Advances Computational Econometrics. Chapter 2: Linear and quadratic classification

Thierry Denoeux

3/29/2022

Exercise 1

Question 1

Loading the data:

```r
credit<-read.csv('/Users/Thierry/Documents/R/Data/Economics/default_credit_card.csv',
                 sep=";",header=TRUE)
```

Redefining variables as factors:

```r
credit$X2<-as.factor(credit$X2)
levels(credit$X2)<-c("M","F")
credit$X3<-as.factor(credit$X3)
credit$X4<-as.factor(credit$X3)
credit$Y<-as.factor(credit$Y)
```

Splitting the data randomly between a training set and a test set:

```r
set.seed(29032022)
n<-nrow(credit)
ntrain<-20000
ntest<-n-ntrain
train<-sample(n,ntrain)
credit.train<-credit[train,]
credit.test<-credit[-train,]
```

Question 2

We first fit the LDA model on the training data, and predict the response for the test data (we exclude the qualitative variables X_2, X_3 and X_4, which cannot be handled by LDA):

```r
library(MASS)
fit.lda<-lda(Y~.,data=credit.train[,-c(2,3,4)])
pred.lda<-predict(fit.lda,newdata=credit.test[,-c(2,3,4)])
```

We compute the confusion matrix and the error rate:

```r
perf.lda<-table(pred.lda$class,credit.test$Y)
print(perf.lda)
```

```
##
##         0    1
##    0 7565 1687
##    1  223  525
```

```
err.lda<-1-sum(diag(perf.lda))/ntest
print(err.lda)
```

```
## [1] 0.191
```

We perform the same operations with QDA:

```
fit.qda<-qda(Y~.,data=credit.train[,-c(2,3,4)])
pred.qda<-predict(fit.qda,newdata=credit.test[,-c(2,3,4)])
perf.qda<-table(pred.qda$class,credit.test$Y)
print(perf.qda)
```

```
##
##         0    1
##    0 3152  389
##    1 4636 1823
```

```
err.qda<-1-sum(diag(perf.qda))/ntest
print(err.qda)
```

```
## [1] 0.5025
```

For naive Bayes, we use function `naive_bayes` from package `naivebayes`. This time, we can put the qualitative variables in the model:

```
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```
fit.naive<-naive_bayes(Y~.,data=credit.train)
pred.naive<-predict(fit.naive,newdata=credit.test)
perf.naive<-table(pred.naive,credit.test$Y)
print(perf.naive)
```

```
##
## pred.naive    0    1
##          0 5587  820
##          1 2201 1392
```

```
err.naive<-1-sum(diag(perf.naive))/ntest
print(err.naive)
```

```
## [1] 0.3021
```

Finally, we apply logistic regression:

```
fit.logreg<- glm(Y~.,data=credit.train,family=binomial)
pred.logreg<-predict(fit.logreg,newdata=credit.test,type='response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
perf.logreg <-table(credit.test$Y,pred.logreg>0.5)
print(perf.logreg)
```

```
##
```

```
##      FALSE TRUE
##    0  7593  195
##    1  1698  514
```

```
err.logreg <-1-sum(diag(perf.logreg))/ntest
print(err.logreg)
```

```
## [1] 0.1893
```

Comparison of error rates:

```
print(c(err.lda,err.qda,err.naive,err.logreg))
```

```
## [1] 0.1910 0.5025 0.3021 0.1893
```

The linear classifiers perform better on this data set.

Question 3

To generate the ROC curve of the LDA classifier, we need to provide the discriminant variable `pred.lda$x` to function `roc` of package `pROC`:

```
library(pROC)
roc_lda<-roc(credit.test$Y,as.vector(pred.lda$x))
```

We do the same for QDA, and plot the ROC curve on the same graph as LDA:

```
roc_qda<-roc(credit.test$Y,as.vector(pred.qda$posterior[,1]))
```

We generate the ROC curves of the naive Bayes classifier. Again, we need to provide a discriminant variable; here, we provide the posterior probability of class 1:

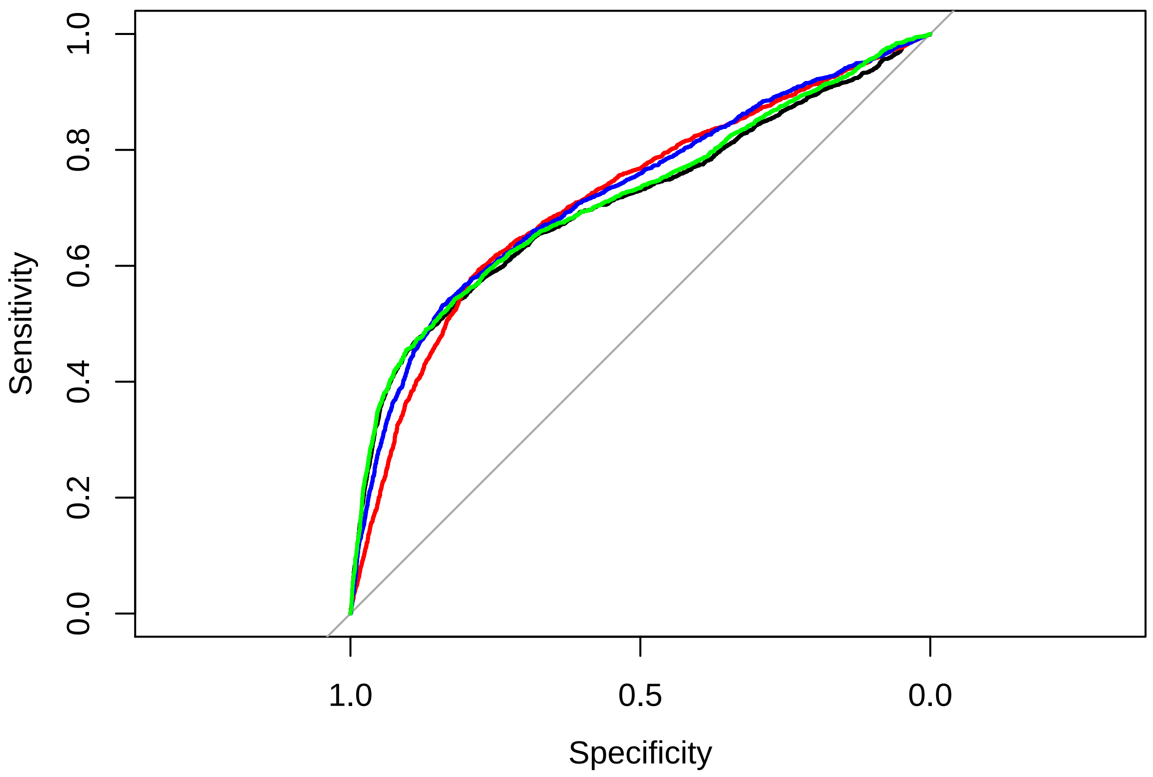
```
pred.nb.prob<-predict(fit.naive,newdata=credit.test,type="prob")
roc_nb<-roc(credit.test$Y,as.vector(pred.nb.prob[,1]))
```

We generate the ROC curve of the logistic regression classifier:

```
roc_logreg<-roc(credit.test$Y,as.vector(pred.logreg))
```

Finally, we plot the four curves on the same graph:

```
plot(roc_lda)
plot(roc_qda,add=TRUE,col='red')
plot(roc_nb,add=TRUE,col='blue')
plot(roc_logreg,add=TRUE,col='green')
```



Exercise 2

Question 1

Loading package `mvtnorm` and setting the parameters:

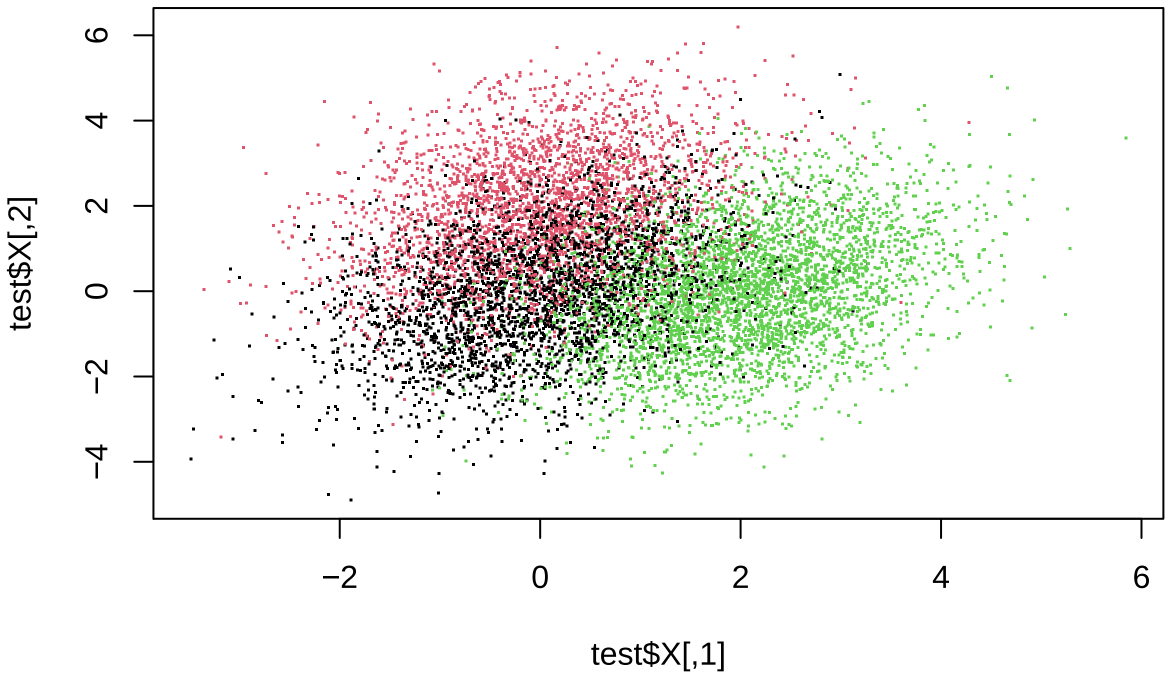
```r
library(mvtnorm)
mu1<-c(0,0)
mu2<-c(0,2)
mu3<-c(2,0)
Sigma<-matrix(c(1,0.5,0.5,2),2,2)
p1<-0.3
p2<-0.3
p3<-1-p1-p2
```

We start by writing a function that generates a sample of size n:

```r
gen.data<- function(n,mu1,mu2,mu3,Sigma1,Sigma2,Sigma3,p1,p2){
  y<-sample(3,n,prob=c(p1,p2,1-p1-p2),replace=TRUE)
  X<-matrix(0,n,2)
  N1<-length(which(y==1))
  N2<-length(which(y==2))
  N3<-length(which(y==3))
  X[y==1,]<-rmvnorm(N1,mu1,Sigma1)
  X[y==2,]<-rmvnorm(N2,mu2,Sigma2)
  X[y==3,]<-rmvnorm(N3,mu3,Sigma3)
  return(list(X=X,y=y))
}
```

We generate a test set of size 10,000 and plot the data:

```
test<-gen.data(n=10000,mu1,mu2,mu3,Sigma,Sigma,Sigma,p1=p1,p2=p2)
plot(test$X,col=test$y,pch=".")
```



We now create a matrix g containing the discriminant functions $p_k(x)\pi_k$, and the class predictions by the Bayes rule:

```
g<-cbind(p1*dmvnorm(test$X,mu1,Sigma),
         p2*dmvnorm(test$X,mu2,Sigma),
         p3*dmvnorm(test$X,mu3,Sigma))
ypred<-max.col(g)
```

Finally, we estimate the Bayes error rate as:

```
errb<-mean(test$y != ypred)
print(errb)
```

```
## [1] 0.235
```

Question 2

Let us set the training set size to $n = 50$. We are going to generate 100 training sets, fit LDA and QDE on these training sets, and compute the corresponding test error rates.

Initialization:

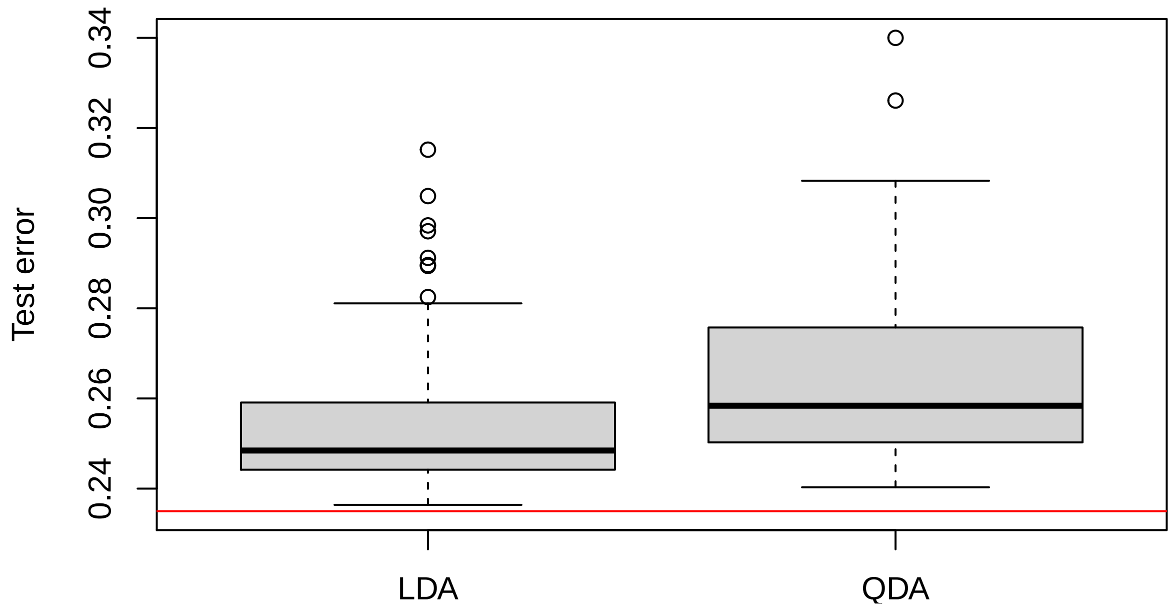
```
N<- 100 # Number of replications
err.lda<-rep(0,N)
err.qda<-rep(0,N)
n<-50 # Training set size
test.frame<-data.frame(test)
```

Main loop:

```
for(i in 1:100){
  # training set generation
  train<-gen.data(n,mu1,mu2,mu3,Sigma,Sigma,Sigma,p1=p1,p2=p2)
  train.frame<-data.frame(train)
  # LDA
  fit.lda<- lda(y~.,data=train.frame)
  pred.lda<-predict(fit.lda,newdata=test.frame)
  err.lda[i]<-mean(test$y != pred.lda$class)
  # QDA
  fit.qda<- qda(y~.,data=train.frame)
  pred.qda<-predict(fit.qda,newdata=test.frame)
  err.qda[i]<-mean(test$y != pred.qda$class)
}
```

Plotting the results:

```
boxplot(cbind(err.lda,err.qda),ylim=range(errb,err.lda,err.qda),ylab="Test error",
        names=c("LDA","QDA"))
abline(h=errb,col="red")
```



Exercise 3

Question 1

We define a new ordered variable medv_ord, and we create a new data frame Boston1 with that variable in place of medv:

```
attach(Boston)
q<-quantile(medv,c(1/3,2/3))
medv_ord<-(medv>q[1]) + (medv>q[2])
medv_ord<-ordered(medv_ord,levels=c(0,1,2),labels=c("low","medium","high"))
Boston1<-cbind(Boston[,1:13],medv_ord)
```

Question 2

We partition the data frame `Boston1` into training and test sets:

```r
set.seed(220322)
n<-nrow(Boston1)
ntrain<-round(2*n/3)
ntest<-n-ntrain
train<-sample(n,ntrain)
Boston1.train<-Boston1[train,]
Boston1.test<-Boston1[-train,]
```

We fit logit and probit ordered regression models:

```r
fit_ord1<-polr(medv_ord~.,data=Boston1.train,method = "logistic")
fit_ord2<-polr(medv_ord~.,data=Boston1.train,method = "probit")
```

The coefficients of the two models are similar:

```r
summary(fit_ord1)
```

```
##
## Re-fitting to get Hessian

## Call:
## polr(formula = medv_ord ~ ., data = Boston1.train, method = "logistic")
##
## Coefficients:
##             Value Std. Error   t value
## crim    -0.226721   0.084774   -2.6744
## zn       0.013492   0.010811    1.2481
## indus    0.011626   0.038336    0.3033
## chas     0.575741   0.649173    0.8869
## nox     -7.929831   0.024676 -321.3626
## rm       1.694579   0.278254    6.0900
## age     -0.012878   0.009049   -1.4231
## dis     -0.627235   0.133470   -4.6995
## rad      0.236795   0.055947    4.2325
## tax     -0.008073   0.002827   -2.8556
## ptratio -0.388405   0.078631   -4.9396
## black    0.004490   0.002352    1.9091
## lstat   -0.322171   0.047655   -6.7605
##
## Intercepts:
##             Value     Std. Error t value
## low|medium    -9.6826    0.0259  -374.3850
## medium|high   -5.5887    0.3578   -15.6202
##
## Residual Deviance: 329.7747
## AIC: 359.7747
```

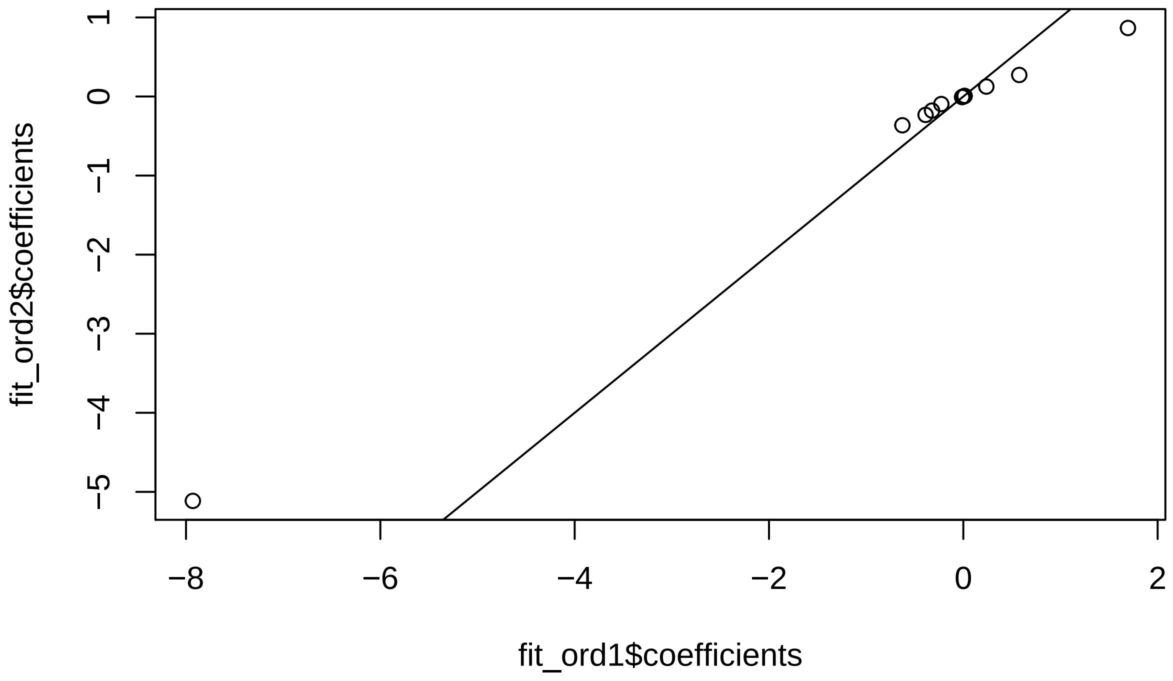
```r
summary(fit_ord2)
```

```
##
## Re-fitting to get Hessian

## Call:
## polr(formula = medv_ord ~ ., data = Boston1.train, method = "probit")
```

```
##
## Coefficients:
##              Value Std. Error   t value
## crim    -0.095469   0.043615   -2.1889
## zn       0.008178   0.006217    1.3155
## indus    0.004852   0.021467    0.2260
## chas     0.272292   0.347466    0.7837
## nox     -5.114376   0.007544 -677.9553
## rm       0.866344   0.144040    6.0146
## age     -0.006900   0.005088   -1.3563
## dis     -0.363878   0.073241   -4.9682
## rad      0.125360   0.030103    4.1643
## tax     -0.004536   0.001533   -2.9594
## ptratio -0.232778   0.043748   -5.3209
## black    0.002033   0.001193    1.7046
## lstat   -0.178332   0.024711   -7.2166
##
## Intercepts:
##              Value     Std. Error t value
## low|medium    -6.8054    0.0121  -564.0200
## medium|high   -4.5257    0.1845   -24.5302
##
## Residual Deviance: 331.8913
## AIC: 361.8913
```

```r
plot(fit_ord1$coefficients,fit_ord2$coefficients)
abline(0,1)
```



Finally, we evaluate the two models on the test set:

```r
pred1<-predict(fit_ord1,Boston1.test)
conf1<-table(Boston1.test$medv_ord,pred1)
print(conf1)
```

```
##         pred1
##          low medium high
##   low     49      7    0
##   medium  12     36    7
##   high     3     10   45
```

```
print(1-sum(diag(conf1))/ntest)
```

```
## [1] 0.2307692
```

```
pred2<-predict(fit_ord2,Boston1.test)
conf2<-table(Boston1.test$medv_ord,pred2)
print(conf2)
```

```
##         pred2
##          low medium high
##   low     50      6    0
##   medium  13     34    8
##   high     3     11   44
```

```
print(1-sum(diag(conf2))/ntest)
```

```
## [1] 0.2426036
```

McNemar test:

```
pred1<-ordered(pred1)
correct.logit<-(pred1==Boston1.test$medv_ord)
pred2<-ordered(pred2)
correct.probit<-(pred2==Boston1.test$medv_ord)
mcnemar.test(correct.probit,correct.logit)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  correct.probit and correct.logit
## McNemar's chi-squared = 0.25, df = 1, p-value = 0.6171
```

The error rates are not significantly different.

Question 3

For multinomial logistic regression, we use function `multinom` of package `nnet`:

```
library(nnet)
fit<-multinom(medv_ord~.,data=Boston1.train)
```

```
## # weights:  45 (28 variable)
## initial  value 370.232341
## iter  10 value 252.255081
## iter  20 value 204.981662
## iter  30 value 149.363145
## iter  40 value 145.665559
## iter  50 value 145.060525
## final  value 145.033666
## converged
```

```
pred3<-predict(fit,Boston1.test)
conf3<-table(Boston1.test$medv_ord,pred3)
print(conf3)
```

```
##         pred3
##          low medium high
##   low     50      6    0
##   medium  13     33    9
##   high     3     14   41
```

```
print(1-sum(diag(conf3))/ntest)
```

```
## [1] 0.2662722
```

The error rate is slightly higher than those of logit and probit ordered regression, but the difference is not significant, as shown by the McNemar test applied to the error rates of the ordinal logit and multinomial logistic regression:

```
pred3<-ordered(pred3, levels=c("low","medium","high"))
correct.multinom<-(pred3==Boston1.test$medv_ord)
mcnemar.test(correct.logit,correct.multinom)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  correct.logit and correct.multinom
## McNemar's chi-squared = 2.5, df = 1, p-value = 0.1138
```