with $C > 0$ a tuning parameter.

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

- If $\epsilon_i = 0$, then observation i lies on the correct side of the margin.

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

- If $\epsilon_i = 0$, then observation i lies on the correct side of the margin.
- If $\epsilon_i > 0$, then observation i lies on the wrong side of the margin.

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

- If $\epsilon_i = 0$, then observation i lies on the correct side of the margin.
- If $\epsilon_i > 0$, then observation i lies on the wrong side of the margin.
- If $\epsilon_i > 1$, then observation i lies on the wrong side of the maximal margin hyperplane.

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

- If $\epsilon_i = 0$, then observation i lies on the correct side of the margin.
- If $\epsilon_i > 0$, then observation i lies on the wrong side of the margin.
- If $\epsilon_i > 1$, then observation i lies on the wrong side of the maximal margin hyperplane.

Not surprisingly, the value of C turns out to be very important.

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

- If $\epsilon_i = 0$, then observation i lies on the correct side of the margin.
- If $\epsilon_i > 0$, then observation i lies on the wrong side of the margin.
- If $\epsilon_i > 1$, then observation i lies on the wrong side of the maximal margin hyperplane.

Not surprisingly, the value of C turns out to be very important.

- Because $\sum_{i=1}^{n} \epsilon_i \leq C$, the value of C limits the extent to which the ϵ_i can collectively exceed zero.

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

- If $\epsilon_i = 0$, then observation i lies on the correct side of the margin.
- If $\epsilon_i > 0$, then observation i lies on the wrong side of the margin.
- If $\epsilon_i > 1$, then observation i lies on the wrong side of the maximal margin hyperplane.

Not surprisingly, the value of C turns out to be very important.

- Because $\sum_{i=1}^{n} \epsilon_i \leq C$, the value of C limits the extent to which the ϵ_i can collectively exceed zero.
- When $C = 0$, we are back to the constraints (6) and (7).

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

- If $\epsilon_i = 0$, then observation i lies on the correct side of the margin.
- If $\epsilon_i > 0$, then observation i lies on the wrong side of the margin.
- If $\epsilon_i > 1$, then observation i lies on the wrong side of the maximal margin hyperplane.

Not surprisingly, the value of C turns out to be very important.

- Because $\sum_{i=1}^{n} \epsilon_i \leq C$, the value of C limits the extent to which the ϵ_i can collectively exceed zero.
- When $C = 0$, we are back to the constraints (6) and (7).
- For $C > 0$, no more than C observations can be on the wrong side of the hyperplane, because $\epsilon_i > 1$ for every such observation.

We now have to choose the ϵ_i as well as M, β_0, and $\boldsymbol{\beta}$. The ϵ_i are called **slack variables**.

- If $\epsilon_i = 0$, then observation i lies on the correct side of the margin.
- If $\epsilon_i > 0$, then observation i lies on the wrong side of the margin.
- If $\epsilon_i > 1$, then observation i lies on the wrong side of the maximal margin hyperplane.

Not surprisingly, the value of C turns out to be very important.

- Because $\sum_{i=1}^{n} \epsilon_i \leq C$, the value of C limits the extent to which the ϵ_i can collectively exceed zero.
- When $C = 0$, we are back to the constraints (6) and (7).
- For $C > 0$, no more than C observations can be on the wrong side of the hyperplane, because $\epsilon_i > 1$ for every such observation.
- Since every violation of the margin increases the sum of the ϵ_i, we can afford more violations when C is large than when it is small. Thus M will almost surely increase with C.

Figure 23.4 — Effects of the Tuning Parameter C

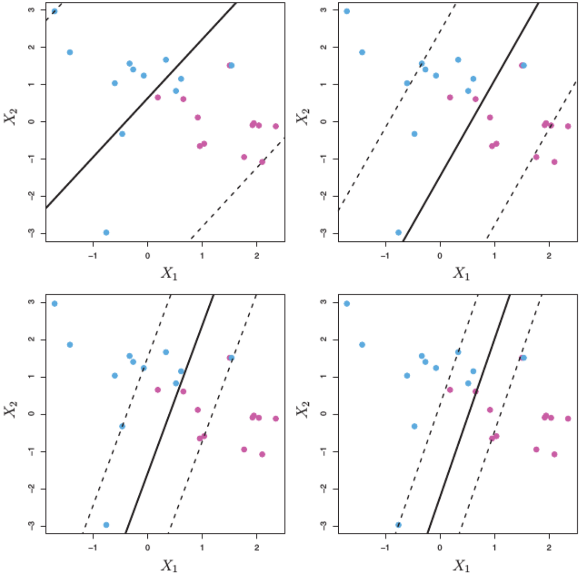


Figure 23.4 shows the effect of changing C.

Figure 23.4 shows the effect of changing C.

As C diminishes from upper left to lower right, the margin shrinks.

Figure 23.4 shows the effect of changing C.

As C diminishes from upper left to lower right, the margin shrinks.

- When C is large, the margin is wide, and many observations are allowed to violate it. There will tend to be low variance, because there are many support vectors, but high bias.

Figure 23.4 shows the effect of changing C.

As C diminishes from upper left to lower right, the margin shrinks.

- When C is large, the margin is wide, and many observations are allowed to violate it. There will tend to be low variance, because there are many support vectors, but high bias.
- Notice that the slope of the (non-separating) hyperplane becomes considerably steeper as C shrinks.

Figure 23.4 shows the effect of changing C.

As C diminishes from upper left to lower right, the margin shrinks.

- When C is large, the margin is wide, and many observations are allowed to violate it. There will tend to be low variance, because there are many support vectors, but high bias.
- Notice that the slope of the (non-separating) hyperplane becomes considerably steeper as C shrinks.
- When C is small, the margin is narrow, and few observations are allowed to violate it. There will tend to be low bias, because there are few support vectors, but high variance.

Figure 23.4 shows the effect of changing C.

As C diminishes from upper left to lower right, the margin shrinks.

- When C is large, the margin is wide, and many observations are allowed to violate it. There will tend to be low variance, because there are many support vectors, but high bias.
- Notice that the slope of the (non-separating) hyperplane becomes considerably steeper as C shrinks.
- When C is small, the margin is narrow, and few observations are allowed to violate it. There will tend to be low bias, because there are few support vectors, but high variance.

The SV classifier is totally insensitive to observations on the correct side of the margin, and therefore (for a wide margin) on observations that are on the correct side of the hyperplane by far.

In contrast, estimates from logistic regression are never totally insensitive to any observation, but they are not very sensitive to observations that are far from the hyperplane on the correct side.

In contrast, estimates from logistic regression are never totally insensitive to any observation, but they are not very sensitive to observations that are far from the hyperplane on the correct side.

The support vector classifier has some merit when most observations lie on one side or the other of a separating hyperplane (in other words, when the model fits really well).

In contrast, estimates from logistic regression are never totally insensitive to any observation, but they are not very sensitive to observations that are far from the hyperplane on the correct side.

The support vector classifier has some merit when most observations lie on one side or the other of a separating hyperplane (in other words, when the model fits really well).

- The logit model may classify well in such cases, but the parameter estimates can be severely biased.

In contrast, estimates from logistic regression are never totally insensitive to any observation, but they are not very sensitive to observations that are far from the hyperplane on the correct side.

The support vector classifier has some merit when most observations lie on one side or the other of a separating hyperplane (in other words, when the model fits really well).

- The logit model may classify well in such cases, but the parameter estimates can be severely biased.
- In such cases, logit performance on test data may be much worse than on training data.

In contrast, estimates from logistic regression are never totally insensitive to any observation, but they are not very sensitive to observations that are far from the hyperplane on the correct side.

The support vector classifier has some merit when most observations lie on one side or the other of a separating hyperplane (in other words, when the model fits really well).

- The logit model may classify well in such cases, but the parameter estimates can be severely biased.
- In such cases, logit performance on test data may be much worse than on training data.
- However, in most cases, support vector classifiers do not seem to have a big advantage over logit.

In contrast, estimates from logistic regression are never totally insensitive to any observation, but they are not very sensitive to observations that are far from the hyperplane on the correct side.

The support vector classifier has some merit when most observations lie on one side or the other of a separating hyperplane (in other words, when the model fits really well).

- The logit model may classify well in such cases, but the parameter estimates can be severely biased.
- In such cases, logit performance on test data may be much worse than on training data.
- However, in most cases, support vector classifiers do not seem to have a big advantage over logit.

Consider Figure 23.5. Here the boundaries between the two classes are very nonlinear.

In contrast, estimates from logistic regression are never totally insensitive to any observation, but they are not very sensitive to observations that are far from the hyperplane on the correct side.

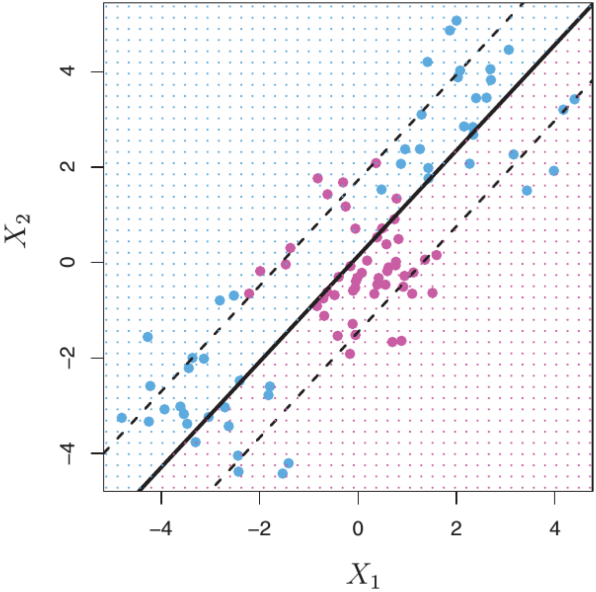
The support vector classifier has some merit when most observations lie on one side or the other of a separating hyperplane (in other words, when the model fits really well).

- The logit model may classify well in such cases, but the parameter estimates can be severely biased.
- In such cases, logit performance on test data may be much worse than on training data.
- However, in most cases, support vector classifiers do not seem to have a big advantage over logit.

Consider Figure 23.5. Here the boundaries between the two classes are very nonlinear.

The linear boundary found by the SV classifier works poorly.

Figure 23.5 — Support Vector Classifier Works Badly



Support Vector Machines

Support Vector Machines

So far, we have only considered decision boundaries that are hyperplanes. But if the boundaries are actually nonlinear, hyperplanes won't work well.

Support Vector Machines

So far, we have only considered decision boundaries that are hyperplanes. But if the boundaries are actually nonlinear, hyperplanes won't work well.

The **support vector machine**, or **SVM**, extends the support vector classifier by enlarging the feature space using **kernels**.

Support Vector Machines

So far, we have only considered decision boundaries that are hyperplanes. But if the boundaries are actually nonlinear, hyperplanes won't work well.

The **support vector machine**, or **SVM**, extends the support vector classifier by enlarging the feature space using **kernels**.

The linear SV classifier for any point $\boldsymbol{x}$ can be represented as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{12}$$

Support Vector Machines

So far, we have only considered decision boundaries that are hyperplanes. But if the boundaries are actually nonlinear, hyperplanes won't work well.

The **support vector machine**, or **SVM**, extends the support vector classifier by enlarging the feature space using **kernels**.

The linear SV classifier for any point $\boldsymbol{x}$ can be represented as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{12}$$

where there is one parameter α_i for each training observation.

Support Vector Machines

So far, we have only considered decision boundaries that are hyperplanes. But if the boundaries are actually nonlinear, hyperplanes won't work well.

The **support vector machine**, or **SVM**, extends the support vector classifier by enlarging the feature space using **kernels**.

The linear SV classifier for any point $\boldsymbol{x}$ can be represented as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{12}$$

where there is one parameter α_i for each training observation.

To estimate the parameters β_0 and α_i, we need every inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$. There are $n(n-1)/2$ of these.

Support Vector Machines

So far, we have only considered decision boundaries that are hyperplanes. But if the boundaries are actually nonlinear, hyperplanes won't work well.

The **support vector machine**, or **SVM**, extends the support vector classifier by enlarging the feature space using **kernels**.

The linear SV classifier for any point $\boldsymbol{x}$ can be represented as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{12}$$

where there is one parameter α_i for each training observation.

To estimate the parameters β_0 and α_i, we need every inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$. There are $n(n-1)/2$ of these.

However, it turns out that $\alpha_i = 0$ if $\boldsymbol{x}_i$ is not a support vector.

Thus we can rewrite (12) as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \, \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{13}$$

Thus we can rewrite (12) as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{13}$$

where $\mathcal{S}$ is the set of support vectors. This makes it very inexpensive to classify a new observation.

Thus we can rewrite (12) as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \, \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{13}$$

where $\mathcal{S}$ is the set of support vectors. This makes it very inexpensive to classify a new observation.

Whenever the inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$ appears in calculations for the support vector classifier, we can replace it by the kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}), \tag{14}$$

Thus we can rewrite (12) as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{13}$$

where $\mathcal{S}$ is the set of support vectors. This makes it very inexpensive to classify a new observation.

Whenever the inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$ appears in calculations for the support vector classifier, we can replace it by the kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}), \tag{14}$$

where $K(\cdot)$ can be chosen in various ways. Now (12) becomes

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}). \tag{15}$$

Thus we can rewrite (12) as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i, \tag{13}$$

where $\mathcal{S}$ is the set of support vectors. This makes it very inexpensive to classify a new observation.

Whenever the inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$ appears in calculations for the support vector classifier, we can replace it by the kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}), \tag{14}$$

where $K(\cdot)$ can be chosen in various ways. Now (12) becomes

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}). \tag{15}$$

We will of course get different results by using different kernels.

The **linear kernel** is just $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$, which gives us the support vector classifier.

The **linear kernel** is just $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$, which gives us the support vector classifier.

The **polynomial kernel** of degree d is

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = (1 + \boldsymbol{x}_i^\top \boldsymbol{x}_{i'})^d. \tag{16}$$

The **linear kernel** is just $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$, which gives us the support vector classifier.

The **polynomial kernel** of degree d is

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = (1 + \boldsymbol{x}_i^\top \boldsymbol{x}_{i'})^d. \tag{16}$$

We are now fitting an SV classifier in a higher-dimensional space involving polynomials of degree d, not in the original feature space.

The **linear kernel** is just $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$, which gives us the support vector classifier.

The **polynomial kernel** of degree d is

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = (1 + \boldsymbol{x}_i^\top \boldsymbol{x}_{i'})^d. \tag{16}$$

We are now fitting an SV classifier in a higher-dimensional space involving polynomials of degree d, not in the original feature space.

Of particular interest is the **radial kernel**, which is

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \exp\big(-\gamma(\boldsymbol{x}_i - \boldsymbol{x}_{i'})^\top(\boldsymbol{x}_i - \boldsymbol{x}_{i'})\big). \tag{17}$$

The **linear kernel** is just $\boldsymbol{x}_i^\top \boldsymbol{x}_{i'}$, which gives us the support vector classifier.

The **polynomial kernel** of degree d is

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = (1 + \boldsymbol{x}_i^\top \boldsymbol{x}_{i'})^d. \tag{16}$$

We are now fitting an SV classifier in a higher-dimensional space involving polynomials of degree d, not in the original feature space.

Of particular interest is the **radial kernel**, which is

$$K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \exp\big(-\gamma(\boldsymbol{x}_i - \boldsymbol{x}_{i'})^\top(\boldsymbol{x}_i - \boldsymbol{x}_{i'})\big). \tag{17}$$

The value of γ, which is a positive constant, may be chosen in advance or by cross-validation.

The radial kernel has very local behavior. When a test observation $\boldsymbol{x}_0$ is far from the training observation $\boldsymbol{x}_i$, then

$$K(\boldsymbol{x}_0, \boldsymbol{x}_i) = \exp\big(-\gamma(\boldsymbol{x}_0 - \boldsymbol{x}_i)^\top(\boldsymbol{x}_0 - \boldsymbol{x}_i)\big) \tag{18}$$

will be very small unless γ is tiny.

The radial kernel has very local behavior. When a test observation $\boldsymbol{x}_0$ is far from the training observation $\boldsymbol{x}_i$, then

$$K(\boldsymbol{x}_0, \boldsymbol{x}_i) = \exp\big(-\gamma(\boldsymbol{x}_0 - \boldsymbol{x}_i)^\top(\boldsymbol{x}_0 - \boldsymbol{x}_i)\big) \tag{18}$$

will be very small unless γ is tiny.

This means that $\boldsymbol{x}_i$ will play virtually no role in $f(\boldsymbol{x}_0)$. Training observations that are far from $\boldsymbol{x}_0$ will have very little impact on the predicted class label for $\boldsymbol{x}_0$.

The radial kernel has very local behavior. When a test observation $\boldsymbol{x}_0$ is far from the training observation $\boldsymbol{x}_i$, then

$$K(\boldsymbol{x}_0, \boldsymbol{x}_i) = \exp\big(-\gamma(\boldsymbol{x}_0 - \boldsymbol{x}_i)^\top(\boldsymbol{x}_0 - \boldsymbol{x}_i)\big) \tag{18}$$

will be very small unless γ is tiny.

This means that $\boldsymbol{x}_i$ will play virtually no role in $f(\boldsymbol{x}_0)$. Training observations that are far from $\boldsymbol{x}_0$ will have very little impact on the predicted class label for $\boldsymbol{x}_0$.

In this sense, an SVM with a radial kernel is like kernel regression and smoothing splines, where observations far from $\boldsymbol{x}_0$ have little or no effect on the fitted value for $\boldsymbol{x}_0$.

The radial kernel has very local behavior. When a test observation $\boldsymbol{x}_0$ is far from the training observation $\boldsymbol{x}_i$, then

$$K(\boldsymbol{x}_0, \boldsymbol{x}_i) = \exp\big(-\gamma(\boldsymbol{x}_0 - \boldsymbol{x}_i)^\top(\boldsymbol{x}_0 - \boldsymbol{x}_i)\big) \tag{18}$$

will be very small unless γ is tiny.

This means that $\boldsymbol{x}_i$ will play virtually no role in $f(\boldsymbol{x}_0)$. Training observations that are far from $\boldsymbol{x}_0$ will have very little impact on the predicted class label for $\boldsymbol{x}_0$.

In this sense, an SVM with a radial kernel is like kernel regression and smoothing splines, where observations far from $\boldsymbol{x}_0$ have little or no effect on the fitted value for $\boldsymbol{x}_0$.

One advantage of using kernels instead of simply adding functions of the original inputs is that the data continue to affect the results only through the $n(n-1)/2)$ distinct values of $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$.

For any vector $\boldsymbol{x}$, we can compute $\hat{f}(\boldsymbol{x})$ and classify an observation based on the value of $\hat{f}(\boldsymbol{x}) - t$ for some cutoff value t.

For any vector $\boldsymbol{x}$, we can compute $\hat{f}(\boldsymbol{x})$ and classify an observation based on the value of $\hat{f}(\boldsymbol{x}) - t$ for some cutoff value t.

SVMs can be generalized to handle more than two classes.

For any vector $\boldsymbol{x}$, we can compute $\hat{f}(\boldsymbol{x})$ and classify an observation based on the value of $\hat{f}(\boldsymbol{x}) - t$ for some cutoff value t.

SVMs can be generalized to handle more than two classes.

These generalizations are not very natural, because we have to do everything for pairs of outcomes.

For any vector $\boldsymbol{x}$, we can compute $\hat{f}(\boldsymbol{x})$ and classify an observation based on the value of $\hat{f}(\boldsymbol{x}) - t$ for some cutoff value t.

SVMs can be generalized to handle more than two classes.

These generalizations are not very natural, because we have to do everything for pairs of outcomes.

They have also been generalized to **support vector regression**, where only residuals larger in absolute value than some positive constant contribute to the loss function.

For any vector $\boldsymbol{x}$, we can compute $\hat{f}(\boldsymbol{x})$ and classify an observation based on the value of $\hat{f}(\boldsymbol{x}) - t$ for some cutoff value t.

SVMs can be generalized to handle more than two classes.

These generalizations are not very natural, because we have to do everything for pairs of outcomes.

They have also been generalized to **support vector regression**, where only residuals larger in absolute value than some positive constant contribute to the loss function.

This seems like the opposite of robust regression, which gives less weight to extreme residuals than least squares does.

For any vector $\boldsymbol{x}$, we can compute $\hat{f}(\boldsymbol{x})$ and classify an observation based on the value of $\hat{f}(\boldsymbol{x}) - t$ for some cutoff value t.

SVMs can be generalized to handle more than two classes.

These generalizations are not very natural, because we have to do everything for pairs of outcomes.

They have also been generalized to **support vector regression**, where only residuals larger in absolute value than some positive constant contribute to the loss function.

This seems like the opposite of robust regression, which gives less weight to extreme residuals than least squares does.

SVM with a well-chosen kernel can work well. See Figure 23.6, which is taken from ESL, and Figure 23.7, which is ISLR/ISLP Figure 9.9.

Figure 18.7 — Support Vector Classifier with Radial Kernel

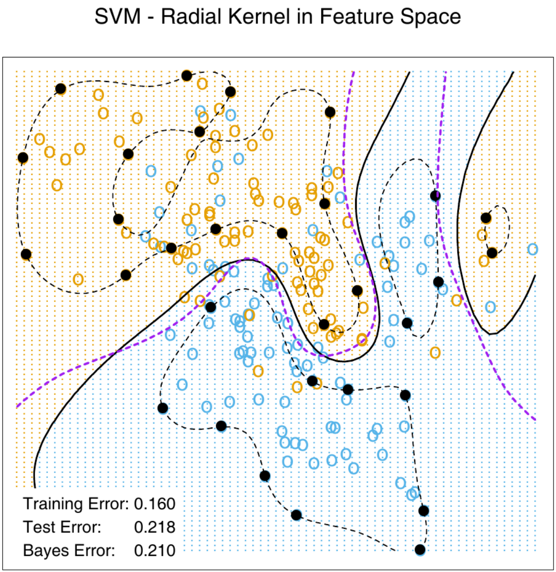
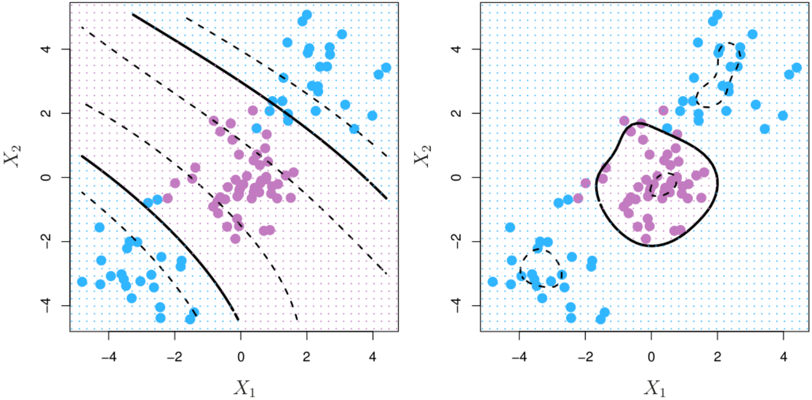


Figure 23.7 — Support Vector Classifiers with Two Kernels



Left panel uses polynomial kernel; right panel uses radial kernel.

Support Vector Classifiers and Machines in R

Support Vector Classifiers and Machines in R

Section 9.6 of ISLR/ISLP shows how to use the `svm` function in the `e1071` library to estimate SVCs and SVMs.

Support Vector Classifiers and Machines in R

Section 9.6 of ISLR/ISLP shows how to use the `svm` function in the `e1071` library to estimate SVCs and SVMs.

Instead of specifying M, the width of the margin, we specify a parameter `cost` which is the cost of violating the margin.

Support Vector Classifiers and Machines in R

Section 9.6 of ISLR/ISLP shows how to use the `svm` function in the `e1071` library to estimate SVCs and SVMs.

Instead of specifying M, the width of the margin, we specify a parameter `cost` which is the cost of violating the margin.

- When `cost` is small, the margins will be wide, and there will be many support vectors.

Support Vector Classifiers and Machines in R

Section 9.6 of ISLR/ISLP shows how to use the `svm` function in the `e1071` library to estimate SVCs and SVMs.

Instead of specifying M, the width of the margin, we specify a parameter `cost` which is the cost of violating the margin.

- When `cost` is small, the margins will be wide, and there will be many support vectors.
- When `cost` is large, the margins will be narrow, and there will be few support vectors.

Support Vector Classifiers and Machines in R

Section 9.6 of ISLR/ISLP shows how to use the `svm` function in the `e1071` library to estimate SVCs and SVMs.

Instead of specifying M, the width of the margin, we specify a parameter `cost` which is the cost of violating the margin.

- When `cost` is small, the margins will be wide, and there will be many support vectors.
- When `cost` is large, the margins will be narrow, and there will be few support vectors.

The `svm` function can handle linear, polynomial, and radial kernels. Using a linear kernel means estimating an SVC.

Support Vector Classifiers and Machines in R

Section 9.6 of ISLR/ISLP shows how to use the `svm` function in the `e1071` library to estimate SVCs and SVMs.

Instead of specifying M, the width of the margin, we specify a parameter `cost` which is the cost of violating the margin.

- When `cost` is small, the margins will be wide, and there will be many support vectors.
- When `cost` is large, the margins will be narrow, and there will be few support vectors.

The `svm` function can handle linear, polynomial, and radial kernels. Using a linear kernel means estimating an SVC.

`svm` can also perform support vector regression, it can deal with more than two classes, and it can output fitted values which can be used to plot ROC curves.

Support Vector Machines versus the Logit Model

Support Vector Machines versus the Logit Model

SVMs have benefited from good marketing, much of it misleading.

Support Vector Machines versus the Logit Model

SVMs have benefited from good marketing, much of it misleading.

They take a very different approach to computation, but SVMs are not as different from other methods as they were once claimed to be.

Support Vector Machines versus the Logit Model

SVMs have benefited from good marketing, much of it misleading.

They take a very different approach to computation, but SVMs are not as different from other methods as they were once claimed to be.

We can rewrite the criterion for estimating a support vector classifier as

$$\min_{\beta_0, \boldsymbol{\beta}} \left(\sum_{i=1}^{n} \max\left(0, y_i f(\boldsymbol{x}_i)\right) + \lambda \boldsymbol{\beta}^\top \boldsymbol{\beta} \right), \tag{19}$$

Support Vector Machines versus the Logit Model

SVMs have benefited from good marketing, much of it misleading.

They take a very different approach to computation, but SVMs are not as different from other methods as they were once claimed to be.

We can rewrite the criterion for estimating a support vector classifier as

$$\min_{\beta_0, \boldsymbol{\beta}} \left(\sum_{i=1}^{n} \max\left(0, y_i f(\boldsymbol{x}_i)\right) + \lambda \boldsymbol{\beta}^\top \boldsymbol{\beta} \right), \tag{19}$$

where $\lambda \geq 0$ is a tuning parameter, and $f(\boldsymbol{x})$ is the support vector classifier.

Support Vector Machines versus the Logit Model

SVMs have benefited from good marketing, much of it misleading.

They take a very different approach to computation, but SVMs are not as different from other methods as they were once claimed to be.

We can rewrite the criterion for estimating a support vector classifier as

$$\min_{\beta_0,\boldsymbol{\beta}}\left(\sum_{i=1}^{n}\max\left(0, y_i f(\boldsymbol{x}_i)\right)+\lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}\right), \tag{19}$$

where $\lambda \geq 0$ is a tuning parameter, and $f(\boldsymbol{x})$ is the support vector classifier.

(19) looks a lot like other penalized criterion functions, but the thing we minimize is a bit different.

Support Vector Machines versus the Logit Model

SVMs have benefited from good marketing, much of it misleading.

They take a very different approach to computation, but SVMs are not as different from other methods as they were once claimed to be.

We can rewrite the criterion for estimating a support vector classifier as

$$\min_{\beta_0,\boldsymbol{\beta}} \left(\sum_{i=1}^{n} \max\big(0, y_i f(\boldsymbol{x}_i)\big) + \lambda \boldsymbol{\beta}^\top \boldsymbol{\beta} \right), \tag{19}$$

where $\lambda \geq 0$ is a tuning parameter, and $f(\boldsymbol{x})$ is the support vector classifier.

(19) looks a lot like other penalized criterion functions, but the thing we minimize is a bit different.

When λ is large, the β_j will tend to be small, many violations of the margin will be tolerated, and we obtain an estimator with low variance but high bias.

When λ is small, the β_j will tend to be large, fewer violations of the margin will be tolerated, and we obtain an estimator with high variance but low bias.

When λ is small, the β_j will tend to be large, fewer violations of the margin will be tolerated, and we obtain an estimator with high variance but low bias.

A small value of λ corresponds to a small value of C.

When λ is small, the β_j will tend to be large, fewer violations of the margin will be tolerated, and we obtain an estimator with high variance but low bias.

A small value of λ corresponds to a small value of C.

- The form of the objective function (19) for SVM is very similar to the one for ridge-regularized logistic regression.

When λ is small, the β_j will tend to be large, fewer violations of the margin will be tolerated, and we obtain an estimator with high variance but low bias.

A small value of λ corresponds to a small value of C.

- The form of the objective function (19) for SVM is very similar to the one for ridge-regularized logistic regression.
- In (19), $\lambda \boldsymbol{\beta}^\top \boldsymbol{\beta}$ looks just like the penalty term for ridge regression.

When λ is small, the β_j will tend to be large, fewer violations of the margin will be tolerated, and we obtain an estimator with high variance but low bias.

A small value of λ corresponds to a small value of C.

- The form of the objective function (19) for SVM is very similar to the one for ridge-regularized logistic regression.
- In (19), $\lambda \boldsymbol{\beta}^\top \boldsymbol{\beta}$ looks just like the penalty term for ridge regression.
- The only real difference between ridge-regularized logistic regression and a support vector classifier is the loss function.

When λ is small, the β_j will tend to be large, fewer violations of the margin will be tolerated, and we obtain an estimator with high variance but low bias.

A small value of λ corresponds to a small value of C.

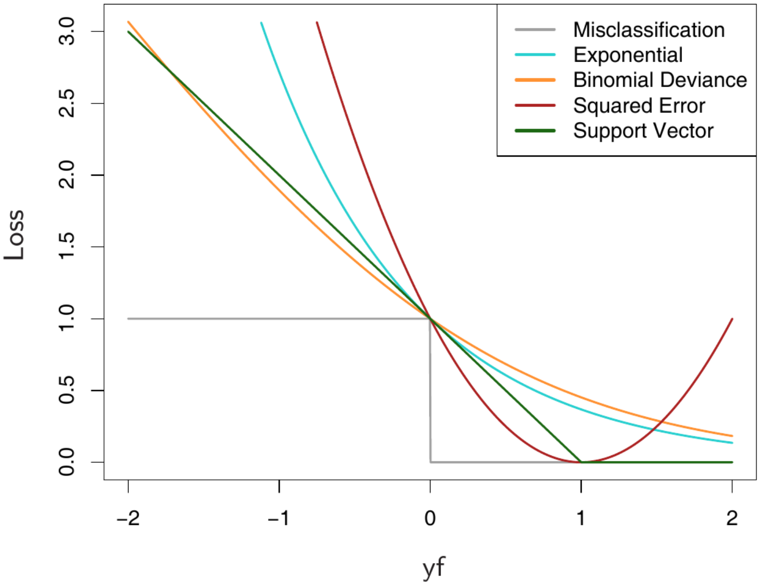
- The form of the objective function (19) for SVM is very similar to the one for ridge-regularized logistic regression.
- In (19), $\lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}$ looks just like the penalty term for ridge regression.
- The only real difference between ridge-regularized logistic regression and a support vector classifier is the loss function.
- Instead of deviance, we minimize $\sum_{i=1}^{n} \max\left(0, y_i f(\boldsymbol{x}_i)\right)$, which is called **hinge loss**. See Figure 23.8, which is from ESL.

When λ is small, the β_j will tend to be large, fewer violations of the margin will be tolerated, and we obtain an estimator with high variance but low bias.

A small value of λ corresponds to a small value of C.

- The form of the objective function (19) for SVM is very similar to the one for ridge-regularized logistic regression.
- In (19), $\lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}$ looks just like the penalty term for ridge regression.
- The only real difference between ridge-regularized logistic regression and a support vector classifier is the loss function.
- Instead of deviance, we minimize $\sum_{i=1}^n \max\big(0, y_i f(\boldsymbol{x}_i)\big)$, which is called **hinge loss**. See Figure 23.8, which is from ESL.
- If an observation is on the correct side of the margin, the loss is 0. If it is on the wrong side, the loss is linear with a slope of 1.

Figure 23.8 — Loss Functions for Binary Classification



The loss functions for logistic regression (binomial deviance) and support vector classifiers (hinge loss) are quite similar.

The loss functions for logistic regression (binomial deviance) and support vector classifiers (hinge loss) are quite similar.

- For SVMs, the loss function is zero for observations on the correct side of the margin.

The loss functions for logistic regression (binomial deviance) and support vector classifiers (hinge loss) are quite similar.

- For SVMs, the loss function is zero for observations on the correct side of the margin.
- For logit, the loss function is close to zero for observations that are classified correctly and are far from the decision boundary.

The loss functions for logistic regression (binomial deviance) and support vector classifiers (hinge loss) are quite similar.

- For SVMs, the loss function is zero for observations on the correct side of the margin.
- For logit, the loss function is close to zero for observations that are classified correctly and are far from the decision boundary.

Due to the similarities between their loss functions, logistic regression and the support vector classifier often give very similar results.

The loss functions for logistic regression (binomial deviance) and support vector classifiers (hinge loss) are quite similar.

- For SVMs, the loss function is zero for observations on the correct side of the margin.
- For logit, the loss function is close to zero for observations that are classified correctly and are far from the decision boundary.

Due to the similarities between their loss functions, logistic regression and the support vector classifier often give very similar results.

When the classes are well separated, SVMs tend to behave better than logit, but, when there is more overlap, logit is often preferred.

The loss functions for logistic regression (binomial deviance) and support vector classifiers (hinge loss) are quite similar.

- For SVMs, the loss function is zero for observations on the correct side of the margin.
- For logit, the loss function is close to zero for observations that are classified correctly and are far from the decision boundary.

Due to the similarities between their loss functions, logistic regression and the support vector classifier often give very similar results.

When the classes are well separated, SVMs tend to behave better than logit, but, when there is more overlap, logit is often preferred.

Other features of Figure 23.8 are also interesting.

The loss functions for logistic regression (binomial deviance) and support vector classifiers (hinge loss) are quite similar.

- For SVMs, the loss function is zero for observations on the correct side of the margin.
- For logit, the loss function is close to zero for observations that are classified correctly and are far from the decision boundary.

Due to the similarities between their loss functions, logistic regression and the support vector classifier often give very similar results.

When the classes are well separated, SVMs tend to behave better than logit, but, when there is more overlap, logit is often preferred.

Other features of Figure 23.8 are also interesting.

- We could perfectly well use nonlinear transformations with (regularized) logistic regression.

The loss functions for logistic regression (binomial deviance) and support vector classifiers (hinge loss) are quite similar.

- For SVMs, the loss function is zero for observations on the correct side of the margin.
- For logit, the loss function is close to zero for observations that are classified correctly and are far from the decision boundary.

Due to the similarities between their loss functions, logistic regression and the support vector classifier often give very similar results.

When the classes are well separated, SVMs tend to behave better than logit, but, when there is more overlap, logit is often preferred.

Other features of Figure 23.8 are also interesting.

- We could perfectly well use nonlinear transformations with (regularized) logistic regression.
- The fact that SVMs routinely use nonlinear transformations (kernels) and logistic regression does not mainly reflects history and software availability.