Corrigé fiche TP2

Vincent Vandewalle, Cristian Preda

Décembre 2019

1. Evaluation de la règle de classement (iris de Fisher)

On importe les résultats du TP précédent :

```
library("knitr")
purl("corrigeTP1.Rmd")
```

```
##
##
## processing file: corrigeTP1.Rmd
```

```
## output file: corrigeTP1.R
```

```
source("corrigeTP1.R")
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## Loading required package: magrittr
```

1. Diagnostic apparent

(a) Matrice de confusion

La matrice de confusion s'obtient en croisant la vraie valeur de Y avec sa valeur prédite

```
table(Yreel = iris$Y,Ypredit = Ypredit)
```

```
##             Ypredit
## Yreel        setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

(b) TBC: taux de bon classement et TMC (mauvais)

```
TBC=mean(iris$Y==Ypredit)
TBC
```

```
## [1] 0.98
```

```
TMC=1-TBC
```

(c) Le biais d'optimisme est dû au fait que les mêmes données ont à la fois étées utilisées pour apprendre et pour tester le modèle.

2. Diagnostic sur échantillon test

(a) Séparation entre un échantillon d'apprentissage et un échantillon test

```
n = nrow(iris)
set.seed(1234)
idx=sample(n,round(n*0.7),replace=FALSE)
```

```
iristrain=iris[idx,]
iristest=iris[-idx,]
```

(b) Apprentissage du score sur les données d'apprentissage et test sur l'échantillon test

Fonctions de l'énoncé

```
calcalpha <-function(X, Y){
  d=ncol(X)
  k=nlevels(Y)
  W=matrix(0,d,d)
  ni=table(Y)
  for (i in levels(Y)){
    W=W+cov.wt(X[Y==i,],method="ML")$cov*ni[i]
  }
  W=W/sum(ni)
  moyennes=by(X,Y,colMeans)
  G=matrix(unlist(moyennes),k,d,byrow=T)
  B=cov.wt(G,method="ML")$cov
  alpha=matrix(0,(d+1),k)
  rownames(alpha) = c("intercept",colnames(X))
  colnames(alpha) = levels(Y)
  for (i in 1:k) {
    Xi=matrix(G[i,],d,1)
    alpha[1,i]=-t(Xi)%*%solve(W)%*%Xi
    alpha[2:(d+1),i]=2*solve(W)%*%Xi
  }
  return (alpha)
}

predictY<- function(X,alpha){
  s=as.matrix(cbind(1,X))%*%alpha
  Ypredit=colnames(alpha)[apply(s,1,which.max)]
}
```

Apprentissage

```
alpha = calcalpha(iristrain[,1:4],iristrain[,5])
alpha
```

```
##               setosa versicolor   virginica
## intercept -170.62852 -148.66321 -224.818762
## X1          51.64174   30.53514   20.661754
## X2          43.44095   13.84094    9.465784
## X3         -38.29711   11.75900   33.290958
## X4         -30.75787   20.88051   46.962893
```

Classement sur l'échantillon test

```
Ypredit = predictY(iristest[,1:4],alpha)
Ypredit[1:10]
```

```
##  [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
##  [9] "setosa" "setosa"
```

(c) Evaluation de la règle de classement

```r
table(Y = iristest[,5], Ypredit)
```

```
##             Ypredit
## Y            setosa versicolor virginica
##   setosa         16          0         0
##   versicolor      0         16         0
##   virginica       0          0        13
```

```r
TBC=mean(iristest[,5]==Ypredit)
TBC
```

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

```r
TMC=1-TBC
```

3. Validation croisée leave-one-out

Ici le faible nombre de données ne permet pas d'évaluer de manière fiable les performances du modèle car l'effectif de l'échantillon test est trop petit.

Pour de petits effectifs on préconise plutôt d'utiliser la validation croisée. Ici leave-one-out.

```r
Ypredit = rep("",nrow(iris))
for (i in 1:n){
  alpha = calcalpha(iris[-i,1:4],iris[-i,5])
  Ypredit[i] = predictY(iris[i,1:4],alpha)
}
table(Y = iris[,5], Ypredit)
```

```
##             Ypredit
## Y            setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

```r
#sapply(1:n,
#       function(i) predictY(iris[i,1:4],calcalpha(iris[-i,1:4],iris[-i,5])))
TBC = mean(iris$Y == Ypredit)
TBC
```

```
## [1] 0.98
```

```r
1-TBC
```

```
## [1] 0.02
```

2. Analyse discriminante linéaire (iris de Fisher)

Reprise du jeu de données *iris*

```r
data(iris)
names(iris) <- c(paste0("X",1:4),"Y")
```

1. Charger le package *MASS*, puis utiliser la fonction *lda* qui permet d'ajuster le modèle d'analyse discriminante linéaire. En utilisant les fonctions *predict* et *table*, réaliser la matrice de confusion. On précise l'option *prior = prop.table(rep(1,nlevels(Y)))* (proportions égales dans chacune des classes) pour retrouver les résultats de l'AFD.

```r
library("MASS")
X<-iris[,1:4]
Y<-iris[,5]
LDAXY <-  lda(X,grouping=Y,prior = prop.table(rep(1,nlevels(Y))))
LDAXY
```

```
## Call:
## lda(X, grouping = Y, prior = prop.table(rep(1, nlevels(Y))))
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##               X1    X2    X3    X4
## setosa     5.006 3.428 1.462 0.246
## versicolor 5.936 2.770 4.260 1.326
## virginica  6.588 2.974 5.552 2.026
##
## Coefficients of linear discriminants:
##           LD1         LD2
## X1  0.8293776  0.02410215
## X2  1.5344731  2.16452123
## X3 -2.2012117 -0.93192121
## X4 -2.8104603  2.83918785
##
## Proportion of trace:
##    LD1    LD2
## 0.9912 0.0088
```

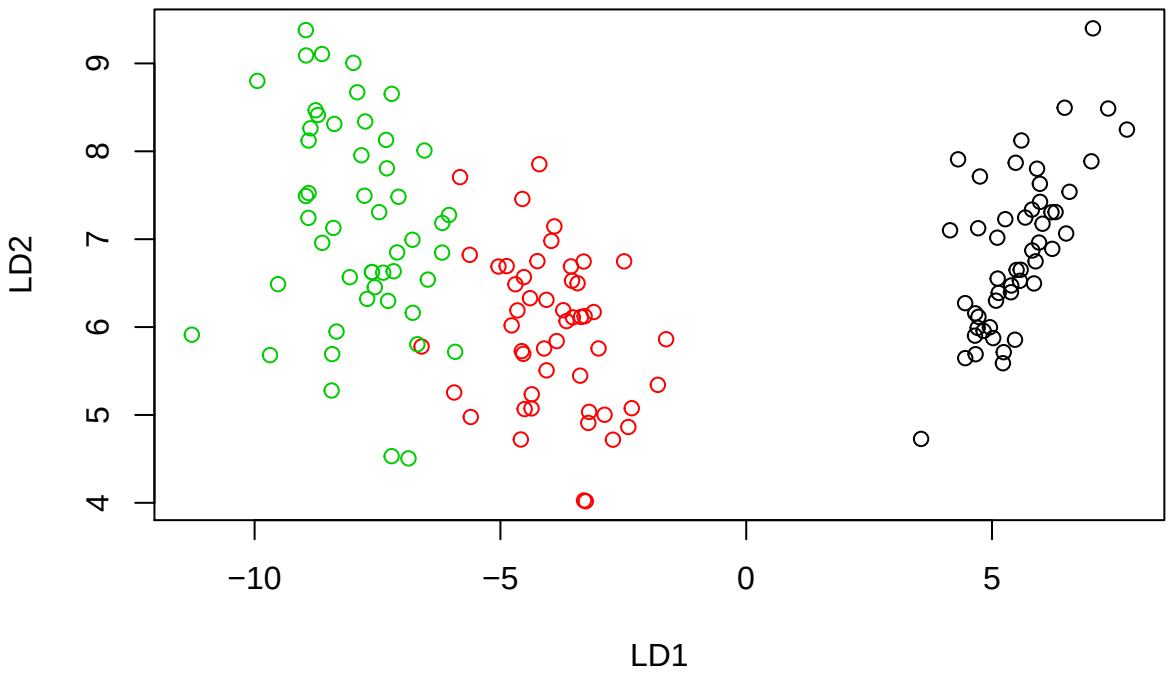
```r
Yp <- predict(LDAXY)
names(Yp)
```

```
## [1] "class"     "posterior" "x"
```

```r
table(Y,Ypredit = Yp$class)
```

```
##             Ypredit
## Y            setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

```r
d = as.matrix(iris[,1:4]) %*% LDAXY$scaling
plot(d, col = iris$Y)
```



2. Evaluer le taux de mauvais classement par validation croisée leave-one-out, en utilisant l'option *CV = TRUE* dans la fonction *lda*. Réaliser aussi la matrice de confusion. On remarque que les classes d'affectation et les probabilités a posteriori sont directement renvoyées dans l'objet de sortie, sans faire appel à *predict*.

```
LDAXYLOO = lda(X,grouping=Y, CV=TRUE,prior = prop.table(rep(1,nlevels(Y))))
str(LDAXYLOO)
```

```
## List of 3
##  $ class    : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ posterior: num [1:150, 1:3] 1 1 1 1 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "setosa" "versicolor" "virginica"
##  $ call     : language lda(x = X, grouping = Y, CV = TRUE, prior = prop.table(rep(1, nlevels(Y))))
```

```
# YpLOO= predict(LDAXYLOO)$class ne fonctionne pas puisque la classe est déjà prédite
dim(LDAXYLOO$posterior)
```

```
## [1] 150   3
```

```
head(LDAXYLOO$posterior)
```

```
##      setosa   versicolor    virginica
## [1,]      1 5.087494e-22 4.385241e-42
## [2,]      1 9.588256e-18 8.888069e-37
## [3,]      1 1.983745e-19 8.606982e-39
## [4,]      1 1.505573e-16 5.101765e-35
```

```
## [5,]      1 2.075670e-22 1.739832e-42
## [6,]      1 5.332271e-21 8.674906e-40
```

```
table(Yreel=Y,LOO=LDAXYLOO$class)
```

```
##             LOO
## Yreel        setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

```
TBC=mean(Y==LDAXYLOO$class)
TBC
```

```
## [1] 0.98
```

```
TMC=1-TBC
TMC
```

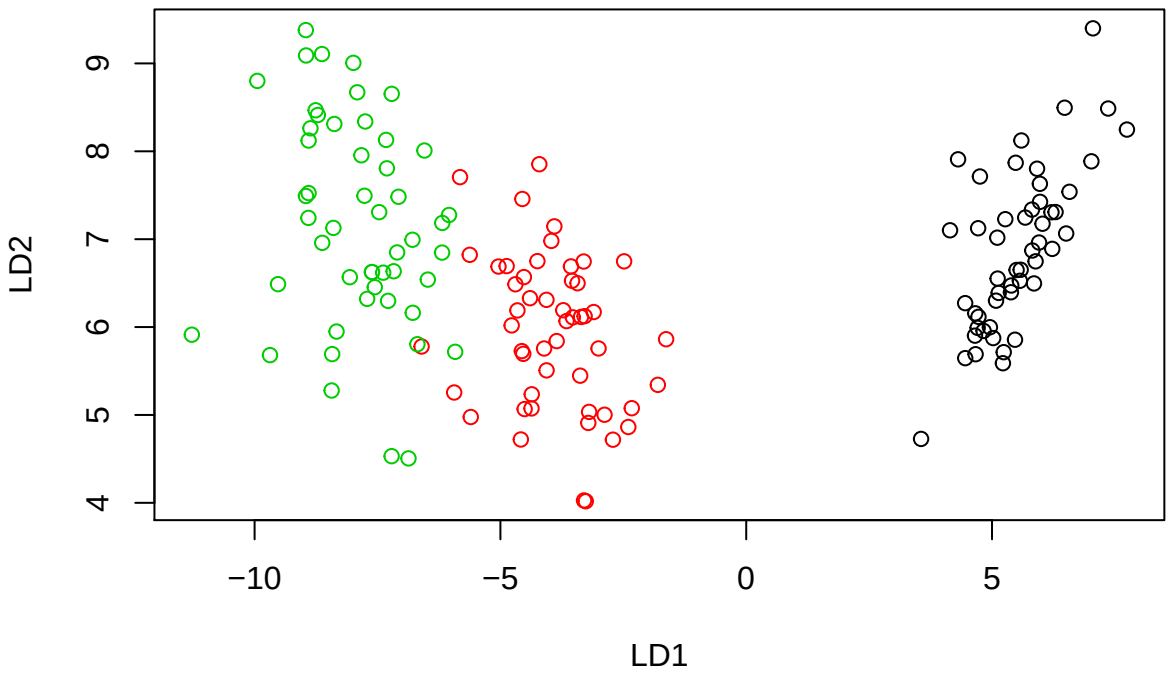
```
## [1] 0.02
```

3. Utiliser à nouveau la fonction *lda* sans préciser l'option *CV=TRUE*. Dans les sorties on remarque que la fonction retourne les coefficients linéaires discriminants *LD1* et *LD2*. Ceux-ci peuvent être récupérés par le champ scaling de l'objet retourné par la fonction *lda* (cette sortie n'est pas disponible dans le cas où on a *CV=TRUE*). En multipliant la matrice de données par la matrice des coefficients linéaires discriminants, obtenir une projection des individus sur ces axes discriminants. Faire le graphique permettant de visualiser ces données.

```
#projection
LDAv=LDAXY$scaling
LDAv
```

```
##           LD1         LD2
## X1  0.8293776  0.02410215
## X2  1.5344731  2.16452123
## X3 -2.2012117 -0.93192121
## X4 -2.8104603  2.83918785
```

```
Projec=as.matrix(X) %*% LDAv
plot(Projec, col=iris$Y)
```



Remarque : ici les coefficients linéaires discriminants sont renormalisés de telle sorte que les variances intra-classe (dans leur version sans biais) sur les axes discriminants soit égale à 1 :

```
# Calcul de la variance intra-classe sur les composantes discriminantes
# en repartant de la variance porjetée
t(LDAXY$scaling[,1,drop=F]) %*% W %*% LDAXY$scaling[,1,drop = F]
```

```
##      LD1
## LD1 0.98
```

```
t(LDAXY$scaling[,2,drop=F]) %*% W %*% LDAXY$scaling[,2,drop = F]
```

```
##      LD2
## LD2 0.98
```

```
# Variance dans chaque classe
Wproj = matrix(0,2,2)
ni = table(Y)
for (i in levels(Y)){
  Wproj = Wproj + ni[i]*cov(Projec[Y==i,])
}
Wproj = Wproj/length(Y)
Wproj
```

```
##               LD1           LD2
## LD1  1.000000e+00 -6.276461e-16
## LD2 -6.276461e-16  1.000000e+00
```

En définitive on remarque aussi que la covariance intra-classe entre les axes discriminants est égale à 0. Ainsi

sur ces axes le classement d'un nouveau point peut se faire par simple minimisation de la distance euclidienne.

```
# Moyennes des classes sur les axes discriminants
centres = t(simplify2array(by(Projec,Y,colMeans)))
centres
```

```
##                  LD1      LD2
## setosa      5.502493 6.876606
## versicolor -3.930156 5.933573
## virginica  -7.887657 7.174239
```

```
distances = matrix(0,nrow(Projec),nlevels(Y),dimnames = list(row.names(Projec),levels(Y)))
# Calcul des distances pour chaque individu projeté au centre des classes
for (i in 1:nrow(Projec)){
  for (j in levels(Y)){
    distances[i,j] = sqrt(sum((Projec[i,] - centres[j,])^2))
  }
}
head(distances)
```

```
##         setosa versicolor virginica
## [1,] 0.4621379   9.940183  13.84598
## [2,] 1.1103815   8.953930  12.97646
## [3,] 0.4947396   9.326353  13.29517
## [4,] 1.1898102   8.638440  12.65122
## [5,] 0.6040845  10.034563  13.91486
## [6,] 1.2501531   9.775382  13.51785
```

```
# Classement obtenu :
classement = apply(distances, 1, function(x) names(which.min(x)))
head(classement)
```

```
## [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
```

```
table(Y,classement)
```

```
##             classement
## Y            setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

Le classement obtenu est le même qu'en utilisant le score linéaire.