Practicals 3 : Supervised Learning

1 Framework

During the lecture, we have introduced two methods to handle the problem of supervised learning. The first one, the Multiple Discriminant Analysis (MDA), is based on an adapted PCA and the other one, the Classification And Regression Tree (CART), is a predictive method built from some binary decision tree.

To get the data and some useful graphical functions for the sequel of these practicals, we start by loading the script *tp3.R*,

```
source("http://www.math.univ-toulouse.fr/~xgendre/ens/m2se/tp3.R")
```

We will also use the package `rpart`. If this package is not installed, you can do it with the command `install.packages("rpart", dependencies=TRUE)`.

2 Lubischew's insects (MDA)

The example used during the lecture to illustrate the MDA is related to some dataset based on physical measures of insects. We have at our disposal $p = 6$ measures taken on $n = 74$ insects distributed among three species numbered from 1 to 3 in the sequel. To obtain this dataset,

```
X <- DataLubischew()
```

As usual, each line of the data matrix `X` contains the measures relative to one individual. Moreover, we also have an extra column for the label of each individual (*i.e.* the species of the insect).

To illustrate the differences between PCA and MDA for this dataset, let us begin by drawing the data in the principal plan.

```
Xbar <- scale(X[, 1:6], scale = F)
Sigma <- t(Xbar) %*% Xbar/nrow(Xbar)
ACP <- eigen(Sigma)
C <- Xbar %*% ACP$vectors
cos2 <- rowSums(C[, 1:2]^2)/rowSums(C^2)
plot(C[, 1:2], cex = cos2)
```

Comment this graph. Are you able to discriminate three groups of insects ? Compare your intuition with the help of the following command,

```
plot(C[, 1:2], cex = cos2, col = X[, 7])
```

To set up the MDA procedure, we need the matrix `T` that assigns each individual to the group of its label.

```
T <- matrix(0, nrow = nrow(X), ncol = 3)
for (i in 1:nrow(X)) T[i, X[i, 7]] <- 1
```

Check and explain the content of this matrix. Each label defines a group of individuals. Using uniform weights, each group can be weighted proportionally to its size.

```
W <- diag(rep(1/nrow(X), nrow(X)))
Wbar <- t(T) %*% W %*% T
```

To pull apart the variance *between* and the variance *within*, we consider the matrix `G` that contains the centers of gravity of each group and its centered version `Gbar`.

```
G <- diag(1/diag(Wbar)) %*% t(T) %*% W %*% X[, 1:6]
g <- colMeans(X[, 1:6])
Gbar <- G - c(1, 1, 1) %*% t(g)
```

This last matrix allows us to compute the *between* variance matrix,

```
SigmaB <- t(Gbar) %*% Wbar %*% Gbar
```

Take the time to understand the previous lines. In particular, you can verify that the matrix `Wbar` is diagonal. Moreover, its diagonal elements are the weights of the groups, by definition. Writing the expression `G[i,j]`, you can also verify that the lines of `G` contain the centers of gravity of the groups.

We have seen that doing a MDA amounts to compute the PCA of `Gbar` with the weight matrix `Wbar` and the Mahalanobis' distance given by the inverse of `Sigma`. Thus, we can do the computations as we did in the previous practicals,

```
M <- solve(Sigma)
EigenM <- eigen(M)
P <- EigenM$vectors
Mhalf <- P %*% diag(sqrt(EigenM$values)) %*% t(P)

Gprime <- Gbar %*% Mhalf
AFD <- eigen(t(Gprime) %*% Wbar %*% Gprime)

C <- Xbar %*% Mhalf %*% AFD$vectors
Cbar <- Gprime %*% AFD$vectors
```

Explain the computation of `Mhalf` and recall why it is easier to handle `Gprime` than `Gbar` for our computations.

In a MDA with m groups, we know that there is only $\kappa = \min(m-1, p)$ non trivial eigenvalues. What is the value of κ for our data ? Comment the results of the following command,

```
cumsum(AFD$values)/sum(AFD$values)
```

What is the value of `AFD$values[1]` and how do you interpret that ? We finish with a plot of the results of the MDA in the plan,

```
plot(C[, 1:2], col = X[, 7])
points(Cbar[, 1:2], pch = 15)
```

Compare this graph with the one obtained from PCA.

3 Mails and spams (CART)

We now focus on the CART procedure and the mail/spam data of Hewlett-Packard. This dataset contains $n = 4601$ mails with 1813 spams among them and, for each mail, we have $p = 57$ variables relative to some word frequencies, to some character frequencies and to sequences of capital characters. These data can be obtained with the command,

```
X <- DataSpam()
```

Our goal is to provide a supervised learning procedure. This time, there are 2 labels : "Mail" and "Spam". Explain why the MDA can only provide a limited answer to this problem. To check if the label of a mail is "Mail" or "Spam", we use the last column of `X`, called `Status`,

```
X$Status
```

To set up the procedure, we use the package `rpart`. Let us load it and have a look to the manual pages related to the main functions,

```
library(rpart)
help(rpart)
```

You see in these pages that there are three mandatory parameters for the function `rpart` :

`formula` this parameter indicates what is the variable to explain and what are the explanatory variables. The syntax is common for the R software and is the same as the one for the function `lm`. We do not give more details here about R objects of type *formula* and we will use the variable `spam_form` defined as follows,

```
spam_form <- paste("Status ~ ", paste(colnames(X)[1:57], collapse = " + "))
```

`method` as explained during the lecture, the CART procedure can be handled for other aims than supervised learning. Here, we just want to classify the data, then we choose the value `"class"`.

`data` this is the set of data used in the above formula. For us, this is simply `X`.

Thus, we get our first decision tree with the command

```
tree <- rpart(formula = spam_form, method = "class", data = X)
```

The command `print(tree)` displays the variables and the thresholds computed for each binary test and `summary(tree)` gives you more informations. To print the tree, you can run the following commands,

```
plot(tree, uniform = TRUE)
text(tree, all = TRUE)
```

Take some time to read the manual page about `text.rpart` in order to understand all the informations that you can display on the tree. Comment these results.

During the lecture, we have discussed about the problem of overfitting due to a too great adequation to the data. In particular, we have introduced a penalized procedure for pruning a decision tree to close to the dataset. The command `rpart` proceeds in the same way and the control parameters are passed through the argument `control`. Read the manual pages about the object `rpart.control` for more details. Consider now the tree obtained by the following commands,

```
spam_ctrl <- rpart.control(cp = 1e-04)
tree <- rpart(formula = spam_form, method = "class", data = X, control = spam_ctrl)
```

What is the role of the parameter `cp` ? Change this value and comment what you get. What is the link with the pruning method seen during the lecture ?

To conclude, let us prune by ourselves the large tree obtained with the parameter `cp=0.0001`. A common method to do that amounts to estimate the error rate by cross-validation (see the parameter `xval` of `spam_ctrl`). This error is computed by `rpart` and you can read it with the command `printcp(tree)` in the column `xerror`. It is feasible to see this error with respect to the size of the tree,

```
plotcp(tree)
```

What is the value of `cp` suggested by this graph ? Prune the tree with this value with the function `prune`. Does the result seem satisfactory ? What is the problem ? With the help of the informations given by `tree$cptable`, propose a method similar to the one seen during the lecture for doing a "graphical" choice of the value `cp`.