Homework 3 Key (100 Pts)

4/20/2018

Problem 1 (6 + 6 + 6 + 6 + 6 = 30 Pts)

(a)

Using Baye's theorem,

$$P(Y = 1|X = x) = \frac{P(X = x|Y = 1)P(Y = 1)}{P(X = x)}$$

where

$$P(X = x) = P(X = x|Y = 1)P(Y = 1) + P(X = 2|Y = 2)P(Y = 2) = \lambda_1 \exp(-\lambda_1 x)\pi_1 + \lambda_2 \exp(-\lambda_2 x)\pi_2$$

Then,

$$P(Y = 1|X = x) = \frac{P(X = x|Y = 1)P(Y = 1)}{P(X = x)} = \frac{\lambda_1 \exp(-\lambda_1 x)\pi_1}{\lambda_1 \exp(-\lambda_1 x)\pi_1 + \lambda_2 \exp(-\lambda_2 x)\pi_2}$$

(b)

Here, since $P(Y = 1|X = x) + P(Y = 2|X = x) = 1$, then

$$P(Y = 2|X = x) = 1 - P(Y = 1|X = x) = \frac{\lambda_2 \exp(-\lambda_2 x)\pi_2}{\lambda_1 \exp(-\lambda_1 x)\pi_1 + \lambda_2 \exp(-\lambda_2 x)\pi_2}$$

The decision boundary is

$$\lambda_1 \exp(-\lambda_1 x)\pi_1 = \lambda_2 \exp(-\lambda_2 x)\pi_2 \rightarrow (\lambda_1 - \lambda_2)x = \log\left(\frac{\lambda_1 \pi_1}{\lambda_2 \pi_2}\right)$$

Note here we are assuming that $\lambda_1 \neq \lambda_2$, since otherwise, $\lambda_1 = \lambda_2$, these two classes are indistinguishable.

(c)

Given this specific value, the decision boundary is then just

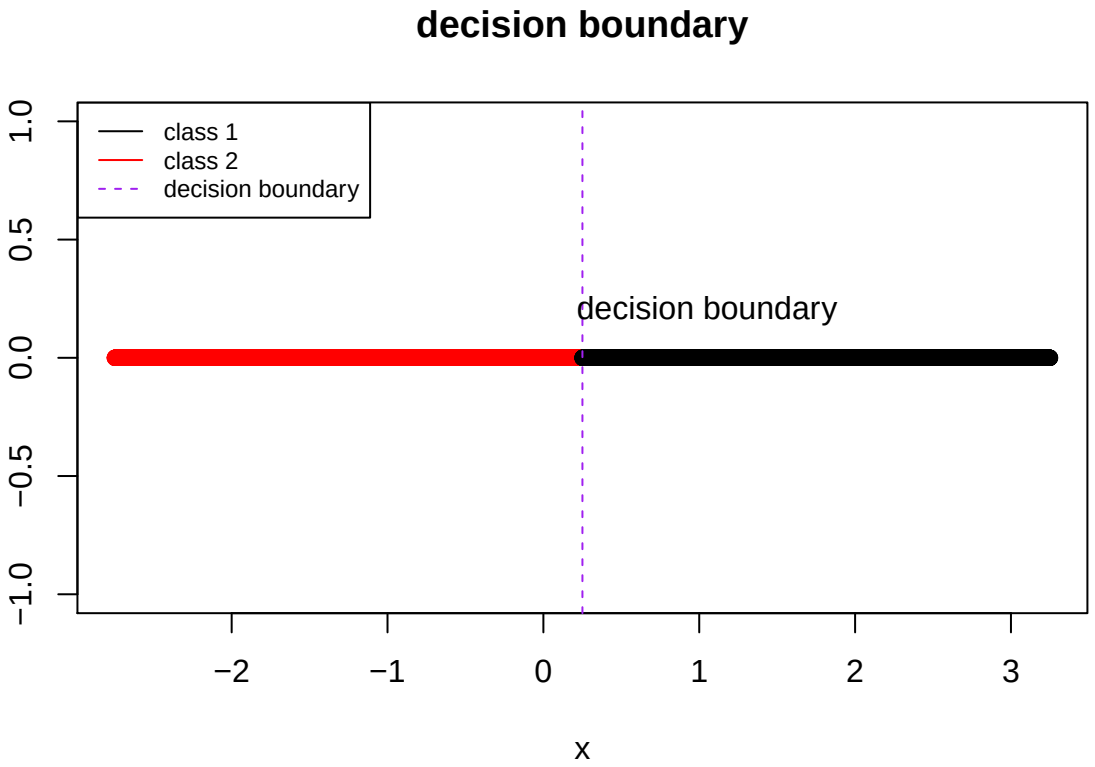
$$x = 0.2 \log_e(3.5)$$

and using bayes classifier,

$$Y = \begin{cases} 1 & x \geq 0.2 \log_e(3.5) \\ 2 & x < 0.2 \log_e(3.5) \end{cases}$$

The plot is following:

```
boundary = 0.2 * log(3.5)
x = seq(boundary - 3, boundary + 3, length = 10000)
class = sapply(x, FUN = function(x) if (x < boundary) 2 else 1)
plot(x, rep(0,10000), col=class, ylab="", main="decision boundary")
abline(v = boundary, lty = 2, col = "purple")
text(x = boundary + 0.8,y = 0.2, label = "decision boundary")
legend("topleft",legend=c("class 1","class 2","decision boundary"), col = c("black","red","purple"),
lty=c(1,1,2), cex = 0.75)
```



Notice here the feature space is actually one dimensional; so the decision boundary is just a point. I am plotting a line here to make it easier to see.

(d)

Since for exponential random variables, $X \sim \text{Exponential}(\lambda)$, $E[X] = \frac{1}{\lambda}$. Thus, define $n_1 = \sum_{i=1}^{n} \mathbb{1}_{y_i=1}$, $n_2 = n - n_1$, $\bar{X}_1 = \frac{1}{n_1}\sum_{i=1}^{n} x_i \mathbb{1}_{y_i=1}$ and $\bar{X}_2 = \frac{1}{n_2}\sum_{i=1}^{n} x_i \mathbb{1}_{y_i=2}$. So basically, n_1 is the number of observations in class 1, n_2 in class 2, and $\bar{X}_1$ is sample average of observations in class 1, similarly for $\bar{X}_2$. simple estimators for $\lambda_1, \lambda_2, \pi_1, \pi_2$ would be

$$\hat{\lambda}_1 = \frac{1}{\bar{X}_1} \qquad \hat{\lambda}_2 = \frac{1}{\bar{X}_2} \qquad \hat{\pi}_1 = \frac{n_1}{n} \qquad \hat{\pi}_2 = \frac{n_2}{n} = 1 - \hat{\pi}_1$$

(e)

Just using the results from part (d) and (a), an estimate of $P(Y = 1|X = x_0)$ would be

$$\hat{P}(Y = 1|X = x_0) = \frac{\hat{\lambda}_1 \exp(-\hat{\lambda}_1 x_0)\hat{\pi}_1}{\hat{\lambda}_1 \exp(-\hat{\lambda}_1 x_0)\hat{\pi}_1 + \hat{\lambda}_2 \exp(-\hat{\lambda}_2 x_0)\hat{\pi}_2}$$

and plug in the estimators and use the definitions in part(d) we obtian

$$\hat{P}(Y = 1|X = x_0) = \frac{\frac{1}{\bar{X}_1}\exp\left(-\frac{x_0}{\bar{X}_1}\right) n_1}{\frac{1}{\bar{X}_1}\exp\left(-\frac{x_0}{\bar{X}_1}\right) n_1 + \frac{1}{\bar{X}_2}\exp\left(-\frac{x_0}{\bar{X}_2}\right) n_2}$$

Problem 2 (4 + 6 + 8 = 18 Pts)

(a)

$$P(Y = A|X_1, X_2) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)}$$

(b)

Basically we want to have $P(Y = A|X_1, X_2) \geq 0.8$

$$\frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)} \geq 0.8 \rightarrow X_2 \geq \frac{\log 4 - \hat{\beta}_0 - \hat{\beta}_1 x_1}{\hat{\beta}_2}$$

assuming that $\hat{\beta}_2 > 0$ meaning study more should help you..

(c)

Here using this formulation for logistic regression, where

$$\log\left(\frac{p}{1-p}\right) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$$

Then assume $p_0 = 0.6$ and p is the probability of getting an A after she increases studying by one hour, then

$$\log\left(\frac{p}{1-p}\right) - \log\left(\frac{p_0}{1-p_0}\right) = \hat{\beta}_2$$

Through some derivations,

$$p = \frac{\frac{p_0}{1-p_0} e^{\hat{\beta}_2}}{1 + \frac{p_0}{1-p_0} e^{\hat{\beta}_2}} = \frac{1.5 e^{\hat{\beta}_2}}{1 + 1.5 e^{\hat{\beta}_2}}$$

Problem 3 (4 + 4 + 4 + 4 + 4 = 20 Pts)

(a)

Since X is uniformly distributed on $[0, 1]$, 10% of the range of X corresponds to on average 10% of the available observations that will be used to make a prediction

(b)

Now, since X_1 and X_2 are both uniformly distributed on $[0, 1]$, then 10% of the range on both X_1 and X_2 corresponds to on average $10\% \times 10\% = 1\%$ fraction of available observations that will be used.

(c)

Similarly, for this case, $p = 100$, then $10\%^{100} = 0.1^{100} = 10^{-100} = 10^{-98}\%$ fraction of the available observations will be used to make prediction.

(d)

The last two components of this question have illustrated a drawback of KNN when p is large–few training observations are "near" any given test observation. That is, in the setup described in part a, as p increases the fraction of training data used for a test point approaches 0%.

(e)

So basically, the volume for a p-dimenisonal hypercube with length r is r^p, then assume each feature is again on [0,1], to make the hypercube cover 10% of the training observations, you want

$$r^p = 0.1 \rightarrow r = (0.1)^{1/p} \approx \begin{cases} 0.1 & p = 1 \\ 0.3162 & p = 2 \\ 0.977 & p = 100 \end{cases}$$

Comment: As p increases, the length of the hypercube is increasing to cover the same amount of training observations; in the extreme case of p = 100, to cover 10% of the training observations, you would need a hypercube that covers approximately 98% of each feature's range. Again, this is a quite common phenomenon in high dimensional statistics.

Problem 4 (8 + 5 + 5 + 5 + 5 + 4 = 32 Pts)

(a)

```
library(ISLR)
data(Auto)
n = nrow(Auto)
p = ncol(Auto) - 2 # don't consider origin and response
Auto$origin = as.factor(sapply(Auto$origin, FUN = function(x) if (x == 1)
  "American" else "Non-American"))
```

Here, I am using the `Auto` data in `ISLR` library to predict a car's country of origin. Since there is actually three categories in this case, I would merge European and Japanese as non-American, and use all the features other than "name" to predict the origin.

The data frame contains 392 observations with 7 features.

Y is a car's country of origin, with two categories: American and non-American;

The seven features are :

- mpg: miles per gallon
- cylinders: Number of cylinders between 4 and 8
- displacement: Engine displacement (cu. inches)
- horsepower: Engine horsepower
- weight: Vehicle weight (lbs.)
- acceleration: Time to accelerate from 0 to 60 mph (sec.)
- year: Model year (modulo 100)

(b)

```
library(MASS)
set.seed(1)
train_index = sample(1:n, size = n/2)
train_data = Auto[train_index,]
test_data = Auto[-train_index,]
fit_LDA = lda(origin ~ . -name, data = train_data)
predict_train_LDA = predict(fit_LDA, train_data)
train_error_LDA = mean(predict_train_LDA$class != train_data$origin)
predict_test_LDA = predict(fit_LDA, test_data)
test_error_LDA = mean(predict_test_LDA$class != test_data$origin)
```

Here I am using half of the dataset as training data and another half as test data The training error using LDA is 0.097 and test error is 0.143.

(c)

```
fit_QDA = qda(origin ~ .-name, data = train_data)
predict_train_QDA = predict(fit_QDA, train_data)
train_error_QDA = mean(predict_train_QDA$class != train_data$origin)
predict_test_QDA = predict(fit_QDA, test_data)
test_error_QDA = mean(predict_test_QDA$class != test_data$origin)
```

The training error using QDA is 0.077 and test error is 0.122.

(d)

```
contrasts(Auto$origin)
```

```
##              Non-American
## American                0
## Non-American            1
```

```
fit_logistic = glm(origin ~ ., family = binomial, data = train_data[,-9])
predict_train_logistic_prob = predict(fit_logistic, train_data, type="response")
predict_train_logistic = rep("American", n/2)
predict_train_logistic[predict_train_logistic_prob > .5] = "Non-American"
train_error_logistic = mean(predict_train_logistic!= train_data$origin)

predict_test_logistic_prob = predict(fit_logistic, test_data[,-9], type="response")
predict_test_logistic = rep("American", n/2)
predict_test_logistic[predict_test_logistic_prob > .5] = "Non-American"
test_error_logistic = mean(predict_test_logistic!= test_data$origin)
```

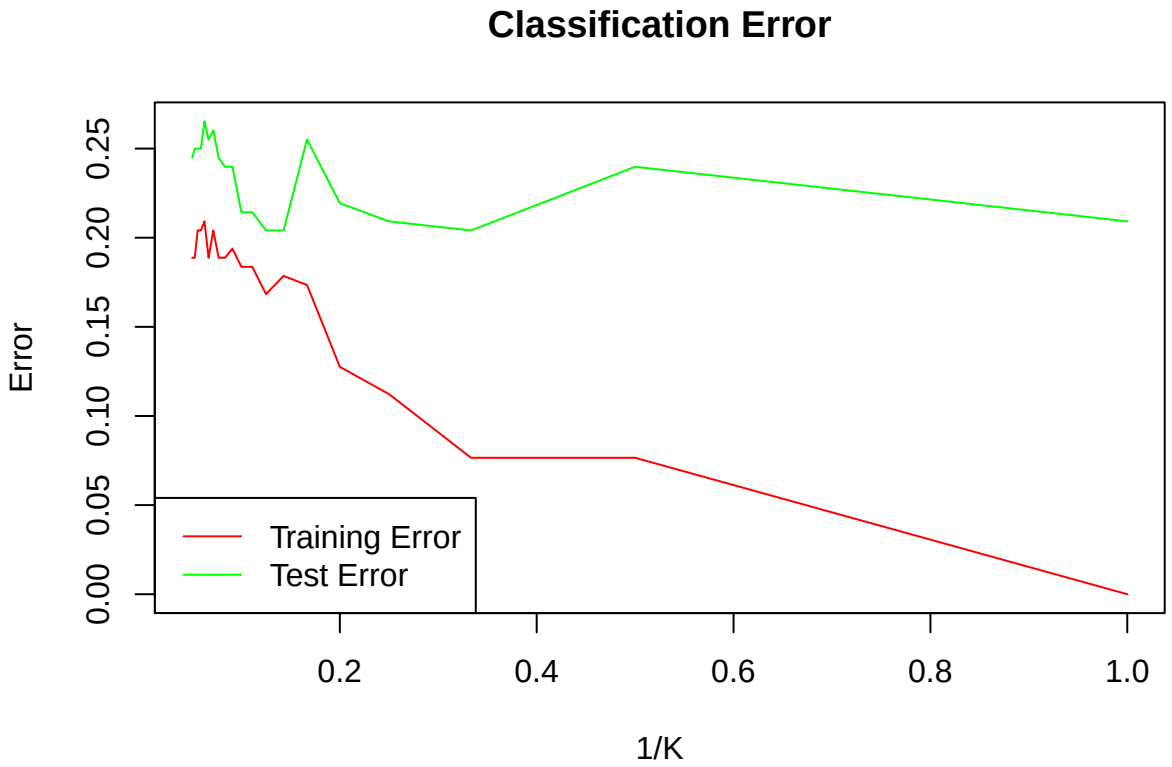
Using contrasts function, we could see that R has created a dummy variable with 1 for "Non-American" and "0" for "American". Then the training error for logistic regression is 0.092 and the test error is 0.168.

(e)

Here I run KNN with K from 1 to 20

```r
library(class)
K = seq(1:20)
knn_train_error = knn_test_error = rep(0, 20)
for (i in 1:20) {
  fit_knn_train = knn(train_data[,-c(8,9)], train_data[,-c(8,9)], train_data$origin, k = K[i])
  knn_train_error[i] = mean(fit_knn_train != train_data$origin)
  fit_knn_test = knn(train_data[,-c(8,9)], test_data[,-c(8,9)], train_data$origin, k = K[i])
  knn_test_error[i] = mean(fit_knn_test != test_data$origin)
}

plot(1/(1:20), knn_train_error, xlab="1/K", ylab="Error", col="red",type="l",
     main="Classification Error",ylim=range(c(knn_train_error,knn_test_error)))
lines(1/(1:20), knn_test_error, xlab="1/K", ylab="Error", col="green")
legend("bottomleft", c("Training Error", "Test Error"), col=c("red","green"), lty=c(1,1))
```



From the classification error plot, it seems that $K = 3$ might be a good candidate for KNN; the corresponding training error is 0.077 and the test error is 0.204.

(f)

```r
train_error = round(c(train_error_LDA, train_error_QDA, train_error_logistic, knn_train_error[3]),3)
test_error = round(c(test_error_LDA, test_error_QDA, test_error_logistic, knn_test_error[3]),3)
error = as.data.frame(cbind(train_error, test_error))
rownames(error) = c("LDA","QDA","logistic","KNN K = 3")
colnames(error) = c("training error", "test error")
library(pander)
pander(error)
```

	training error	test error
LDA	0.097	0.143
QDA	0.077	0.122
logistic	0.092	0.168
KNN K = 3	0.077	0.204

In terms of the training error, QDA and KNN with K = 3 both achieved the lowest error rate. However, with regard to test error, QDA performs much better than KNN with K = 3 and also performs better than LDA and logistic regression. It seems that QDA might actually capture some underlying data pattern and it is the best classifier for this specific dataset.