SY19 - TP 5

Automne 2011

I-Problématique

Le but de ce TP est la mise en œuvre d'un algorithme basé sur les réseaux de neurones pour la classification.

II-Travail Préliminaire

On demande de simuler des données de $\mathbb{R}^2$, suivant des lois gaussiennes et avec des probabilités à priori égales, puis de dessiner les frontières de Bayes.

```
##################################################

p1 <- 0.25
p2 <-0.25
p3 <- 0.25
p4 <- 0.25

n <- 200

s1 <- c(1,2)
s2 <- c(2,1)
s3 <- c(1.5,2)
s4 <- c(1,1)

s <- rbind(s1,s2,s3,s4)

m1 <- c(4,6)
m2 <- c(6,1)
m3 <- c(-4,-4)
m4 <- c(0,0)

m <- rbind(m1,m2,m3,m4)

c <- sample(c(1,2,3,4),size=n,prob=c(p1,p2,p3,p4),replace=TRUE)
x<- cbind(rnorm(n,m[c,1],s[c,1]),rnorm(n,m[c,2],s[c,2]))


couleur <- rep("red",n)

couleur[c==2]<-"blue"
couleur[c==3]<-"green"
couleur[c==4]<-"yellow"

plot(x,col=couleur)
```

```
# Frontiere de Bayes

len<-50

xp<-seq(min(x[,1]),max(x[,1]), length=len)
yp<-seq(min(x[,2]),max(x[,2]), length=len)

grille<-expand.grid(z1=xp,z2=yp)

Z<-p1*dnorm(grille[,1],m[1,1],s[1,1])*dnorm(grille[,2],m[1,2],s[1,2])
Z<-cbind(Z,p2*dnorm(grille[,1],m[2,1],s[2,1])*dnorm(grille[,2],m[2,2],s[2,2]))
Z<-cbind(Z,p3*dnorm(grille[,1],m[3,1],s[3,1])*dnorm(grille[,2],m[3,2],s[3,2]))
Z<-cbind(Z,p4*dnorm(grille[,1],m[4,1],s[4,1])*dnorm(grille[,2],m[4,2],s[4,2]))

zp<-Z[,4] - pmax(Z[,3],Z[,2], Z[,1])
contour(xp,yp,matrix(zp,len),add=TRUE,levels=0,drawlabels=FALSE)

zp<-Z[,1] - pmax(Z[,2], Z[,3],Z[,4])
contour(xp,yp,matrix(zp,len),add=TRUE,levels=0,drawlabels=FALSE)

zp <- Z[,2] - pmax(Z[,1], Z[,3],Z[,4])
contour(xp,yp,matrix(zp,len),add=TRUE,levels=0,drawlabels=FALSE)



################################################
```

III-Réseaux de neurones sur des données simulées

Sous R, la fonction nnet de la librairie nnet permet de calculer les poids optimaux d'un perceptron à une couche cachée selon la technique de retro-propagation de l'erreur. La syntaxe simplifiée de la procédure nnet sous R est la suivante :

```
model <- nnet(x,T,size=3,decay=10,softmax=TRUE,maxit=500)
```

Les paramètres de la procédure sont :

x	matrice $n \times d$ de variables quantitatives
T	matrice $n \times K$ contentant des 0 ou des 1 selon la classe des observations x
size	nombre de neurones de la couche cachée
decay	paramètre de régularisation
softmax	booléen pour une fonction de sortie $\widetilde{h}$ softmax
maxit=500	nombre maximal d'itérations effectuées par l'algorithme de rétro-propagation

On peut utiliser la fonction predict pour appliquer le réseau de neurones à de nouvelles données et obtenir les probabilités a posteriori d'appartenance aux classes. Par exemple, si grille est un ensemble de points de taille $p \times d$ dont on veut avoir les classes :

$>$ predict(model,grille) retourne une matrice de taille $p \times K$ dont les colonnes contiennent les probabilités a posteriori d'appartenance aux classes pour chaque point de grille.

Pour pouvoir appliquer un réseau de neurones au mélange Gaussien décrit précédemment, on peut utiliser les commandes suivantes :

```
#####################################################

# Reseau de neurones : fonction nnet

library(nnet)

T<-class.ind(couleur)
model<-nnet(x,T,size=1,decay=0,softmax=TRUE,maxit=500)

# Valeur des poids

model$wts

# Proba. a posteriori d appartenance aux classes

Z<-predict(model,grille)

#####################################################
```

Question 1. Pour decay = 0 et size = 5 :

1. Calculer les frontières de décision obtenues avec un réseau de neurones pour le mélange de 4 vecteurs Gaussien.
2. Visualiser l'estimation des poids.
3. Lancer plusieurs fois la procédure nnet. Obtenez-vous les mêmes résultats à chaque fois pour l'estimation des poids et des frontières de décision ?
4. Comment pouvez-vous expliquer ce phénomène ?

Afin de ne pas avoir une source de variation dans l'estimation des poids optimaux qui puisse venir de l'initialisation aléatoire des poids, il est important de toujours utiliser les mêmes poids pour initialiser l'algorithme de rétro-propagation de l'erreur, si l'on veut pouvoir juger de l'influence de size ou decay. Pour cela, il suffit d'utiliser la commande set.seed pour initialiser le générateur aléatoire de R à la même valeur avant de lancer la procédure nnet :

```
#####################################################

set.seed(1)
T <- class.ind(couleur)
model<-nnet(x,T,size=1,decay=0,softmax=TRUE,maxit=500)

#####################################################
```

Question 2. On désire voir l'influence du nombre de neurones dans la couche cachée.

1. Pour decay = 0, faire varier size de 1 à 10 (nombre de neurones dans la couche cachée) et visualiser l'estimation des poids.
2. Que constatez-vous sur la forme des frontières et sur le nombre de points mal-classés quand size augmente ?
3. Relancer la procédure sur un nouveau jeu de données en faisant varier size de 1 à 10.
4. Que pouvez-vous dire de la variabilité du modèle en fonction de size ?

Question 3. On désire voir l'influence du paramètre de régularisation.

1. Pour size = 5, comparer l'estimation des poids et des frontières pour decay=0.001, 0.01, 0.1, 1, 10, 100.
2. Que pouvez-vous dire de la valeur des poids optimaux quand decay augmente ?
3. Comment expliquez-vous ce phénomène et Comment cela se traduit-il sur la forme des frontières ?

IV- Classification des données de crabes

Le data.frame *crabs* de la librairie *nnet* contient des données réelles, recueilles par le biologiste Mahon (1974) à partir de mesures effectuées sur 200 crabes de la variété Leptograpsus et Variegatus sur la côte ouest de l'Australie. Les crabes sont des mâles ou des femelles qui sont bleus ou oranges. Pour chaque crabe, 5 mesures ont été effectuées :
– la longueur de la carapace (variable CL)
– la largeur de la carapace (variable CW)
– la taille du lobe frontal (variable FL)
– la largeur de l'arrière du crabe (variable RW)
– l'épaisseur du corps (variable BD)
On souhaite construire une règle de classification pour prédire le sexe ou bien la couleur d'un crabe à partir de deux mesures seulement : par exemple FL et RW. Vous pouvez représenter les crabes en coordonnées logarithmique selon leur sexe ou leur couleur à l'aide des commandes suivantes :

```
################################################
 # Separation selon le sexe
n <- 200
x <- log(cbind(crabs$FL,crabs$RW))
T<-class.ind(crabs$sex)
couleur <- rep('red',n)
couleur[crabs$sex == "M"] <- 'blue'
plot(x,col=couleur)

# Separation selon la couleur

 x<- log(cbind(crabs$FL,crabs$RW))
 T <- class.ind(crabs$sp)
 couleur<- rep('red',n)
 couleur[crabs$sp == "O"] <- 'blue'
plot(x,col=couleur)
################################################
```

Question 4. Ajuster un réseau de neurones pour prédire le sexe des crabes à partir de FL et RW.

1. Quelles valeurs de size et decay vous semblent donner les meilleurs résultats ?
2. Comparer avec l'analyse discriminante quadratique et l'analyse discriminante linéaire.
3. Que constatez-vous ?

Question 5. Ajuster un réseau de neurones pour prédire la couleur des crabes à partir de FL et RW.

1. Quelles valeurs de size et decay vous semblent donner les meilleurs résultats ?
2. Comparer avec l'analyse discriminante quadratique et l'analyse discriminante linéaire.
3. Que constatez-vous ?