SY09 Printemps 2016
TP 3
Discrimination, théorie bayésienne de la décision

1 Classifieur euclidien, K plus proches voisins

On souhaite étudier les performances du classifieur euclidien et des K plus proches voisins sur différents jeux de données binaires (constitués d'individus de $\mathbb{R}^p$ répartis dans $g = 2$ classes).

1.1 Programmation

1.1.1 Classifieur euclidien

Écrire les fonctions ceuc.app et ceuc.val.

```
ceuc.app <- function(Xapp, zapp)
{
...
}
ceuc.val <- function(mu, Xtst)
{
...
}
```

La première fait l'apprentissage des paramètres du classifieur euclidien : elle doit donc prendre en argument d'entrée un tableau individus-variables Xapp (de dimensions napp $\times p$) correspondant aux individus d'apprentissage, et le vecteur zapp (de longueur napp) des étiquettes associées à chacun des individus. Elle doit retourner les paramètres estimés du classifieur euclidien, sous la forme d'une matrice mu de dimensions $g \times p$.

La seconde fait le classement d'un tableau individus-variables Xtst (de dimensions ntst $\times p$) : elle doit donc prendre en argument d'entrée la matrice mu des paramètres estimés et l'ensemble à évaluer Xtst, et retourner un vecteur (de longueur ntst) d'étiquettes prédites. On pourra s'aider de la fonction distXY fournie qui calcule les distances (euclidiennes au carré) entre les individus de deux ensembles X et Y, et de la fonction which.min qui détermine l'indice de l'élément minimal d'un vecteur.

1.1.2 K plus proches voisins

Écrire les fonctions kppv.tune et kppv.val.

```
kppv.tune <- function(Xapp, zapp, Xval, zval, nppv)
{
...
}
kppv.val <- function(Xapp, zapp, K, Xtst)
{
...
}
```

La première détermine le nombre « optimal » de voisins K_{opt} (choisi parmi un vecteur nppv de valeurs possibles), c'est-à-dire donnant les meilleurs performances sur un ensemble de validation étiqueté (matrice Xval de dimensions nval $\times$ p et vecteur zval de longueur nval). Elle doit donc prendre en argument d'entrée le tableau individus-variables Xapp, le vecteur zapp des étiquettes associées, le tableau Xval, le vecteur zval, et un ensemble de valeurs nppv. Elle doit retourner la valeur de K_{opt} (choisie parmi les valeurs contenues dans nppv).

La seconde fait le classement d'un tableau individus-variables Xtst : elle prend donc en argument Xapp et zapp, le nombre de voisins K, et l'ensemble à évaluer Xtst ; elle retourne donc un vecteur (de longueur ntst) d'étiquettes prédites. Il sera judicieux de faire appel à la fonction kppv.val dans la fonction kppv.tune. On pourra également utiliser les fonctions distXY et which.max dans la fonction kppv.val.

1.1.3 Test des fonctions

Vous pouvez utiliser le jeu de données Synth1-40 pour tester les fonctions que vous développez. Une fois téléchargé, vous pouvez charger le jeu de données, puis séparer les individus et les étiquettes de manière à former un ensemble d'apprentissage et un ensemble de test :

```
donn <- read.table("Synth1-40.txt", header=F)
X <- donn[,1:2]
z <- donn[,3]

Xapp <- X[c(1:15,24:35),]
zapp <- z[c(1:15,24:35)]
Xtst <- X[c(16:20,36:40),]
ztst <- z[c(16:20,36:40)]
```

Pour visualiser les frontières de décision obtenues à l'aide des fonctions ceuc.val et kppv.val, vous pouvez utiliser les fonctions front.ceuc et front.kppv fournies (voir paragraphe 1.3.1).

1.2 Évaluation des performances

On dispose de cinq jeux de données (téléchargeables sur le site de l'UV) : Synth1-40, Synth1-100, Synth1-500, Synth1-1000, et Synth2-1000. Pour chacun de ces jeux de données, chaque classe a été générée suivant une loi normale bivariée. Les distributions sont les mêmes pour les jeux de données Synth1-40, Synth1-100, Synth1-500 et Synth1-1000, qui ne diffèrent que par le nombre d'exemples disponibles. En revanche, la distribution des données dans Synth2 est différente.

Pour un jeu de données, lorsqu'on souhaite estimer le taux d'erreur ε d'un classifieur, on utilise généralement la procédure suivante. On répète N fois l'expérience qui consiste à :
- séparer l'ensemble de données disponible aléatoirement de manière à former un ensemble d'apprentissage et un ensemble de test (et, de manière optionnelle, un ensemble de validation) ;
- apprendre les paramètres du modèle sur l'ensemble d'apprentissage formé (après avoir éventuellement optimisé les hyper-paramètres sur l'ensemble de validation, s'il y a lieu), et calculer le taux d'erreur obtenu sur l'ensemble de test.

En déterminant de la sorte N séparations aléatoires de l'ensemble de données en un ensemble d'apprentissage et un ensemble de test, on peut ainsi recueillir un échantillon de N estimations $E_1, \ldots, E_N$ du taux d'erreur (généralement effectuées sur l'ensemble de test). On peut alors déterminer une estimation ponctuelle de ε (moyenne) et un intervalle de confiance sur ε.

1.2.1 Jeux de données Synth1-40, Synth1-100, Synth1-500 et Synth1-1000

On considère tout d'abord les jeux de données Synth1-40, Synth1-100, Synth1-500 et Synth1-1000.

1. Pour chacun des jeux de données, estimer les paramètres μ_k et Σ_k des distributions conditionnelles, ainsi que les proportions π_k des classes.

2. Écrire un script qui effectue $N = 20$ séparations aléatoires de chaque jeu de données en ensembles d'apprentissage et de test, et qui calcule (et stocke) pour chacune le taux d'erreur d'apprentissage et le taux d'erreur de test. On pourra utiliser la fonction separ1 pour séparer les données (voir paragraphe 1.3.2).

   En considérant ensuite l'ensemble des résultats obtenus lors des $N = 20$ expériences, donner l'estimation ponctuelle du taux d'erreur ε ainsi qu'un intervalle de confiance, d'abord à partir des estimations faites sur l'ensemble d'apprentissage, puis celles faites sur l'ensemble de test. Qu'observez-vous ?
3. Effectuer une séparation aléatoire de l'ensemble de données en un ensemble d'apprentissage et un ensemble de test. Déterminer le nombre optimal de voisins à l'aide de la fonction kppv.tune, en utilisant l'ensemble d'apprentissage comme ensemble de validation. Quel est le nombre optimal de voisins déterminé ? Pourquoi ?
4. Comme pour le classifieur euclidien, écrire un script qui effectue $N = 20$ séparations aléatoires de chaque jeu de données en ensembles d'apprentissage, de validation, et de test ; et qui pour chacune détermine le nombre optimal de voisins à partir d'un ensemble de validation spécifique, puis calcule (et stocke) le taux d'erreur sur l'ensemble d'apprentissage et sur l'ensemble de test. On pourra utiliser la fonction separ2 pour séparer les données (voir paragraphe 1.3.2).

   Comme précédemment, déterminer les estimations ponctuelles du taux d'erreur et les intervalles de confiance obtenus à partir des erreurs d'apprentissage, puis des erreurs de test. Comparer avec les résultats obtenus pour le classifieur euclidien, et commenter.

1.2.2 Jeu de données Synth2-1000

On considère maintenant le jeu de données Synth2-1000.

1. Estimer les paramètres μ_k et Σ_k ainsi que les proportions π_k des classes.
2. Comme précédemment, calculer les estimations (ponctuelles et intervalles de confiance) de ε sur l'ensemble d'apprentissage et sur l'ensemble de test. Commenter et interpréter.

1.3 Note sur l'utilisation des fonctions

1.3.1 Frontières de décision

Les fonctions front.ceuc et front.kppv échantillonnent l'espace des caractéristiques en déterminant une grille de points : on détermine les décisions prises pour les points de cette grille, puis on trace les frontières de décision en utilisant la fonction contour. On peut les appeler comme suit.

```
mu <- ceuc.app(Xapp, zapp)
zpred <- ceuc.val(mu, Xtst)
front.ceuc(mu, Xapp, zapp)
```

```
Kopt <- kppv.tune(Xapp, zapp, Xval, zval, 1:10)
zpred <- kppv.val(Xapp, zapp, Kopt, Xtst)
front.kppv(Xapp, zapp, Kopt, Xapp, zapp)
```

1.3.2 Séparation des données

La fonction separ1 détermine un ensemble d'apprentissage (de taille napp $= 2n/3$) et un ensemble de test (de taille ntst $= n/3$). La fonction separ2 détermine des ensembles d'apprentissage (de taille napp $= n/2$), de validation (de taille nval $= n/4$) et de test (de taille ntst $= n/4$).

```
donn.sep <- separ1(X, z)
Xapp <- donn.sep$Xapp
zapp <- donn.sep$zapp
Xtst <- donn.sep$Xtst
ztst <- donn.sep$ztst
```

```
donn.sep <- separ2(X, z)
Xapp <- donn.sep$Xapp
zapp <- donn.sep$zapp
Xval <- donn.sep$Xval
zval <- donn.sep$zval
Xtst <- donn.sep$Xtst
ztst <- donn.sep$ztst
```

2 Règle de Bayes

En réalité, les jeux de données étudiés précédemment ont été obtenus de la manière suivante :

1. tout d'abord, l'effectif n_1 de la classe ω_1 a été déterminé par tirage aléatoire suivant une loi binomiale de paramètres n et $\pi_1 = 0.5$;
2. n_1 individus ont ensuite été générés dans la classe ω_1 suivant une loi normale bivariée de paramètres μ_1 et Σ_1, et $n_2 = n - n_1$ individus ont été générés dans la classe ω_2 suivant une loi normale bivariée de paramètres μ_2 et Σ_2.

   Pour les jeux de données Synth1-40, Synth1-100, Synth1-500 et Synth1-1000, on a utilisé comme paramètres $n = 40$, $n = 100$, $n = 500$, et $n = 1000$, respectivement ; et

$$\mu_1 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \quad \Sigma_1 = \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix};$$

   Pour le jeu de données Synth2-1000, on a utilisé $n = 1000$, et

$$\mu_1 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 0 \\ -5 \end{pmatrix}, \quad \Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}.$$

Pour chacun des jeux de données :

1. quelles sont les distributions marginales des variables X^1 et X^2 dans chaque classe ?
2. Montrer que dans chaque classe, les courbes d'iso-densité sont des cercles dont on précisera les centres et les rayons.
3. Déterminer l'expression de la règle de Bayes pour le problème de discrimination des classes ω_1 et ω_2.
4. Pour les quatre premiers jeux de données, tracer avec R les frontières de décision dans le plan formé par les variables X_1 et X_2.
5. Calculer l'erreur de Bayes pour les quatre premiers jeux de données, et comparer aux résultats obtenus dans l'exercice précédent.