LDA/QDA/KNN

Peilin Chen

R Markdown

This is an R Markdown document. This is an example of using LDA, QDA and KNN for data classfication. It is based on one of my homework for statistical learning, written by Robert Tibshirani, springer press. http://rmarkdown.rstudio.com.

We use auto as an example dataset for applying LDA

```
require(ISLR)
require(MASS)
require(class)
attach(Auto)
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

Category the mpg to mpg consuming (1) when the value of mpg is greater than the median mpg, and mpg saving (0)

```
mpg0<-ifelse(Auto$mpg>median(Auto$mpg),1,0)
mydat<-data.frame(Auto,mpg0)
head(mydat)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name mpg0
## 1 chevrolet chevelle malibu    0
```

```
## 2         buick skylark 320    0
## 3        plymouth satellite    0
## 4             amc rebel sst    0
## 5               ford torino    0
## 6          ford galaxie 500    0
```

separate the dataset to training and testing If the year is even, it is assigned to be training set, if it is odd, it is assigned to be test set.

```
train<-(year%%2==0)
train.auto<-Auto[train,]
test.auto<-Auto[!train,]
mpg0.train<-mpg0[train]
mpg0.test<-mpg0[!train]
```

Applying LDA and get the confusion table for the test dataset

```
lda.auto<-lda(mpg0.train~cylinders+displacement+horsepower+weight,data = train.auto)
lda.pred<-predict(lda.auto,test.auto)
lda.class<-lda.pred$class
Table<-table(lda.class,mpg0.test)
Table
```

```
##          mpg0.test
## lda.class  0  1
##         0 86  9
##         1 14 73
```

The overall test error of the model is 12.64%.

```
type1<-Table[2]/sum(Table[,1])
#type 1 error is false positive, i.e. assume it is mpg saving(0) but turns out to be mpg consuming(1)
type2<-Table[3]/sum(Table[,2])
overall<-mean(lda.class!=mpg0.test)
rbind(type1,type2,overall)
```

```
##              [,1]
## type1   0.1400000
## type2   0.1097561
## overall 0.1263736
```

Applying QDA for classification and get the confusion table in the test dataset. The overall test error is 13.18%

```
qda.fit<-qda(mpg0.train~cylinders+displacement+horsepower+weight, data = train.auto)
qda.pred<-predict(qda.fit,test.auto)
qda.class<-qda.pred$class
Table<-table(qda.class,mpg0.test)
Table
```

```
##          mpg0.test
## qda.class  0  1
##         0 89 13
##         1 11 69
```

Get the type 1 error (False Positive),type 2 error (False Negative rate) and overall error rate on the test dataset

```r
type1<-Table[2]/sum(Table[,1])
#type 1 error is false positive, i.e. assume it is mpg saving(0) but turns out to be mpg consuming(1)
type2<-Table[3]/sum(Table[,2])
overall<-mean(qda.class!=mpg0.test)
rbind(type1,type2,overall)
```

```
##              [,1]
## type1   0.1100000
## type2   0.1585366
## overall 0.1318681
```

Using Logistic regression for classification. We calculate the confusion table in the test dataset, and get the Type 1/2 error, overall error rate.

```r
logit.fit<-glm(mpg0.train~cylinders+displacement+horsepower+weight, data = train.auto,family =binomial)
summary(logit.fit)
```

```
##
## Call:
## glm(formula = mpg0.train ~ cylinders + displacement + horsepower +
##     weight, family = binomial, data = train.auto)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.48027  -0.03413   0.10583   0.29634   2.57584
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  17.658730   3.409012   5.180 2.22e-07 ***
## cylinders    -1.028032   0.653607  -1.573   0.1158
## displacement  0.002462   0.015030   0.164   0.8699
## horsepower   -0.050611   0.025209  -2.008   0.0447 *
## weight       -0.002922   0.001137  -2.569   0.0102 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 289.58  on 209  degrees of freedom
## Residual deviance:  83.24  on 205  degrees of freedom
## AIC: 93.24
##
## Number of Fisher Scoring iterations: 7
```

```r
prob<-predict(logit.fit,test.auto, type = "response")
logit.pred<-rep("0",length(mpg0.test))
logit.pred[prob>0.5]<-"1"
Table<-table(logit.pred, mpg0.test)
Table
```

```
##           mpg0.test
## logit.pred  0  1
##          0 89 11
##          1 11 71
```

```r
mean(logit.pred!=mpg0.test)
```

```
## [1] 0.1208791
```

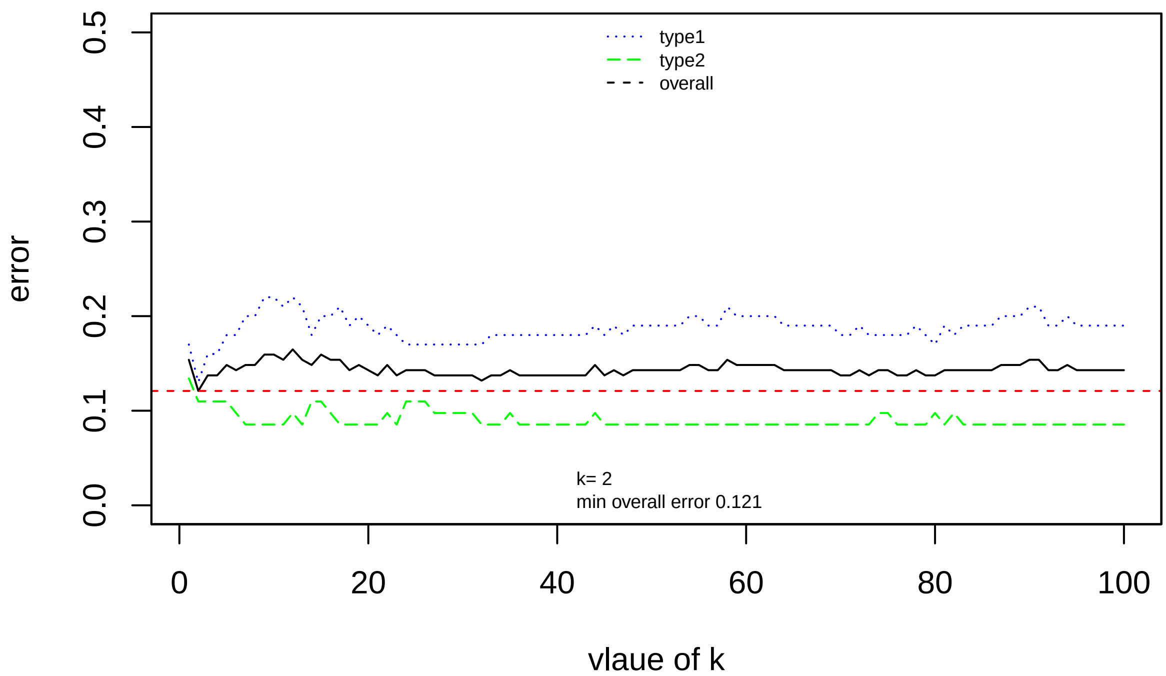
```r
type1<-Table[2]/sum(Table[,1])
type2<-Table[3]/sum(Table[,2])
overall<-mean(logit.pred!=mpg0.test)
rbind(type1,type2,overall)
```

```
##              [,1]
## type1   0.1100000
## type2   0.1341463
## overall 0.1208791
```

Knn method, we need to tune the number of neighbors borrowed for calculation. To tune the number of neighbors (n), write a function knnfun() with n as input

```r
set.seed(010)
knnfun<-function(n){
  type1<-type2<-r<-NULL
  for (i in 1:n){
    knn.fit<-knn(train.auto[,2:5],test.auto[,2:5], mpg0.train,k=i)
    Table<-table(knn.fit, mpg0.test)
    r[i]<-mean(knn.fit!=mpg0.test)
    type1[i]<-Table[2]/sum(Table[,1])
    type2[i]<-Table[3]/sum(Table[,2])
  }
  plot(1:n,r, xlim = c(1,n),ylim = c(0,0.5),
      xlab = "vlaue of k", ylab="error", type = "l")
  abline(h=min(r), col= "red", lty=2)
  legend("bottom",paste(c("k=","min overall error"),c(which(r==min(r)),round(min(r),3))),
        bty = "n", cex=0.6)
  par(new=TRUE)
  plot(1:n,type1,type = "l",ylim=c(0,0.5),xaxt="n",yaxt="n",xlab = "",
      ylab="", col="blue",lty=3)
  par(new=TRUE)
  plot(1:n,type2,type="l",ylim = c(0,0.5), xaxt="n",yaxt="n",xlab = "", ylab="",
      col="green", lty=5)
  legend("top",c("type1","type2", "overall"),
        lty = c(3,5,2),col=c("blue","green","black"),
```

```
          bty = "n", cex=0.6)
  return (cbind(type1,type2,r))
}
knnfun(100)
```




```
##         type1     type2         r
##    [1,]  0.17 0.13414634 0.1538462
##    [2,]  0.13 0.10975610 0.1208791
##    [3,]  0.16 0.10975610 0.1373626
##    [4,]  0.16 0.10975610 0.1373626
##    [5,]  0.18 0.10975610 0.1483516
##    [6,]  0.18 0.09756098 0.1428571
##    [7,]  0.20 0.08536585 0.1483516
##    [8,]  0.20 0.08536585 0.1483516
##    [9,]  0.22 0.08536585 0.1593407
##   [10,]  0.22 0.08536585 0.1593407
##   [11,]  0.21 0.08536585 0.1538462
##   [12,]  0.22 0.09756098 0.1648352
##   [13,]  0.21 0.08536585 0.1538462
##   [14,]  0.18 0.10975610 0.1483516
##   [15,]  0.20 0.10975610 0.1593407
##   [16,]  0.20 0.09756098 0.1538462
##   [17,]  0.21 0.08536585 0.1538462
##   [18,]  0.19 0.08536585 0.1428571
##   [19,]  0.20 0.08536585 0.1483516
##   [20,]  0.19 0.08536585 0.1428571
##   [21,]  0.18 0.08536585 0.1373626
##   [22,]  0.19 0.09756098 0.1483516
##   [23,]  0.18 0.08536585 0.1373626
##   [24,]  0.17 0.10975610 0.1428571
##   [25,]  0.17 0.10975610 0.1428571
```

```
## [26,] 0.17 0.10975610 0.1428571
## [27,] 0.17 0.09756098 0.1373626
## [28,] 0.17 0.09756098 0.1373626
## [29,] 0.17 0.09756098 0.1373626
## [30,] 0.17 0.09756098 0.1373626
## [31,] 0.17 0.09756098 0.1373626
## [32,] 0.17 0.08536585 0.1318681
## [33,] 0.18 0.08536585 0.1373626
## [34,] 0.18 0.08536585 0.1373626
## [35,] 0.18 0.09756098 0.1428571
## [36,] 0.18 0.08536585 0.1373626
## [37,] 0.18 0.08536585 0.1373626
## [38,] 0.18 0.08536585 0.1373626
## [39,] 0.18 0.08536585 0.1373626
## [40,] 0.18 0.08536585 0.1373626
## [41,] 0.18 0.08536585 0.1373626
## [42,] 0.18 0.08536585 0.1373626
## [43,] 0.18 0.08536585 0.1373626
## [44,] 0.19 0.09756098 0.1483516
## [45,] 0.18 0.08536585 0.1373626
## [46,] 0.19 0.08536585 0.1428571
## [47,] 0.18 0.08536585 0.1373626
## [48,] 0.19 0.08536585 0.1428571
## [49,] 0.19 0.08536585 0.1428571
## [50,] 0.19 0.08536585 0.1428571
## [51,] 0.19 0.08536585 0.1428571
## [52,] 0.19 0.08536585 0.1428571
## [53,] 0.19 0.08536585 0.1428571
## [54,] 0.20 0.08536585 0.1483516
## [55,] 0.20 0.08536585 0.1483516
## [56,] 0.19 0.08536585 0.1428571
## [57,] 0.19 0.08536585 0.1428571
## [58,] 0.21 0.08536585 0.1538462
## [59,] 0.20 0.08536585 0.1483516
## [60,] 0.20 0.08536585 0.1483516
## [61,] 0.20 0.08536585 0.1483516
## [62,] 0.20 0.08536585 0.1483516
## [63,] 0.20 0.08536585 0.1483516
## [64,] 0.19 0.08536585 0.1428571
## [65,] 0.19 0.08536585 0.1428571
## [66,] 0.19 0.08536585 0.1428571
## [67,] 0.19 0.08536585 0.1428571
## [68,] 0.19 0.08536585 0.1428571
## [69,] 0.19 0.08536585 0.1428571
## [70,] 0.18 0.08536585 0.1373626
## [71,] 0.18 0.08536585 0.1373626
## [72,] 0.19 0.08536585 0.1428571
## [73,] 0.18 0.08536585 0.1373626
## [74,] 0.18 0.09756098 0.1428571
## [75,] 0.18 0.09756098 0.1428571
## [76,] 0.18 0.08536585 0.1373626
## [77,] 0.18 0.08536585 0.1373626
## [78,] 0.19 0.08536585 0.1428571
## [79,] 0.18 0.08536585 0.1373626
```

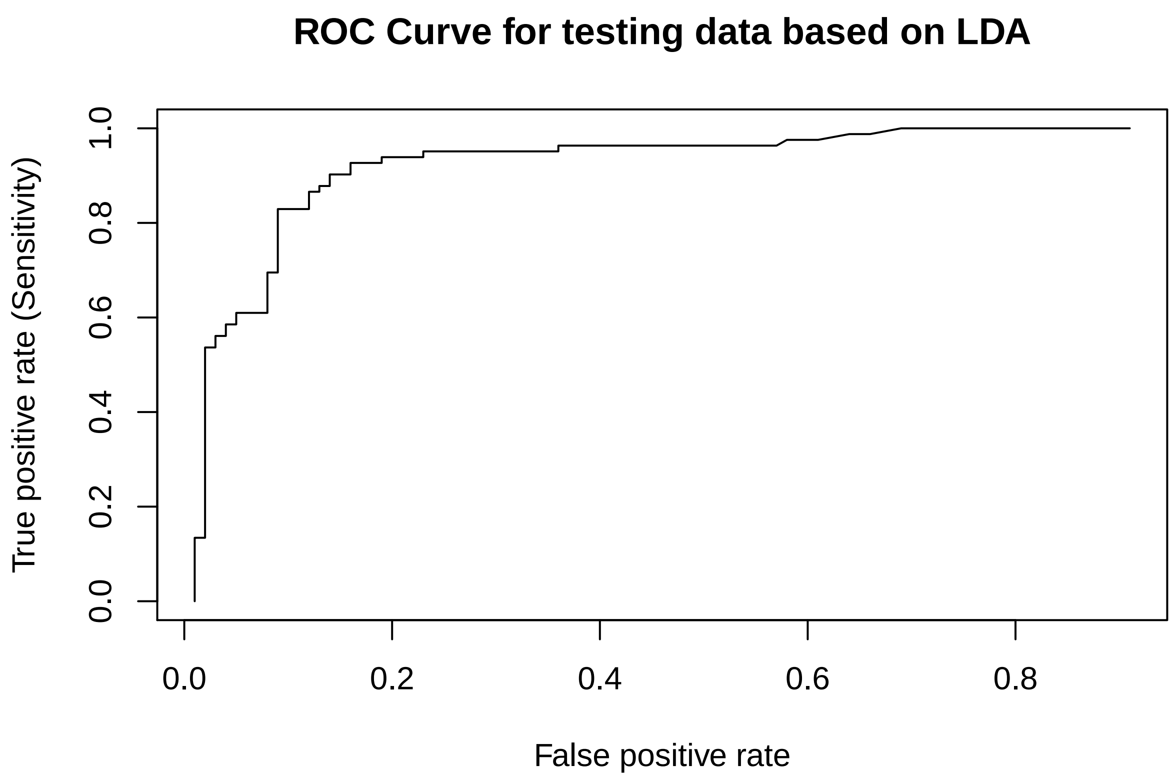
```
## [80,] 0.17 0.09756098 0.1373626
## [81,] 0.19 0.08536585 0.1428571
## [82,] 0.18 0.09756098 0.1428571
## [83,] 0.19 0.08536585 0.1428571
## [84,] 0.19 0.08536585 0.1428571
## [85,] 0.19 0.08536585 0.1428571
## [86,] 0.19 0.08536585 0.1428571
## [87,] 0.20 0.08536585 0.1483516
## [88,] 0.20 0.08536585 0.1483516
## [89,] 0.20 0.08536585 0.1483516
## [90,] 0.21 0.08536585 0.1538462
## [91,] 0.21 0.08536585 0.1538462
## [92,] 0.19 0.08536585 0.1428571
## [93,] 0.19 0.08536585 0.1428571
## [94,] 0.20 0.08536585 0.1483516
## [95,] 0.19 0.08536585 0.1428571
## [96,] 0.19 0.08536585 0.1428571
## [97,] 0.19 0.08536585 0.1428571
## [98,] 0.19 0.08536585 0.1428571
## [99,] 0.19 0.08536585 0.1428571
## [100,] 0.19 0.08536585 0.1428571
```

If we define the False Positive as $P(mpg = 1|mpg = 0)$ and False Negative as $P(mpg = 0|mpg = 1)$. In this case, if we consider False Positive as more severe, and we want to specifically control type 1 error (False Positive rate). Then the ROC curve might help us compare the performance of the classification methods. A good method should have small False Positive rate and high True Positive rate, which leads to the curve approach the left upper corner

```r
#Roc curve for lda
ROC_curv2<-function(n){
# changing the cut off of classification
lda.auto <- lda(mpg0.train~cylinders+displacement+horsepower+weight,data = train.auto)
lda.pred<-predict(lda.auto,test.auto)
pr_mpg1<-lda.pred$posterior[,1]
#lda.pred posterior probablity [,1] is
#the probability that belongs to mpg=1
truepos <- numeric(n)
falsepos <- numeric(n)
cls<-numeric(length(mpg0.test))
p1 <- (1:n)/(n+1)
for (i in 1:n){
  p <- p1[i]
  cls<-ifelse(pr_mpg1>p,1,0)
  Table<-table(cls,mpg0.test)
  falsepos[i]<-Table[3]/sum(Table[,2])
#type 1 error is false positive, i.e. assume it is mpg saving(0) but turns out to be mpg consuming(1)
  truepos[i]<-1-Table[2]/sum(Table[,1])
}

plot(falsepos ~ truepos, type = "l",
     xlab = "False positive rate",
     ylab = "True positive rate (Sensitivity)",
     main="ROC Curve for testing data based on LDA", cex=0.8)
return(cbind(falsepos,truepos))
```

```
}
result<-ROC_curv2(10000)
```




```
#result[,1]
pvalue1<-(which(result[,2]<0.1))/10000
range(pvalue1)
```

```
## [1] 0.0007 0.0530
```

```
pvalue2<-(which(result[,1]>0.8))/10000
range(pvalue2)
```

```
## [1] 0.0306 1.0000
```

similar results can be get for QDA and logistic regression