Classification – Computer exercises

4.1 Getting started with classification in R – Breast cancer diagnosis

In this exercise, we will consider the data set `biopsy` in the `MASS` library with data from breast biopsies, for the purpose of diagnosing breast cancer. For each patient, the data set contains nine different attributes (clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses) scored on a scale from 1 to 10, as well as the physician’s diagnosis (malign or benign).

(a) Load and familiarize yourself with the data set, using, e.g., `?biopsy`, `summary()`, `pairs()`, `names()`, and `print()`.
(b) Split the data randomly into a training set and a test set of approximately similar size.
(c) Perform logistic regression with `class` as output variable and `V3`, `V4` and `V5` as input variables. Do a prediction on the test set, and compute (i) the fraction of correct predictions and (ii) the confusion matrix. The commands `glm()`, `predict()` and `table()` are useful, and ISL section 4.6.2 can also be helpful. Is the performance any good, and what does the confusion matrix tell you?
(d) Repeat (c) using LDA. A useful command is `lda()` in the `MASS` library, introduced in ISL section 4.6.3.
(e) Repeat (c) using QDA. A useful command is `qda()` in the `MASS` library, introduced in ISL section 4.6.4.
(f) Repeat (c) using k-NN (with \( k = 1 \)). A useful commands is `knn()` in the `class` library (note the different syntax!), introduced in ISL section 4.6.5.
(g) Use a `for`-loop to explore the performance of k-NN for different values of \( k \), and plot the fraction of correct predictions as a function of \( k \).
(h) Use a `for`-loop to explore how the true and false positive rates in logistic regression are affected by different threshold values, and plot the result as a ROC curve (see Figure 4.8 and Table 4.6 and 4.7 in ISL).
(i) Try to find another set of inputs (perhaps by also considering transformations of the attributes) which gives a better result than you have achieved so far. You may also play with the threshold values. (“Better” is on purpose left vague. For this problem, the implications of a false negative (=benign) misclassification is probably more severe than a false positive (=malignant) misclassification.)
4.2 Decision boundaries

The following code generates some data with $x_1$ and $x_2$ both in $[0,10]$ and $y$ either 0 or 1, and plots the decision boundary for a logistic regression model.

```r
# generate data
set.seed(2); N = 100
x1 <- runif(n=N, min = 0, max = 10)
x2 <- runif(n=N, min = 0, max = 10)
y <- rep(1,N)
y[x1<4] <- 0; y[x2<4] <- 0

# learn a logistic regression model
model.fit <- glm(y~x1+x2,family="binomial",data=data.frame(y,x1,x2))

# open a plot with a good size
plot(x1,x2,type="n",main="logistic regression decision boundary")

# classify many points, and plot a colored square around each point
res <- 0.1 # resolution of the squares
for (xs1 in seq(0,10,res))
{
  for (xs2 in seq(0,10,res))
  {
    pred <- predict(model.fit,newdata=data.frame(x1=xs1,x2=xs2),type="response")
    if (pred>.5)
    {
      polygon(x=c(xs1-res/2,xs1+res/2,xs1+res/2,xs1-res/2),
        y=c(xs2-res/2,xs2-res/2,xs2+res/2,xs2+res/2),col=rgb(1,0,0,0.5),border=NA)
    } else
    {
      polygon(x=c(xs1-res/2,xs1+res/2,xs1+res/2,xs1-res/2),
        y=c(xs2-res/2,xs2-res/2,xs2+res/2,xs2+res/2),col=rgb(0,0,1,0.5),border=NA)
    }
  }
}

# plot the data
points(x1[y==0],x2[y==0],pch=16,col="blue"); points(x1[y==1],x2[y==1],pch=16,col="red")
```

(a) Run the code and verify that it reproduces the figure, and make sure you understand the figure. What is the misclassification rate?

(b) Modify the code to plot the decision boundary for a LDA classifier. What differences do you see? What is the misclassification rate?

(c) Modify the code to plot the decision boundary for a QDA classifier. What differences do you see? What is the misclassification rate?

(d) Modify the code to plot the decision boundary for a $k$-NN classifier. What differences do you see? What is the misclassification rate?

(e) What happens with the decision boundary for logistic regression if you include the term $x_1x_2$ as an input? What is the misclassification rate?
4.3 Why not linear regression?
In this exercise, we explore why linear regression might not be well suited for classification problems.

(a) Construct and plot a data set as follows: Let \( x_i \) be samples \( x_i = i \) in a sequence from \( i = 1 \) to \( i = 40 \). Let \( y_i = 0 \) for all \( i = 1 : 40 \), except for \( i = 34, 38, 39, 40 \) where \( y_i = 1 \). Hence, \( y \) belongs to either of two classes, 0 and 1.

(b) Now, the problem is to fit a model which is able to predict the output \( y \) from the input \( x \). Start with a linear regression model (command \( \text{lm()} \)), and simply threshold its predictions at 0.5 (the average of 0 and 1, the two classes). Plot the prediction. How good is the prediction?

(c) Try instead logistic regression (\( \text{glm()} \) command, see 4.6.2 in ISL for an introduction), and plot the prediction. How good is the prediction, and what advantages does logistic regression have over linear regression for this classification problem?

4.4 k-NN
In this exercise, we are going to explore an important user aspect of k-NN

(a) Make 200 draws \( x_1 \) from a \( \mathcal{N}(0, 1^2) \) distribution, and 200 draws \( x_2 \) from \( \mathcal{N}(0, 10^4) \). Also construct \( y \) such that \( y = 1 \) if \( x_1 \cdot x_2 \) is positive, and 0 otherwise. Split the data set randomly into a test and a training data set (equally sized).

(b) Use k-NN (choose \( k \) yourself) to predict the test output \( y \) using \( x_1 \) and \( x_2 \) as inputs. How well do you perform?

(c) Now replace \( x_2 \) with 200 draws \( x_2 \) from \( \mathcal{N}(0, 1^2) \), and perform k-NN classification anew. How well do you perform this time? Explain the difference!

(d) Explore how the \( \text{scale()} \) function can help for such problems encountered in (b)!

4.5 Multiclass classification
In the course, we have focused on the classification problem for 2 classes. The methods can, however, be generalized to more than two classes. In R, the commands \( \text{lda()} \), \( \text{qda()} \) and \( \text{knn()} \) can all be used directly for multi-class problems as well, which we will do in this exercise.

(a) Load and familiarize yourself with the data set \( \text{iris} \), and split it randomly into a training and a test data set.

(b) Use all inputs (\( \text{Sepal.Length} \), \( \text{Sepal.Width} \), \( \text{Petal.Length} \), \( \text{Petal.Width} \)) to predict the output \( \text{Species} \) (\text{setosa}, \text{versicolor} and \text{virginica}) using LDA, QDA, and k-NN, respectively.
Solutions

4.1 (a) `library(MASS)`
`?biopsy`
`summary(biopsy)`
`pairs(biopsy[,2:11])`

(b) `set.seed(1) # for reproducible results`
`training_indices <- sample(nrow(biopsy), size = 300, replace = FALSE)`
`biopsy.training <- biopsy[training_indices,]`
`biopsy.test <- biopsy[-training_indices,]`

(c) `glm.fit <- glm(formula = class ~ V3 + V4 + V5, data = biopsy.training, family = binomial)`
`glm.probs <- predict(object = glm.fit, newdata = biopsy.test, type = "response")`
`glm.pred <- rep("benign", length(glm.probs))`
`glm.pred[glm.probs > 0.5] <- "malignant"`
`table(biopsy.test$class, glm.pred)`
`mean(glm.pred == biopsy.test$class)`

94% correct classifications (but based on the number of false negative, i.e., 19 malignant tumors misleadingly predicted as benign, one should perhaps in a real situation consider changing the threshold).

(d) `library(MASS)`
`lda.fit <- lda(formula = class ~ V3 + V4 + V5, data = biopsy.training)`
`lda.testdata <- predict(object = lda.fit, newdata = biopsy.test)`
`lda.pred <- lda.testdata$class`
`table(biopsy.test$class, lda.pred)`
`mean(lda.pred == biopsy.test$class)`

92% correct classifications.

(e) `qda.fit <- qda(formula = class ~ V3 + V4 + V5, data = biopsy.training)`
`qda.testdata <- predict(object = qda.fit, newdata = biopsy.test)`
`qda.pred <- qda.testdata$class`
`table(biopsy.test$class, qda.pred)`
`mean(qda.pred == biopsy.test$class)`

93% correct classifications. QDA happens to produce fewer false negatives than LDA.
(f) library(class)
biopsy.training.KNN <- as.matrix(biopsy.training[c("V3","V4","V5")])
biopsy.test.KNN <- as.matrix(biopsy.test[c("V3","V4","V5")])
knn.pred <- knn(train = biopsy.training.KNN, test = biopsy.test.KNN,
cl = as.matrix(biopsy.training["class"]), k = 1)
table(biopsy.test$class, knn.pred)
mean(knn.pred == biopsy.test$class)

<table>
<thead>
<tr>
<th>( \hat{Y} = \text{Benign} )</th>
<th>( \hat{Y} = \text{Malignant} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>249</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>125</td>
</tr>
</tbody>
</table>

94% correct classifications.

(g) misclassification <- c()
for (kt in 1:50) #try k 1, 2, ..., 20
{
  knn.pred <- knn(train = biopsy.training.KNN, test = biopsy.test.KNN,
  cl = as.matrix(biopsy.training["class"]), k = kt)
misclassification[kt] = mean(knn.pred != biopsy.test$class)
}
plot(x = 1:50, y = misclassification, type="l",xlab="k")

It seems as if \( k \) around 20 would be optimal for this data set. (To reduce the uncertainty in the plot, this could be repeated multiple times over different selections of test and training data, which is the procedure called cross validation.)

(h) falsepositiverate <- c()
truepositiverate <- c()
N <- sum(biopsy.test$class == "benign")
P <- sum(biopsy.test$class == "malignant")
for (i in 1:99) #try threshold 0.01, 0.02, ..., 0.99
{
  glm.pred <- rep("benign", length(glm.probs))
  glm.pred[glm.probs > (i/100)] <- "malignant"

  FP <- sum((glm.pred == "malignant")*(biopsy.test$class == "benign"))
  TP <- sum((glm.pred == "malignant")*(biopsy.test$class == "malignant"))

  falsepositiverate[i] <- FP/N
  truepositiverate[i] <- TP/P
}
plot(x=falsepositiverate, y=truepositiverate,type="l",main="ROC curve")
Perfect classification is achieved in the upper left corner (and this ROC curve is not too far away from that corner).

(i) -

4.2 (a) In this problem, the misclassification rate for the logistic regression is 11% (the number of points that are in the wrong region in the figure)

(b) Misclassification rate 11%. Note that the decision boundaries for both logistic regression and LDA are linear, but not identical.

(c) Misclassification rate 7%. The decision boundary of QDA is not linear.
The misclassification rate is 0% (which always is the case when \( k = 1 \)). The misclassification rate for a test data set could still be much worse.

Misclassification rate 5%. Using nonlinear transformations of the inputs is one way to create a nonlinear decision boundary in a linear model. However, the decision boundary in a 3D-plot plot with axes \( x_1 \), \( x_2 \) and \( x_1 x_2 \) would still be linear.

All points are predicted to belong to class 0.
Logistic regression makes fewer misclassifications than linear regression, since the logistic function is more suited for the 0/1-classification setting than the linear function.

4.4 (a) 

```r
library(class)
N <- 200
X1 <- rnorm(n=N,mean=0,sd=1)
X2 <- rnorm(n=N,mean=0,sd=100)
y <- rep(x=0,times=N)
y[X1*X2>0] <- 1
X <- cbind(X1,X2)
```

(b) 

```r
knn.pred <- knn(train = X.train, test = X.test, cl = y.train, k = 2)
mean(knn.pred==y.test)
```

(c) $k$-NN is based on the Euclidian distance between data points. In our problem in (b), the values of $x_2$ is on average 100 times larger than the values of $x_1$, and hence does the prediction essentially only depend on $x_2$ (e.g., the distance between (0,1, 10) and (0,1, −10) is larger than the distance between (0,1,10) and (−0.1,−9), e.g, $x_1$ does effectively not matter when determining the $k$ nearest neighbors). However, since $y$ depends both on $x_1$ and $x_2$, the performance is deteriorated. Now, when removing the magnitude difference between $x_1$ and $x_2$, both inputs will impact the $k$-NN prediction equally.

(d) The command `scale()` can be used to normalize the inputs prior to classification: `X.normalized = scale(X)`. Note that both the training and test inputs have to be scaled!
?iris
pairs(iris)
summary(iris)

training_indices <- sample(nrow(iris), size = 100, replace = FALSE)
iris.training <- iris[training_indices,
iris.test <- iris[-training_indices,]

lda.fit <- lda(formula = Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
data = iris.training)
lda.testdata <- predict(object = lda.fit, newdata = iris.test)
lda.pred <- lda.testdata$class

mean(lda.pred == iris.test$Species)

and similar for qda. For k-NN,

library(class)
iris.training.KNN <- as.matrix(iris.training[,1:4])
iris.test.KNN <- as.matrix(iris.test[,1:4])
knn.pred <- knn(train = iris.training.KNN, test = iris.test.KNN,
               cl = as.matrix(iris.training["Species"]), k = 1)

mean(knn.pred == iris.test$Species)