Classification Exercises

Simone Collier

Getting Started

This section includes the code that was shown in class along with more tips and tricks. Go through this and run the code chunks before moving on to the exercises.

Start by loading the packages that have the data we need. If you need to install the packages first then run `install.packages("PACKAGENAME")` in your console before running the code chunk.

```
library(ISLR2)
library(MASS)
library(e1071)
library(class)
```

We will be making use of the Smarket data set which has the percentage returns for 1,250 days from the years 2001-2005. We are interested in predicting the qualitative response `Direction` as `Up` or `Down` using the other variables in the dataset. Use the `?` tool to find what each of the variables recorded mean.

```
# This allows us to call the variables direction without using `Smarket$...`
attach(Smarket)
?Smarket
```

In order to run classification methods on this data set, it is a good idea to separate the data into training and testing sets. This allows us to get an idea of the accuracy of our classification models with both the training and testing error rates. Our training data set will span 2001-2004 and out test set will be the data from 2005.

```
train <- (Year < 2005)
Smarket.2005 <- Smarket[!train, ]
Direction.2005 <- Smarket.2005$Direction
```

Logistic Regression

We will fit a logistic regression model to our training data to predict `Direction` using all the Lag variables and `Volume`. We can use the `glm()` function to fit a variety of generalized linear models. By specifying `family = binomial` we are instructing R to run a logistic regression. `subset = train` tells R to only run the regression on the training data subset of `Smarket`.

```
log.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
               data = Smarket, family = binomial, subset = train)
summary(log.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket, subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.302  -1.190   1.079   1.160   1.350
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.191213   0.333690   0.573    0.567
## Lag1        -0.054178   0.051785  -1.046    0.295
## Lag2        -0.045805   0.051797  -0.884    0.377
## Lag3         0.007200   0.051644   0.139    0.889
## Lag4         0.006441   0.051706   0.125    0.901
## Lag5        -0.004223   0.051138  -0.083    0.934
## Volume      -0.116257   0.239618  -0.485    0.628
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1383.3  on 997  degrees of freedom
## Residual deviance: 1381.1  on 991  degrees of freedom
## AIC: 1395.1
##
## Number of Fisher Scoring iterations: 3
```

The column labelled `Pr(>|z|)` gives the p-values associated with each variables. Recall that the p-values indicate whether or not to reject the null hypothesis that there is no association between the response and predictor variable.

Is there evidence of an association between any of the predictor variables and the response? If so, which ones?

The `predict()` function can be used to predict the probability of the market direction given the value of the predictors. We pass `type = "response"` into the `predict()` function in order to output the probabilities in the form $P(Y = 1 \mid X)$. Use the `contrasts()` function to determine how R has assigned the dummy variables to the qualitative predictor `Direction`.

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

```
log.probs <- predict(log.fit, type = "response")
head(log.probs)
```

```
##         1         2         3         4         5         6
## 0.4985061 0.4893137 0.4849791 0.5098788 0.5145737 0.4982044
```

Using the probabilities, we can set a threshold and make a table of predictions. Let's use a threshold of 0.5 to start which means that the market direction will be predicted to o up if the probability that day is greater than 0.5.

```
# Create a vector of length equal to the number of days in the data set with each element as "Down"
log.pred <- rep("Down", nrow(Smarket))
# Change the days with a probability greater than 0.5 to "Up
log.pred[log.probs > 0.5] = "Up"
head(log.pred)
```

```
## [1] "Down" "Down" "Down" "Up"   "Up"   "Down"
```

We can make a confusion matrix from the predictions using `table()` in order to determine how many observations were correctly or incorrectly classified.

```
table(log.pred, Direction)
```

```
##         Direction
## log.pred Down  Up
##     Down  209 191
##     Up    393 457
```

The diagonal entries are the correct predictions and the off-diagonals are the incorrect predictions. The `mean()` function can be used to compute the fraction of days for which the market direction prediction was correct.

```
mean(log.pred == Direction)
```

```
## [1] 0.5328
```

Note that 100% - 53.3% = 47.7% is the training error rate.

Now we can try predictin the outcomes of the test data using `predict(log.fit, Smarket.2005, type = "response")`. ***Try this out yourselves! Find the confusion matrix and test error rate as well.***

How does the training error rate compare to the test error rate?

Is logistic regression method good at predicting the direction of the market? Why or why not? Use the training/testing error rate to support your answer.

Linear Discriminant Analysis (LDA)

We will perform LDA on the Smarket data in order to predict `Direction` using `Lag1` and `Lag2`. We can fit an LDA model using the `lda()` function which belongs to the `MASS` library.

```
lda.fit <- lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
```

```
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

- The prior probabilities for each class (i.e. Up, Down) indicate the proportion of the training observations that belong to each class. For example $\hat{\mu}_1 = 0.492$ indicates that 49.2% of the training observations correspond to days where the market went down.
- The group means are the average of each predictor within each class which are used as estimates of μ_k.
- The coefficients of linear discriminants are used to form the LDA decision rule. So, if -0.642×`Lag1` - 0.515×`Lag2` is large then the LDA classifier will predict a market increase and vice versa.

We can use the `predict()` function to use our LDA model on our test data set.

```
lda.pred <- predict(lda.fit, Smarket.2005)
```

`lda.pred` contains three elements

- `class`: the predictions for the market direction each day in 2005
- `posterior`: a table (or matrix) with a row for for each observation and two columns labelled `Up` and `Down` which give the probabilities that the market will go up or down on the specified day.
- `x`: linear discriminants.

We can create a confusion matrix from `class` and find the test error rate.

```
lda.class <- lda.pred$class

table(lda.class, Direction.2005)
```

```
##          Direction.2005
## lda.class Down  Up
##      Down   35  35
##      Up     76 106
```

```
mean(lda.class == Direction.2005)
```

```
## [1] 0.5595238
```

The test error rate of the LDA model is 44%.

Try fitting a LDA model using predictor variables of the Smarket data of your choice. Discuss the results.

Quadratic Discriminant Analysis (QDA)

We will fit a QDA model to the `Smarket` usin the `qda()` function within the `MASS` library.

```
qda.fit <- qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

Note that the `qda()` output is similar to the `lda()` output except it does not contain the coefficients of the linear discriminants since the QDA classifier is quadratic. We can predict the market direction for the 2005 data using the same method as in the LDA case.

```
qda.pred <- predict(qda.fit, Smarket.2005)
qda.class <- qda.pred$class
table(qda.class, Direction.2005)
```

```
##          Direction.2005
## qda.class Down  Up
##      Down   30  20
##      Up     81 121
```

```
mean(qda.class == Direction.2005)
```

```
## [1] 0.5992063
```

The test error rate of the QDA model is 40%.

Try fitting a QDA model using predictor variables of the Smarket data of your choice. Discuss the results.

Naive Bayes

We will fit a naive Bayes model to the Smarket data using the `naiveBayes()` function which is part of the `e1071` library.

```
naiveB.fit <- naiveBayes(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
naiveB.fit
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
```

```
##
## A-priori probabilities:
## Y
##     Down       Up
## 0.491984 0.508016
##
## Conditional probabilities:
##       Lag1
## Y             [,1]     [,2]
##   Down  0.04279022 1.227446
##   Up   -0.03954635 1.231668
##
##       Lag2
## Y             [,1]     [,2]
##   Down  0.03389409 1.239191
##   Up   -0.03132544 1.220765
```

The conditional probability tables in the output give the mean (column 1) and standard deviation (column 2) for each of the predictors in each class.

```
naiveB.class <- predict(naiveB.fit, Smarket.2005)
table(naiveB.class, Direction.2005)
```

```
##             Direction.2005
## naiveB.class Down  Up
##         Down   28  20
##         Up     83 121
```

```
mean(naiveB.class == Direction.2005)
```

```
## [1] 0.5912698
```

The test error rate of the naive Bayes model is 41%.

The `predict()` function is also capable of generating estimates for the probability that an observation belongs to each class.

```
naiveB.pred <- predict(naiveB.fit, Smarket.2005, type = "raw")
head(naiveB.pred)
```

```
##           Down        Up
## [1,] 0.4873164 0.5126836
## [2,] 0.4762492 0.5237508
## [3,] 0.4653377 0.5346623
## [4,] 0.4748652 0.5251348
## [5,] 0.4901890 0.5098110
## [6,] 0.4911907 0.5088093
```

Try fitting a naive Bayes model using predictor variables of the Smarket data of your choice. Discuss the results.

K-Nearest Neighbors

We can fit a KNN model using the `knn()` function which belongs to the `class` library. Unlike the previous three classification methods we have been running, this function fits and runs predictions in one step. There are four inputs that are required:

- A data frame or matrix of the predictors in the training data.
- A data frame or matrix of the predictors in the test data for which we want to make predictions on.
- A vector containing the true classification of the training data.
- An integer indicating the number of nearest neighbors to be considered by the classifier

```
train.X <- cbind(Lag1, Lag2)[train, ]
test.X <- cbind(Lag1, Lag2)[!train, ]
train.Y <- Direction[train]
```

Now that we have three of our inputs we can predict the direction of the market in 2005 using K=1. We set a random seed to ensure our results are reproducible since if there are several points that are the nearest then R will choose one of the them randomly.

```
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Y, k = 1)
table(knn.pred, Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```
mean(knn.pred == Direction.2005)
```

```
## [1] 0.5
```

As we can see KNN for K=1 only gives 50% accuracy which is no better than random chance. ***Try running KNN for several values of K and summarize the results for the best model you find.***

Out of all the classification methods we tried, which performs best on the `Smarket` data? Give some explanation for why that might be.

These exercises were adapted from : James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R, 2nd ed., Springer, 2021.