

DS 740

Data Mining

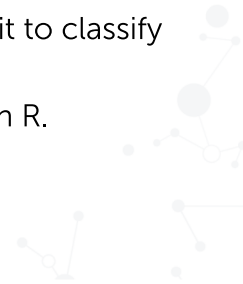
Maximal Margin and Support Vector Classifiers

Important note: Transcripts are **not** substitutes for textbook assignments.

Learning Objectives

By the end of this lesson, you will be able to:

- Explain how to find the maximal margin hyperplane for a data set.
- Use a graph of a support vector classifier and its margins to identify which points are support vectors.
- Explain how the cost affects the margin size, bias, and variance of a support vector classifier.
- Create a support vector classifier in R, plot it, and use it to classify points in the training data
- Choose an optimal cost value using cross-validation in R.



Support Vector Machines

Primary use: Classification

In this presentation:

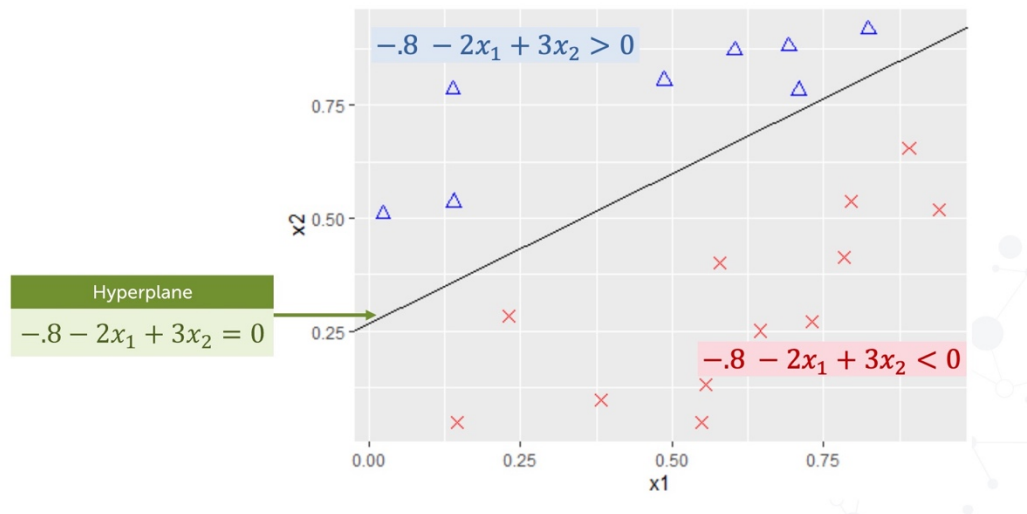
- Maximal margin classifier
- Support vector classifier
- Support vector machine



Support vector machines are a form of supervised learning that are primarily used for classification although they can also be used for regression type problems. To understand how they work, we'll start by discussing the maximal margin classifier and then generalize it to the support vector classifier and then go to the full generalization of the support vector machine.

Separating Lines

Hyperplane: decision boundary



Suppose we want to classify our data into two categories, blue triangles and red x's, and our training data look like this. In that case, it seems intuitively reasonable to classify new datapoints by drawing a line between the two categories in our training data.

In this case, that line has the equation negative 0.8 minus 2 times x_1 plus 3 times x_2 equals 0. If we want to classify new datapoints, we can plug in their x_1 and x_2 values into the left-hand side of this equation and check whether it's above 0 or lower than 0-- meaning that the point is above or below the line. If it's above the line, we can classify it as a blue triangle. If it's below the line, we can classify it as a red x.

If we generalize this to more than two predictor variables, in three dimensions, instead of a separating line, we would have a separating plane. In four or more dimensions, we would have a separating hyperplane. So in general, we sometimes refer to the decision boundary as a hyperplane even if it's only in two or three dimensions.

Notes:

```
n = 20
```

```
set.seed(524)
```

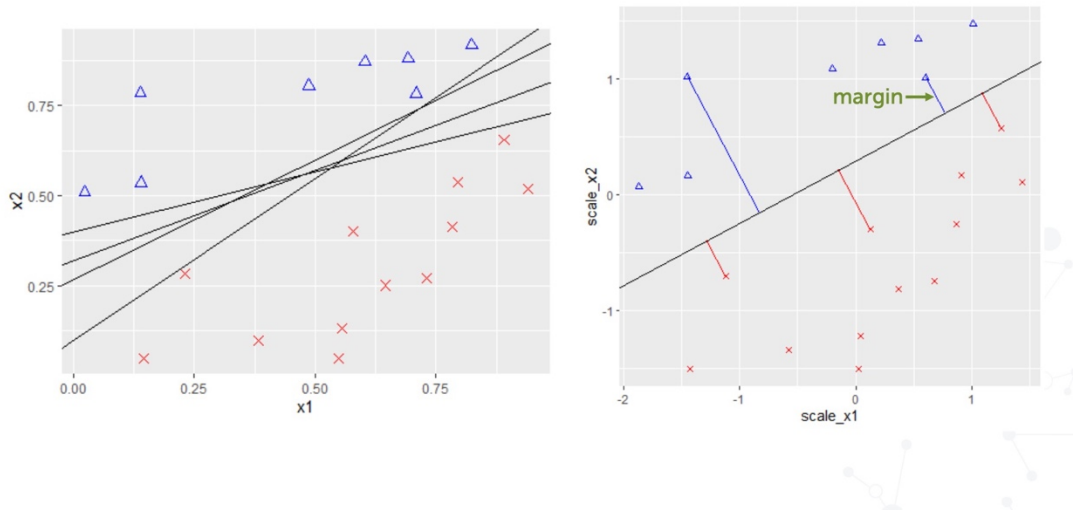
```
x = matrix(runif(n*2),nc=2) y = rep(-1, n)
```

```
y[which(-.8-2*x[,1]+3*x[,2]>0)] = 1
```

```
mycol = rep("red", 20) mycol[which(y==1)] = "blue"
```

```
plot(x, pch = 3-y, col = mycol, las=1, cex.axis=1.2, xlab="x1",  
ylab = "x2")
```

Optimal Separating Hyperplane

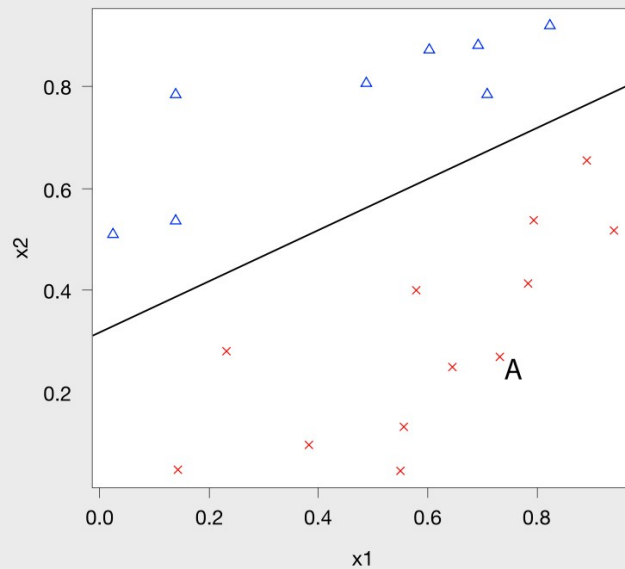


If our datapoints can be perfectly separated into categories by a hyperplane, then there are actually infinitely many choices for which hyperplane to use. So an important question is-- how do we choose the best one, the optimal separating hyperplane? The way that the maximal margin classifier does this is by choosing the line that maximizes the margin or the shortest distance between any datapoint and the line measuring perpendicularly from the point to the line. Hence, the maximal margin classifier.

Question 1

Question for Self Assessment: Multiple Choice

If we added a small amount of noise to point A, how would the maximum margin hyperplane change?



- ☐ Both the slope and y-intercept would change.
- ☐ The slope would change but the y-intercept would not.
- ☐ Neither the slope nor the y-intercept would change.

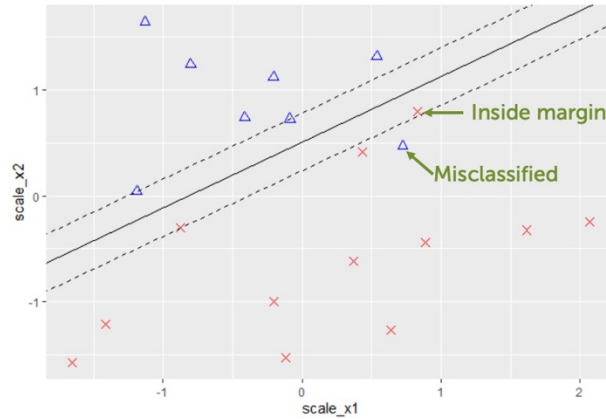
SUBMIT

Answers are at end of the transcript.

Non-Separable Data

Soft margin, but...

Cost penalty for points that are misclassified or inside margin

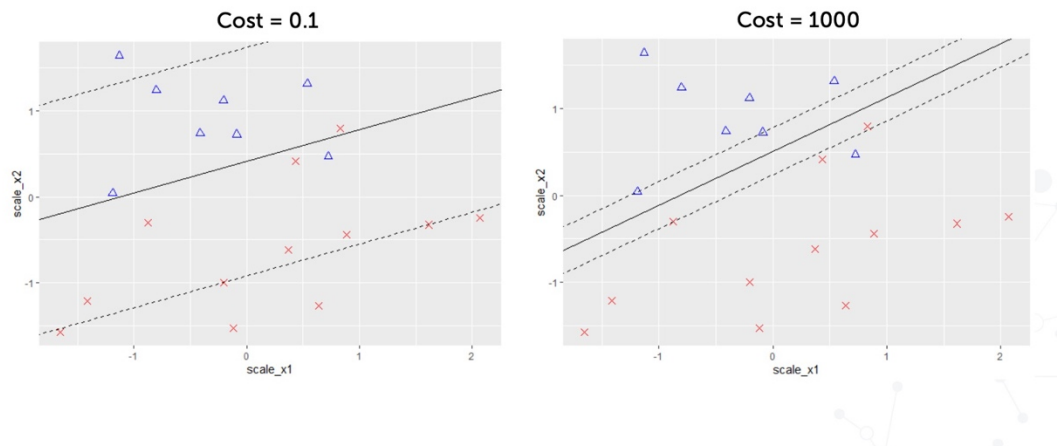


Real data often can't be perfectly separated by a hyperplane. In this case, we need to generalize the idea of a maximal margin classifier to a support vector classifier, as shown by the solid black line here. In a support vector classifier, we allow for a soft margin-- that is, a margin that can be violated by some datapoints. But we have a cost penalty for points that are either misclassified on the wrong side of the solid black line or inside the margin-- that is, closer to the decision boundary than the margin would allow for in the maximal margin classifier.

Effect of Cost

Low cost: wide margins, many points on wrong side of margin.

High cost: narrow margins, few points on wrong side.



When the cost is low, we typically get wide margins with many points on the wrong side of the margin. When the cost is high, we get narrow margins with fewer points inside the margin. The maximal margin classifier with no points inside the margin corresponds to a cost of infinity. You'll notice that, when we change the cost, the slope and y-intercept of the separating line also changes. So it's a good idea to tune the cost using cross-validation to get the optimal model.

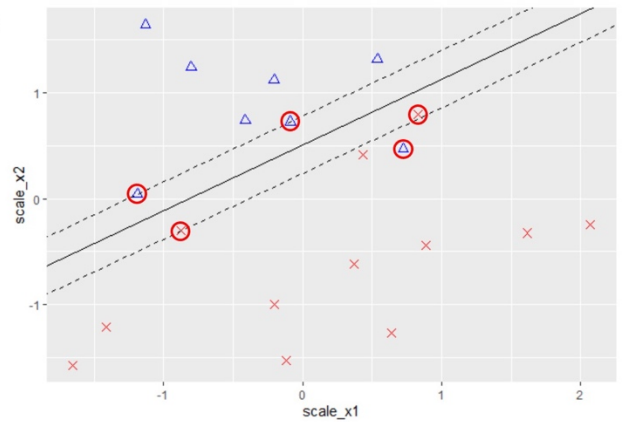
Notes:

Note that p. 346 of our book parameterizes the support vector classifier differently than R does. A low cost (in R) corresponds to a high value for C , the "budget" for error, on p. 346.

Support Vectors

Support vector classifier not affected by points outside the margins, on correct side.

Support vectors: Any data points that are on the margin, inside it, or misclassified.



Like the maximal margin classifier, the support vector classifier isn't affected by points that are far away from the separating line. In fact, the only points that affect the support vector classifier are the datapoints that are on the margin, inside it, or misclassified. We call these points "support vectors." We call them vectors because, if you have multiple predictor variables and you list the coordinates of each datapoint, then it looks like a vector.

Question 2

 Question for Self Assessment: Multiple Choice

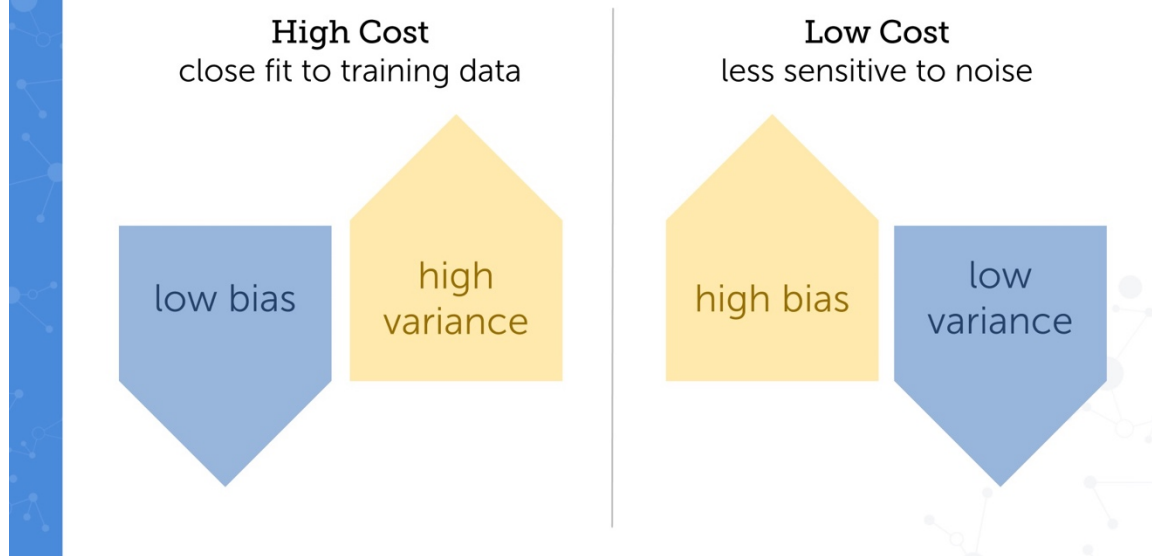
For a given data set, what do we expect to happen to the number of support vectors as the cost increases?

- ☐ It will decrease.
- ☐ It will increase.

SUBMIT

Answers are at end of the transcript.

Bias and Variance Tradeoff

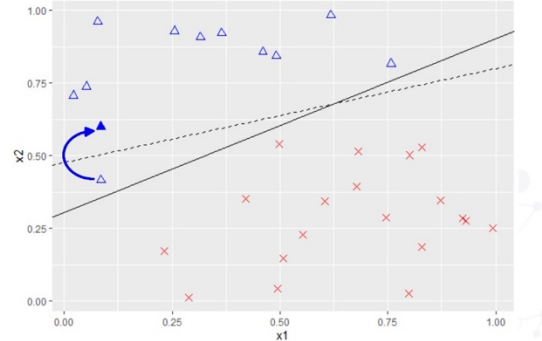
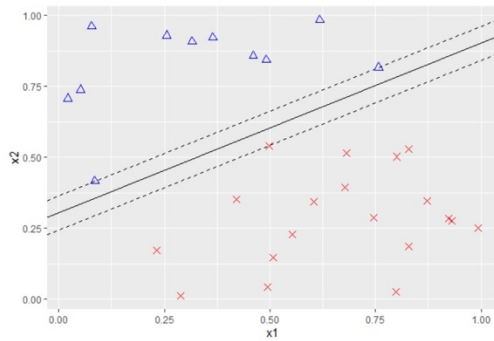


Our choice of cost influences where we fall in the trade-off between bias and variance. With a high cost, we get a close fit to the training data-- meaning very few points, if any, will be misclassified. That means we have low bias because we fit the training data well, but we also end up with high variance-- meaning small changes in the training data set can have a big influence on the equation of the separating hyperplane.

With low cost, we're more likely to get points that are misclassified or on the wrong side of the margin. That means we have high bias, but we're also less sensitive to noise or small changes in the training data set. So we have low variance.

Bias and Variance

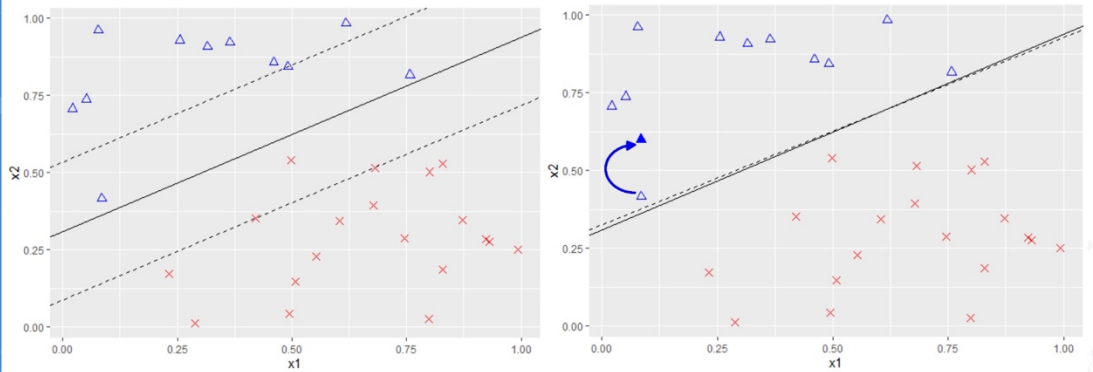
May want a soft margin even if data are separable



This trade-off means we may want a soft margin even if the data are separable. For example, in this dataset the two categories can be perfectly separated by a line. In this case, using a cost of 1,000 turns out to be close enough to infinity that we get the maximal margin classifier. So there are no points that are inside the margins.

But now, look at what happens if we take just one of the support vectors and add a little noise to it so it moves up to the solid blue triangle shown here. In this case, the maximal margin classifier moves from the solid line to the dashed line. Quite a big difference in the slope and y-intercept.

Separable Data with a Soft Margin

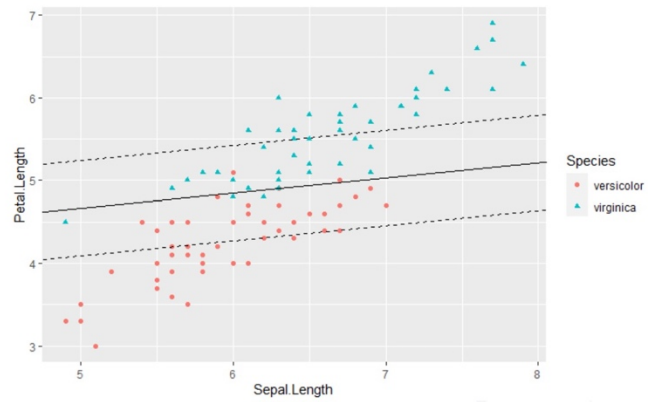


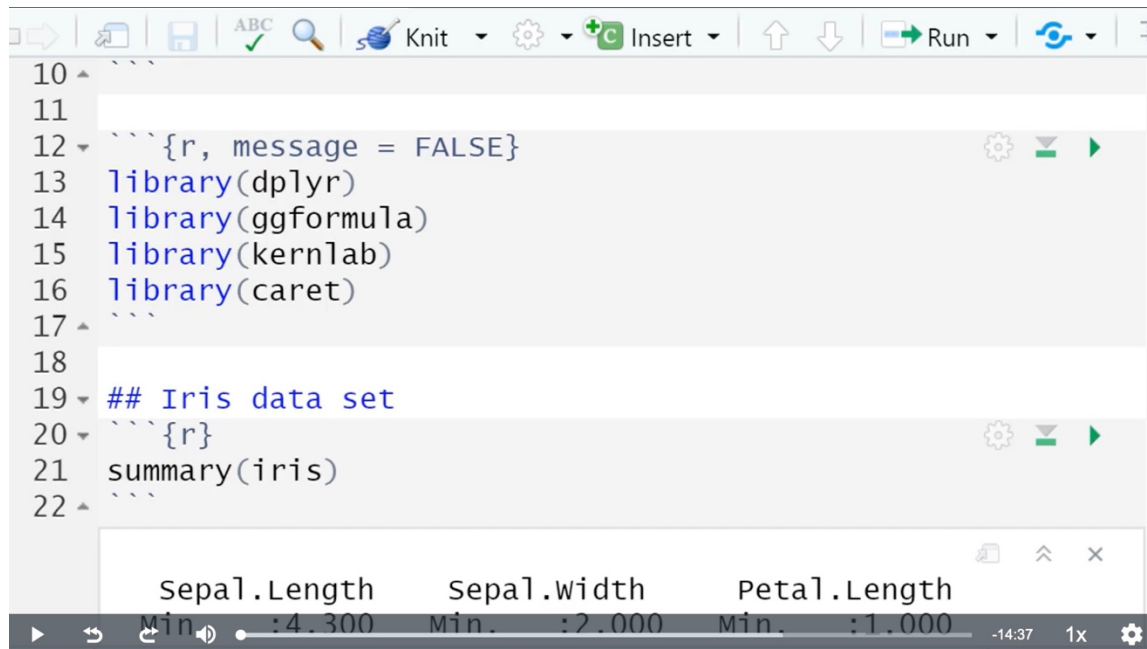
In contrast, here's what the support vector classifier looks like for the same dataset with a cost of 1. Here, the margins are wider, so we do have some points that are inside the margins. But if we move one support vector the same amount as we did before, we get only a tiny change in the slope and y-intercept of the separating line.

Sample Problem

Use petal length and sepal length to classify species of iris.

Dataset: iris





```
10  ```
11
12  ```{r, message = FALSE}
13  library(dplyr)
14  library(ggformula)
15  library(kernlab)
16  library(caret)
17  ```
18
19  ## Iris data set
20  ```{r}
21  summary(iris)
22  ```
```

	Sepal.Length	Sepal.Width	Petal.Length
Min.	4.300	2.000	1.000

Let's use a support vector classifier to model the species of irises. Here, I've loaded the packages `dplyr`, `ggformula`, and `caret` as usual, and I've also loaded the package `kernlab`, which we'll use to fit our support vector classifiers and support vector machines.

The iris data set is built-in in R. It actually contains irises of three different species. And we can use support vector classifiers to do classification of more than two categories at a time. But to keep things simple, let's subset our data to focus on just two species-- `versicolor` and `virginica` irises.

It's a good idea to graph your data to check whether a support vector classifier is likely to be a good model. So here, I'm graphing my data using a different color for each species of iris, and a different plotting character for each species as well. That will help make my graph more readable for any readers who might be color blind, as well as anybody who might be printing my report on a black and white printer.

So here, it looks like the two types of irises are not completely separable. There is some overlap among the points. So a maximal margin classifier won't work. But it does look like the dividing line between the two species of iris is roughly linear, so I think we can use a support vector classifier, or a linear kernel, rather than a support vector machine with a polynomial or radial kernel.

To fit our support vector classifier, we'll use `caret` with method `equals` `svmlinear`, which is available in the `kernlab` package. The rest of this is pretty similar to what you've seen in `caret` before.

A couple of things to point out are that we are preprocessing our data by centering and scaling it, and that we're tuning one parameter, C , which is the cost. So I'm testing a variety of different costs, ranging from 0.001 up to 100.

Here, it looks like all of our costs performed fairly similarly. The optimal model that caret selected was a cost of 0.1, which had an accuracy of 0.93. We can see a bit more detail about that final model by using our caret object, which I called `fit_iris`, dollar sign, `finalModel`.

So this is giving us information about that model with a cost of 0.1 fit on the entire data set. And this is telling us that we used a linear kernel function. That's correct. That's what we wanted to use.

We were doing classification. That's good. We were trying to classify two different types of irises. And that we ended up with 51 support vectors.

We can see which data points were the support vectors by looking at the `SVindex` attribute of the final model. So this will give us the indices of which data points were the support vectors.

We can see those data points by making a graph highlighting the support vectors. To do this, we'll start by using the `row.names` to `column` function from the `tibble` package to convert the row numbers into a new column, which I'll call `row`, so that we can work with it to compare the row numbers to the support vector list.

Then I'm using the `mutate` function to create a new column called `is_SV`. So if the row number is in that list of support vectors, then I'm going to call the row a support vector.

Now we can make our modified graph. So this second part with `gf` point is the same as what we did earlier, but I'm adding in a new part where I'm filtering to just the rows where `is_SV` equals `true`, meaning those data points are support vectors. And then I'm graphing petal length versus sepal length with a plotting character based on the species.

But I'm just leaving the color as the default, black. And I'm using `size` equals 2.5 to make these plotting characters a little bit bigger than the plotting characters we use for all the data.

So what this does is it plots a black triangle or circle underneath the data point, so that once we plot, the blue triangle or red circle on top of it, we get sort of a highlight with a black outline. So here, we can see all of the support vectors highlighted in black, and we can see that they are all of the data points that are

sort of near the division between the two species.

Another way we can understand our model is by plotting the support vector classifier itself. To do this, we need to start by scaling the variables that we used-- in this case, petal length and sepal length.

Then we can get the B attribute from our final model. So this will be information to give us our y -intercept of our model. This is the negative value of β_0 .

Then we can get the coefficients of our final model, and look at the double square bracket 1 component of that. So this is just a vector of negative and positive 0.1, because our optimal cost ended up being 0.1. So just negative and positive that value.

Next, we want to extract the support vectors. So we're again filtering to just the rows that are support vectors. Selecting the columns that are the scaled sepal length and petal length, and converting that to a matrix.

So this gets us a matrix of the scaled values of sepal length and petal length for all the support vectors. And in order for the graph to work out correctly, we need these columns to be in the order x , comma, y , relative to how we're going to graph them. So I'm going to graph my data in the order sepal length on the x -axis, petal length on the y -axis.

So the reason we just did that-- getting the scaled sepal length and petal length-- was so that we could multiply that by the coefficients-- the plus and minus 0.1-- and then take the sums of those two columns. So this gets us our vector w , which contains β_1 and β_2 .

So our support vector classifier is β_0 plus β_1 , which is w_1 times x_1 , plus β_2 , which is w_2 times x_2 equals 0. So here, x_1 is our first predictor variable-- sepal length-- and x_2 is our second predictor variable-- petal length.

In order to graph this line, we want to get it into slope intercept format. So we can use some basic algebra to rearrange this equation to get it into the form x_2 equals negative w_1 over w_2 , times x_1 , minus β_0 over w_2 .

Remember that b was negative β_0 , so this becomes x_2 equals negative w_1 over w_2 , times x_1 . So our slope is negative w_1 over w_2 , plus our y -intercept of b over w_2 .

So then we can graph these lines. So I'm starting with a scatter plot using `gf_point` as before. Then I'm using `gf_abline` to plot the lines. So here, we have a y-intercept of b over w_2 just like we had in our equation, and a slope of negative w_1 over w_2 .

I'm also graphing the margin lines which have the same slope, and their y-intercepts are just b plus 1 over w_2 , and b minus 1 over w_2 . And here, the plus and minus 1 stay the same regardless of the cost of the classifier.

So there is our support vector classifier. This graph is in terms of the units of the scaled sepal length and petal length. What if we wanted a graph that used the original sepal length and petal length? In that case, we would need to convert the slope and y-intercept of the support vector classifier to the units of the unscaled data.

To make this a bit easier, we'll start by letting x be the scaled sepal length of all the data that is not just the support vectors, and letting y be the unscaled petal length of all the data. We compute the standard deviation ratio-- that's the ratio of the standard deviation of y to the standard deviation of x .

And then we compute these slightly more complicated formulas. So the new slope is just the original slope from the scaled data times the standard deviation ratio. And the y-intercept is this somewhat more complicated formula based on the y-intercept that we had, along with the means and standard deviations of the data.

The y-intercepts of the margin lines are computed in the exact same way. So it's the same formula that we had for the y-intercept, but instead of b , we now have b plus 1 , or B minus 1 . So now we can plot the unscaled petal length and sepal length, and get our graph of our support vector classifier.

To view the predictions of our model, we can use the `predict` function. And this will predict the class that each point belongs to. So in this case, our first six data points were all predicted to be versicolor irises. So we can compute a confusion matrix in the same way as we've done for other methods by making a table of our predictions and the true value of the species.

So here's our confusion matrix. We can compute the accuracy, as we've done before, by taking the sum of the diagonal entries of the confusion matrix, and dividing it by the total number of data points in our data set. So in this case, this will be 48 plus 46 , divided by 100 .

So here we get an accuracy of 0.94 . But recall from our caret output that we had an accuracy of 0.93 . So why doesn't the confusion matrix give us 0.93 ?

This is because the confusion matrix uses the predictions about the full data set using the final model that was built on the full data set.

So we expect this to be higher than the accuracy on new data. Whereas caret is reporting the cross-validation accuracy, which is more representative of the accuracy on new data. So it's expected that our confusion matrix gives us a slightly higher accuracy than caret does.

Summary

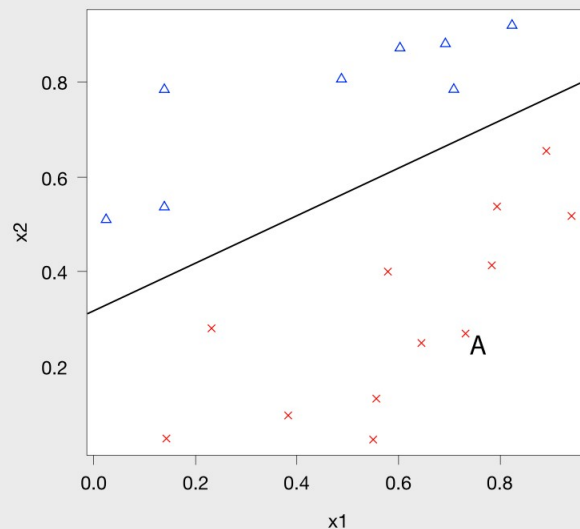
- A maximal margin hyperplane is the separating line/plane/hyperplane that maximizes the *margin*, or smallest distance to any data point.
- Support vector classifiers generalize this idea by allowing some points to be on the wrong side of the margin.
- A higher cost (for points on the wrong side of the margin) will result in narrower margins and a better fit to the training data, but higher variance.
- Use cross-validation to choose the optimal cost.

Question 1 Answer

Feedback for Self Assessment

✓ Correct!

If we added a small amount of noise to point A, how would the maximum margin hyperplane change?



Your answer:

Neither the slope nor the y-intercept would change.

Correct answer:

Neither the slope nor the y-intercept would change.

Feedback:

Correct! Adding a small perturbation to a point far from the line does not change the line, because we choose the line that maximizes the margin, or distance from the closest point.

Question 2 Answer

Feedback for Self Assessment

✓ Correct!

For a given data set, what do we expect to happen to the number of support vectors as the cost increases?

Your answer:

It will decrease.

Correct answer:

It will decrease.

Feedback:

Right! As the cost increases, the margins tend to get narrower, so the number of data points inside the margins will tend to decrease. However, if the two categories aren't separable, there will always be some points that are misclassified.