Introduction to Statistical Learning 6/26/2017

Chapter 4: Classification

Name: roninlaw

Email: roninlaw

1

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \Rightarrow 1 - p(X) = \frac{1}{1+e^{\beta_0+\beta_1 X}}$$
$$\Rightarrow \frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}$$

4

(a) 10%

(b) $(10\%)^2 = 1\%$

(c) $\frac{1}{10}^{100} * 100\%$

(d) As p increases the number of points next to any given point decreases. This is a problem because it means the points that we would be using to determine the class of the point are not actually close the point we are classifying.

(e)

$$s^p = \frac{1}{10} \Rightarrow s = \frac{1}{10}^{\frac{1}{p}}$$

As p increases, s approaches 1 which means there is almost no constraint on the value of the p-th predictor.

5

(a) Train: QDA, Test: LDA

(b) Train: QDA Test: QDA

(c) Improve. LDA is a classifier with a very non-flexible decision boundary, which means the model will not be able to fit a large number of data points as well as a model with a more flexible decision boundary.

(d) False. Because the QDA is more flexible we can expect the train error rate to be better, but if the true decision boundary of the data is linear then LDA will produce better results

6

(a)

$$\begin{aligned}\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 &= -6 + .05(40) + 1(3.5) \\ &= -.5 \\ &\Rightarrow \frac{e^{-.5}}{1 + e^{-.5}} \\ &= .377 \\ &= 37.7\%\end{aligned}$$

(b) Let $k = e^{-2.5+\frac{H}{20}}$

$$\begin{aligned}\frac{k}{1+k} = .5 &\Rightarrow k = .5 + .5k \\ &\Rightarrow .5k = .5 \\ &\Rightarrow k = 1 \\ &\Rightarrow e^{-2.5+\frac{H}{20}} = 1 \\ &\Rightarrow -2.5 + \frac{H}{20} = 0 \\ &\Rightarrow H = 50\end{aligned}$$

7

$$\begin{aligned}P(\text{D} = \text{"Yes"}|X = 4) &= \frac{P(X = 4|\text{D} = \text{"Yes"}) * P(\text{D} = \text{"Yes"})}{P(X = 4)} \\ &= \frac{P(X = 4|\text{D} = \text{"Yes"}) * P(\text{D} = \text{"Yes"})}{P(X = 4|\text{D} = \text{"Yes"}) * P(\text{D} = \text{"Yes"}) + P(X = 4|\text{D} = \text{"No"}) * P(\text{D} = \text{"No"})} \\ &= \frac{.8 \cdot \frac{1}{\sqrt{2\pi}\cdot 6} \cdot e^{-\frac{(4-10)^2}{72}}}{.8 \cdot \frac{1}{\sqrt{2\pi}\cdot 6} \cdot e^{-\frac{(4-10)^2}{72}} + .2 \cdot \frac{1}{\sqrt{2\pi}\cdot 6} \cdot e^{-\frac{(4-0)^2}{72}}} \\ &= .757 \\ &= 75.7\%\end{aligned}$$

8

I would go with the second model because the probability that the test error rate is less than 30% based on the fact that the average error rate between the two sets is 18% is greater than $\frac{1}{2}$

9

(a)

$$\begin{aligned}\frac{p(X)}{1 - p(X)} &= .37 \\ p(X) &= .27\end{aligned}$$

(b) odds $= \frac{p(X)}{1-p(X)} = \frac{.16}{.84} = .1904$

10

(a)

```
Weekly = read.csv("Weekly.csv", header = T, na.strings = "?")
attach(Weekly)
cor(Weekly[,-9])
plot(Year, Lag1)
plot(Lag1, Lag2)
plot(Year, Volume)
```

The biggest pattern I noticed was the relationship between Year and Volume. As time went on the data suggests that the volume of shares purchases sky rocketed.

(b)

```
glm.fit = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data = Weekly, family =
    binomial)
summary(glm.fit)
```

Lag2 is the only statistically significant predictor.

(c)

```
dim(Weekly)
contrasts(Direction)
glm.probs = predict(glm.fit,type="response")
glm.pred = rep("Down",1250)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred,Direction)
mean(glm.pred == Direction)
```

The confusion matrix is telling us how the algorithm is misclassifying points. That is whether it is classifying points as "Up" when they are actually "Down" or vice versa. The confusion matrix is telling us that a lot of the error in the model is coming from the points being misclassified as "Up" instead of "Down". The confusion matrix is also telling us that the model does a great job at classifying points as "Down" correctly.

(d)

```
train = Year < 2009
glm.fit = glm(Direction ~ Lag2, family = binomial, subset = train)
Weekly.Test = Weekly[!train,]
dim(Weekly.Test) # check dimensions for table() call
glm.pred = rep("Down", 104)
glm.probs = predict(glm.fit, Weekly.Test, type = "response")
glm.pred[glm.probs > .5] = "Up"
table(glm.pred, Direction.Test)
mean(glm.pred==Direction.Test)
```

(e)

```
library(MASS)
lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.pred = predict(lda.fit, Weekly.Test)
names(lda.pred)
lda.class = lda.pred$class
table(lda.class, Direction.Test)
mean(lda.class == Direction.Test)
```

(f)

```
library(class)
qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
```

```
qda.class = predict(qda.fit, Weekly.Test)$class
table(qda.class, Direction.Test)
mean(qda.class == Direction.Test)
```

(g)

```
train.x = as.matrix(Lag2[train])
test.x = as.matrix(Lag2[!train])
train.direction = Direction[train]
set.seed(1)
knn.pred = knn(train.x, test.x, train.direction, k = 1)
table(knn.pred, Direction.Test)
mean(knn.pred == Direction.Test)
```

(h) The QDA model appears to provide the best results on the data.

(i)

```
pairs(Weekly)
# LDA w/ Lag2:Volume Interaction
lda.fit = lda(Direction ~ Lag2:Volume, data = Weekly, subset = train)
lda.class = predict(lda.fit, Weekly.Test)$class
table(lda.class, Direction.Test)
mean(lda.class == Direction.Test) # 0.605

# LDA w/ Lag2^2 + Volume
lda.fit = lda(Direction ~ I(Lag2^2) + Volume, data = Weekly, subset = train)
lda.class = predict(lda.fit, Weekly.Test)$class
table(lda.class, Direction.Test)
mean(lda.class == Direction.Test) # 0.451

# QDA w/ Lag2:Volume Interaction
qda.fit = qda(Direction ~ Lag2:Volume, data = Weekly, subset = train)
qda.class = predict(qda.fit, Weekly.Test)$class
table(qda.class, Direction.Test)
mean(qda.class == Direction.Test) # .548

# KNN w/ k = 10
knn.pred = knn(train.x, test.x, train.direction, k = 100)
mean(knn.pred == Direction.Test) # 0.557
```

11

(a)

```
Auto = read.csv("Auto.csv", header = T, na.strings = "?")
attach(Auto)
dim(Auto)
mpg01 = rep(0, 392)
mpg01[mpg > median(mpg)] = 1
Auto = data.frame(Auto, mpg01)
```

(b)

```
pairs(Auto)
cor(subset(Auto,select=-name))
plot(mpg01, weight)
plot(mpg01, displacement)
```

mpg01 has a strong negative relationship with cylinders, displacement, weight, and horsepower. For example in the plot between mpg01, and displacement we see that cars with mpg over the median have an overall lower displacement.

(c)

```
train = (year <= 79)
Auto.Test = Auto[!train,]
dim(Auto.Test)
```

(d)

```
mpg01.Test = mpg01[!train]
lda.fit = lda(mpg01 ~ cylinders + displacement + weight + horsepower, data = Auto,
    subset = train)
lda.class = predict(lda.fit, Auto.Test)$class
table(lda.class, mpg01.Test)
mean(lda.class != mpg01.Test) # 0.1294
```

(e)

```
qda.fit = qda(mpg01 ~ cylinders + displacement + weight + horsepower, data = Auto,
    subset = train)
qda.class = predict(qda.fit, Auto.Test)$class
table(qda.class, mpg01.Test)
mean(qda.class != mpg01.Test) # 0.1294
```

(f)

```
glm.fit = glm(mpg01 ~ cylinders + displacement + weight + horsepower, data = Auto,
    family = binomial, subset = train)
glm.probs = predict(glm.fit, Auto.Test, type = "response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > .5] = 1
mean(glm.pred != mpg01.Test) # 0.176
```

(g)

```
# KNN w/ k = 1
train.x = cbind(cylinders, displacement, weight, horsepower)[train,]
test.x = cbind(cylinders, displacement, weight, horsepower)[!train,]
train.mpg01 = mpg01[train]
set.seed(1)
knn.pred = knn(train.x, test.x, train.mpg01, k=1)
mean(knn.pred != mpg01.Test) # .2

# KNN w/ k = 50
knn.pred = knn(train.x, test.x, train.mpg01, k=50)
table(knn.pred, mpg01.Test)
mean(knn.pred != mpg01.Test) # 0.188

# KNN w/ k = 100
knn.pred = knn(train.x, test.x, train.mpg01, k=100)
mean(knn.pred != mpg01.Test) # 0.223

# KNN w/ k = 200
knn.pred = knn(train.x, test.x, train.mpg01, k=200)
mean(knn.pred != mpg01.Test) # 0.176
```

The largest value of k performed best on this dataset.

12

(a)

```
Power = function() {
   print(2^3)
}
```

(b)

```
Power2 = function(x,a) {
   print(x^a)
}
```

(c)

```
Power2(10,3) = 1000
Power2(8,17) = 2.2518e+15
Power2(131,3) = 2248091
```

(d)

```
Power3 = function(x,a) {
   return(x^a)
}
```

(e)

```
x = c(1,2,3,4,5,6,7,8,9,10)
plot(x, Power3(x,2), log = "xy", xlab = "log(x)", ylab = "log(x^2)")
```

(f)

```
PlotPower = function(x,a) {
   plot(x,Power3(x,a))
}
```

13

```
Boston = read.csv("BostonHousing.csv", header = T, na.strings = "?")
train = 1: (dim(Boston)[1]/2)
Boston.train = Boston[train,]
test = (dim(Boston)[1]/2): dim(Boston)[1]
Boston.Test = Boston[test, ]
crim01.Test = crim01[test]
# Logistic Regression w/ nox
glm.fit = glm(crim01 ~ nox, family = binomial, subset = train)
glm.probs = predict(glm.fit, Boston.Test, type = "response")
glm.pred = rep(0, dim(Boston.Test)[1])
glm.pred[glm.probs > .5] = 1
mean(glm.pred != crim01.Test) # .1102
# LDA w/ nox
lda.fit = lda(crim01 ~ nox, data = Boston, subset = train)
lda.class = predict(lda.fit, Boston.Test)$class
mean(lda.class != crim01.Test) # .1417
#LDA w/ nox*age
lda.fit = lda(crim01 ~ nox*age, data = Boston, subset = train)
lda.class = predict(lda.fit, Boston.Test)$class
mean(lda.class != crim01.Test) # .137
table(lda.class, crim01.Test)
```

```r
#KNN w/ k = 10
train.x = as.matrix(nox[train])
test.x = as.matrix(nox[test])
set.seed(1)
knn.pred = knn(train.x, test.x, train.crim01, k = 10)
mean(knn.pred != crim01.Test) # .078
table(knn.pred, crim01.Test)
# KNN w/ k = 50
knn.pred = knn(train.x, test.x, train.crim01, k = 50)
mean(knn.pred != crim01.Test)# 0.17
```

The model that gave the best results on the data was KNN with $k = 10$. It seems like this means that you can gauge the amount of crime a specific area will have based on the nitrous oxide of the closest areas. Once we increased the value of k we see that the test error increases.