Lab 9: Support Vector Machines

Introduction

In this lab, we will practice using the method of support vector machines for classification. To visualize some easy two-dimensional examples, we'll begin using the `iris` data set. Then, we'll use support vector machines to classify images in a database of handwritten digits.

Support Vector Machine Lab

1. Load the `iris` data set (from the `datasets` package in R). As usual, create a `train` set consisting of 60% of the data and a `test` consisting of the remaining 40% of the data, ensuring that each species is represented proportionally in the test and train sets.

2. Create a scatterplot of `Petal.Length` versus `Sepal.Width` colored by `Species` using the data in `train`.

Our first task in this lab is going to be to try to distinguish observations that are **virginica** from observations that are **not virginica** using a *linear classifier*.

3. Let x_1 represent sepal width and x_2 represent petal length. Write below the equation of a line that you think will best separate virginica from non-virginica observations. Write your line in the form:

$$\beta_1 x_1 + \beta_2 x_2 + \beta_0 = 0.$$

4. Add your line to the plot in Problem 2.

To use this line as a classifier, we classify all the points above the line as "virginica" and all the points below the line as "not virginica." If the coefficient of x_2 in your classifier is positive, this means that we classify a new observation x as "virginica" if $\beta_1 x_1 + \beta_2 x_2 + \beta_0 > 0$ and "not virginica" if $\beta_1 x_1 + \beta_2 x_2 + \beta_0 < 0$ (otherwise it's the opposite).

5. Classify the points in the `test` set using your linear classifier. Print the resulting confusion matrix. Note that you'll have two classes for your predictions and three classes for the actual species.

This is essentially the idea behind support vector machines. The only difference is that when the data are in higher dimensions, the *linear classifier* is a hyperplane of the form

$$\beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \beta_0 = 0.$$

Using the package `e1071`, you can easily build support vector machines. If the data are in two dimensions, then you can also use this package to visualize the decision boundaries.

6. Uncomment the code below to train a support vector machine and to visualize the decision boundaries. **Note:** You should modify `cols` so that it includes the index of the columns for `Sepal.Width`, `Petal.Length`, and `Species`.

```
#library(e1071)
#cols <- c(2, 3, 5)
#iris2 <- iris[ , cols]
#iris2$Species <- factor(iris2$Species)

#svm.fit <- svm( Species ~ Petal.Length + Sepal.Width, data = iris2, kernel = "linea
r")

#plot(svm.fit, iris2)
```

You may notice that there are actually three classes in this problem. If there are more than two classes, then the `svm()` function uses a "one-against-one" approach. That means that it constructs a linear classifier for each pair of classes. To classify a new observation, each linear classifier is applied to to the observation, and the classifiers "vote" to decide how to classify the new observation.

You might also notice that we specified `kernel = linear`. Another powerful way to use support vector machines is to first transform the data via a *kernel function* and then to use linear classifiers on the transformed data. If you use a non-linear kernel function, this results in non-linear decision boundaries in the original (untransformed) space. This is discussed more in Section 9.3.1 of the text.

The Digits Data Set

For a more interesting data set, we're going to use one that consists of images of handwritten 4's and 9's. This data set is compiled from a larger data set of handwritten characters published by the National Institute of Standards and Technology. Each of the handwritten digits has been converted into a grayscale grid image represented by a 28×28 matrix. The numerical matrix entries correspond to how dark that square of the grid should be.

For each image, the rows were then appended to create a vector of length $28^2 = 784$. These vectors are stored in the data frame `test_numerals.csv` and `train_numerals.csv`. These data frames have 785 columns, because there is also a `label` column indicating whether the image is a 4 or 9.

7. Download the data sets `test_numerals.csv` and `train_numerals.csv`.
8. First, we'd like to plot one of these images. There are probably many ways to do this, but the easiest is to represent each image as a heatmap. So, to do this:

- Create a *numerical vector* of length 784 from one of the rows of `train_numerals.csv` that you would like to plot.
- Convert this vector to a 28×28 numerical matrix.
- Plot your matrix below using the `plot()` function from the `plot.matrix` package

```
#library(plot.matrix)
```

9. Now, build a support vector machine model to classify each image using the `svm` function as we did for the `iris` dataset. Obviously, you won't be able to plot the decision boundaries, because the data live in 784-dimensional space.

Notes:

- You will want to first convert the `label` column of your test and train sets to factor vectors. This is probably what is going wrong if this takes a really really long time.
- You will want to specify the option `scale = FALSE` to avoid getting warnings in `svm()`. Usually, it's a good idea to scale your data by dividing each column by the mean. But here, we have some all zero columns, so this will result in dividing by zero.

10. Use the `predict` function to classify the points in the test set and print the confusion matrix here. What is the accuracy of your model?
11. Assuming you didn't get perfect accuracy, print below a 4 that was misclassified as a 9 (it's fun to search around for the "nine-iest" looking four you can find).

What?! That's a terrible looking four (I assume). Given how different people's handwriting is, it's kind of impressive that support vector machines do so well at classification even on a relatively small data set.