SY09 : Rapport TP3
Discrimination, théorie bayésienne de la décision

Cathelain Valentin Marechal Anaig

25 mai 2016

Résumé

Dans les TPs précédents, nous avons pu découvrir d'une part *l'analyse exploratoire*, qui permet d'obtenir une vision globale des données, et d'autre part nous avons appris à prendre en main *la classification automatique*, qui permet de créer des classes à partir de groupes homogènes d'individus.
Ces méthodes et outils précédemment utilisés faisaient partie de *l'analyse supervisée* : on met les individus en relation les uns avec les autres. Cela permet d'avoir un premier regard sur les données et de mieux les connaître. Le défi face à nous à présent est d'étendre nos observations grâce aux *méthodes supervisées*. Celles-ci consistent à attribuer des étiquettes à de nouveaux individus à partir d'un jeu de données déjà connues.
Le but de ce TP est tout d'abord d'utiliser l'apprentissage supervisé comme méthode de discrimination. Pour cela, nous utiliserons deux classifieurs différents : *le classifieur euclidien* et *la méthodes des K plus proches voisins*. Dans le cadre de ce TP nous considérerons uniquement les cas avec deux classes ω_1 et ω_2. Puis dans un second temps nous aborderons la théorie bayésienne de la décision, qui va nous permettre de trouver une règle de décision *optimale*.

1 Programmation des classifieurs

1.1 Classifieur euclidien

Le classifieur euclidien a un fonctionnement relativement simple. Chaque individu de test est affecté au groupe dont le centre de gravité est le plus proche. Tout d'abord on dispose d'un ensemble d'apprentissage composé des individus-variables **Xapp** et du vecteur **zapp** des étiquettes associées à chaque individu.
On calcule les centres de gravité des groupes d'individus (ici il n'y en aura que 2) avec la fonction **ceuc.app** (voir en annexe). Ensuite en on parcours chaque individu de test et on lui attribut la classe pour laquelle la distance avec le centre de gravité est minimale à l'aide de la fonction **ceuc.val**.

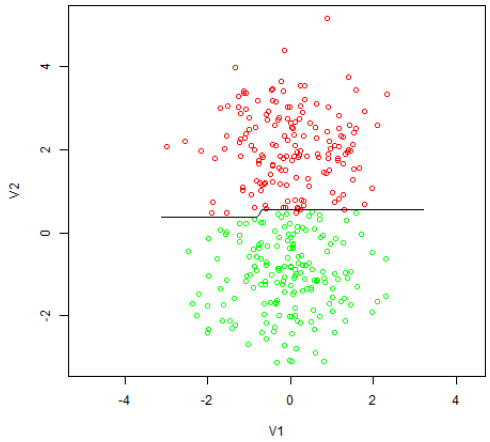

FIGURE 1 – Discrimination avec le classifieur euclidien

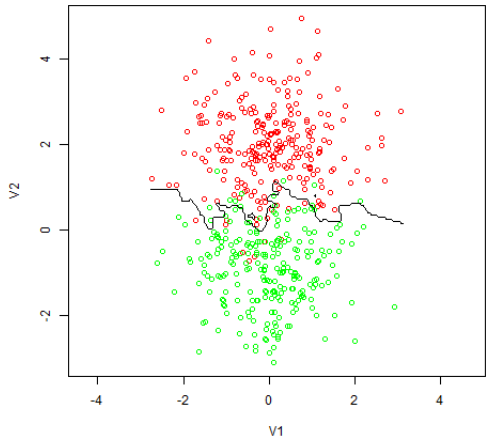

FIGURE 2 – Discrimination avec les K plus proches voisins

1.2 Règle des K plus proches voisins

Cette règle de décision consiste à affecter une étiquette à un individu en fonction de la classe la plus représentée parmi ses K plus proches voisins.
La fonction **kppv.val** affectera les étiquettes aux vecteurs **Xtst**. Pour connaître le K optimal, nous ferons appel à la fonction **kppv.tune** comme nous le verrons dans la partie suivante.

2 Évaluation des performances

2.1 Estimation des paramètres des distributions conditionnelles

Pour les 4 jeux de données Synth1-40, Synth 1-100, Synth 1-500, Synth 1-1000, les classes ont été générées suivant une loi normale bivariée de paramètres ν_k et Σ_k, avec les proportions π_k.

	Synth 1-40	Synth 1-100	Synth 1-500	Synth 1-1000
π_1 π_2	0.58 0.42	0.46 0.54	0.52 0.48	0.48 0.52
μ_1 μ_2	-0.05 ; 1.98 0.11 ; -0.70	0.17 ; 2.06 -0.19 ; -1.06	0.03 ; 2.00 -0.03 ; -0.90	-0.08 ; 1.98 -0.04 ; -1.01
Σ_1	$\begin{pmatrix} 1.45 & -0.11 \\ -0.11 & 0.89 \end{pmatrix}$	$\begin{pmatrix} 0.89 & 0.07 \\ 0.07 & 0.89 \end{pmatrix}$	$\begin{pmatrix} 0.99 & 0 \\ 0 & 0.94 \end{pmatrix}$	$\begin{pmatrix} 1.03 & 0.05 \\ 0.05 & 1.02 \end{pmatrix}$
Σ_2	$\begin{pmatrix} 1.75 & -0.32 \\ -0.32 & 0.83 \end{pmatrix}$	$\begin{pmatrix} 0.90 & 0.07 \\ 0.07 & 0.61 \end{pmatrix}$	$\begin{pmatrix} 0.90 & -0.02 \\ -0.02 & 0.82 \end{pmatrix}$	$\begin{pmatrix} 0.98 & -0.02 \\ -0.02 & 0.95 \end{pmatrix}$

On remarque que plus l'ensemble de données est grand, plus les paramètres tendent vers les valeurs suivantes :

$$\pi_1 = \pi_2 = 0.5, \mu_1 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \mu_2 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \Sigma_1 = \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

2.2 Détermination de K_{opt}

La fonction **kppv.tune** nous permet de déterminer le nombre K optimal, c'est à dire celui qui minimisera le taux d'erreur. Ici $K \in 1 \dots 10$.

Exemple On charge le fichier de données Synth 1-1000, composé de 1000 individus. On sépare ensuite ces données en un ensemble d'apprentissage (de longueur 500), un ensemble de validation (de longueur 250) et un ensemble de test (de longueur 250). Et on soumet une liste de chiffres de 1 à 10 en entrée de la fonction **kppv.tune**.
Il en ressort les taux d'erreur suivants :

K	1	2	3	4	5	6	7	8	9	10
ϵ	0.072	0.1	0.06	0.06	0.056	0.056	0.056	0.056	0.056	0.056

On peut alors facilement affirmer que $K_{opt} = 5$. En effet on prendra la plus petite valeur de K qui minimise le taux d'erreur de façon à limiter les calculs lors de l'appel à la fonction **kppv.val**.

2.3 Estimation des taux d'erreur

2.3.1 Classifieur euclidien

Nous avons calculé les taux d'erreur pour les 4 jeux de données. Les valeurs sont des moyennes obtenues après 20 séparations.

	Synth 1-40	Synth 1-100	Synth 1-500	Synth 1-1000
ϵ apprentissage	0.08	0.06	0.05	0.07
Intervalle de confiance ensemble d'apprentissage	0 - 0.12	0.03-0.07	0.03 - 0.07	0.06 - 0.07
ϵ test	0.09	0.05	0.05	0.06
Intervalle de confiance ensemble de test	0 - 0.21	0 - 0.09	0.02 - 0.08	0.04 - 0.08

2.3.2 K plus proches voisins

	Synth 1-40	Synth 1-100	Synth 1-500	Synth 1-1000
ϵ apprentissage	0.02	0.01	0.05	0.05
Intervalle de confiance ensemble d'apprentissage	0 - 0.1	0 - 0.04	0.02 - 0.06	0 - 0.08
ϵ test	0.08	0.06	0.07	0.07
Intervalle de confiance ensemble de test	0 - 0.2	0 - 0.13	0.03 - 0.10	0.05 - 0.1

Les taux d'erreur sur l'ensemble d'apprentissage et sur l'ensemble de test sont relativement égaux pour les deux classifieurs. Plus la quantité de donnée augmente, plus les taux d'erreur sur l'ensemble de test diminuent, ce qui parait logique car plus il y a de données plus l'ensemble d'apprentissage est grand et plus les résultat sont fiables. Cependant, si les taux d'erreur des invidus d'apprentissage peuvent paraître satisfaisants, on peut émettre l'hypothèse que les résultats sont biaisés. Ce sont en effet sur ces données mêmes que nous nous sommes basés pour construire la règle de décision, elles ont donc trop d'influence pour pouvoir être utilisées comme individus de test par la suite. Les individus d'apprentissage ne sont pas forcément représentatifs des autres classes, c'est pourquoi il vaut mieux utiliser un autre ensemble pour les tests pour gagner en objectivité.

Même s'il est difficile de dégager une tendance des taux d'erreur pour les ensembles d'apprentissage, on remarque que les intervalle de confiance deviennent rétrécissent globalement avec l'augmentation de la taille des données.
Par ailleurs on remarque que les taux d'erreur sont plus faibles avec le classifieur des K plus proches voisins qu'avec le classifieur euclidien.

2.4 Paramètres du jeu de données Synth2-1000

2.4.1 Paramètres

Sur l'échantillon Synth2-1000 nous avons les valeurs suivantes :

$$\pi_1 = 0.49, \pi_2 = 0.51, \mu_1 = \begin{pmatrix} -0.07 \\ 2.94 \end{pmatrix}, \mu_2 = \begin{pmatrix} -0.10 \\ -5.10 \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 1.07 & -0.05 \\ -0.05 & 1.07 \end{pmatrix} \Sigma_2 = \begin{pmatrix} 5.22 & 0.11 \\ 0.11 & 5.17 \end{pmatrix}$$

On peut donc en conclure que le jeu de donnée Synth2-1000 suit une loi bivariée avec les paramètres suivants :

$$\pi_1 = \pi_2 = 0.5, \mu_1 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \mu_2 = \begin{pmatrix} 0 \\ -5 \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}$$

2.4.2 Performances

Méthode	Classifieur euclidien	K plus proches voisins
ϵ apprentissage	0.0227	0.0008
Intervalle de confiance ensemble d'apprentissage	0.016 - 0.300	0 - 0.006
ϵ test	0.0195	0.0058
Intervalle de confiance ensemble de test	0.006 - 0.033	0 - 0.012

3 Règle de Bayes

3.1 Distributions marginales

Les individus n_1 et n_2 des classes w_1 et w_2 ont été générés suivant une loi binomiale bivariée de paramètres μ_1 et Σ_1 et μ_2 et Σ_2. On a donc $X_1 \sim (\mu_1, \Sigma_1)$ et $X_2 \sim (\mu_2, \Sigma_2)$, où μ_k représente l'espérance et Σ_k la matrice de covariance.
Pour les quatre premiers jeux de données, pour X_1, on a alors :
$X_{11} \sim (\mu_{11}, \Sigma_{11}) \Rightarrow$X$_{11} \sim (0, 1)$ et $X_{12} \sim (\mu_{12}, \Sigma_{12}) \Rightarrow$X$_{12} \sim (2, 1)$.
Pour X_2 :
$X_{21} \sim (\mu_{21}, \Sigma_{21}) \Rightarrow$X$_{21} \sim (0, 1)$ et $X_{22} \sim (\mu_{22}, \Sigma_{22}) \Rightarrow$X$_{22} \sim (-1, 1)$.
Concernant le dernier jeu de données, on a de la même façon pour X_1 :
$X_{11} \sim (0, 1)$ et $X_{12} \sim (3, 1)$.
Et pour X_2 :
$X_{21} \sim (0, 5)$ et $X_{22} \sim (-5, 5)$.
La *distribution marginale* est la densité de chaque variable sans prendre en compte les autres.
A partir des informations précédentes, on peut donc calculer la distribution marginale de chaque variable à partir de la formule de la densité de la loi normale multivariée :

$$f_X(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \mu)^{\mathrm{t}} \Sigma^{-1} (\boldsymbol{x} - \mu)\right) \quad (1)$$

Ainsi, sachant que pour les premiers jeux de données $|\Sigma_1| = |\Sigma_2| = 1$ et $\Sigma_1^{-1} = \Sigma_2^{-1} = 1$, on remplace dans la formule et on obtient les distributions marginales :

— $f_{11}(\boldsymbol{x}_1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\boldsymbol{x}_1^2\right)$
— $f_{12}(\boldsymbol{x}_1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_2 - 2)^2\right)$
— $f_{21}(\boldsymbol{x}_1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\boldsymbol{x}_1^2\right)$
— $f_{22}(\boldsymbol{x}_1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_2 + 1)^2\right)$

Pour le dernier jeu de données, on a $|\Sigma_1| = 1$, $|\Sigma_2| = 25$, $\Sigma_1^{-1} = 1$ et$\Sigma_2^{-1} = 1$, ce qui nous donne comme distribution marginale :

— $f_{11}(\boldsymbol{x}_1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\boldsymbol{x}_1^2\right)$
— $f_{12}(\boldsymbol{x}_1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_2 - 3)^2\right)$
— $f_{21}(\boldsymbol{x}_1) = \frac{1}{5\sqrt{2\pi}} \exp\left(-\frac{1}{2}\boldsymbol{x}_1^2\right)$
— $f_{22}(\boldsymbol{x}_1) = \frac{1}{5\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_2 + 5)^2\right)$

3.2 Courbe d'iso-densité

La *courbe d'iso-densité* est la densité conditionnelle du vecteur X dans chacune des classes ω_K. Ainsi, on veut que pour chaque classe la densité soit la même, c'est-à-dire une constante. On peut donc définir la classe d'iso-densité comme $\{\boldsymbol{x} \text{ tq } f_k(\boldsymbol{x}) = K\}$. Nous allons maintenant prouver que dans notre cas cette expression est équivalente à un *cercle*.
Pour chaque jeu de données, on observe que la matrice de covariance est diagonale. On en déduit que les variables sont indépendantes. Cela signifie qu'on a :
$f_{X1} = f_{11}(\boldsymbol{x}_1)f_{12}(\boldsymbol{x}_2)$ et $f_{X2} = f_{21}(\boldsymbol{x}_1)f_{22}(\boldsymbol{x}_2)$.
Commençons par les quatre premiers jeux de données. Pour ω_1 :

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\boldsymbol{x}_1^2\right) * \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_2 - 2)^2\right) = K_1 \tag{2}$$

$$\Leftrightarrow \frac{1}{2\pi} \exp\left(-\frac{1}{2}(\boldsymbol{x}_1^2 + (\boldsymbol{x}_2 - 2)^2)\right) = K_1 \tag{3}$$

$$\Leftrightarrow -\frac{1}{2}(\boldsymbol{x}_1^2 + (\boldsymbol{x}_2 - 2)^2) = \ln(2\pi K_1) \tag{4}$$

$$\Leftrightarrow \boldsymbol{x}_1^2 + (\boldsymbol{x}_2 - 2)^2 = -2\ln(2\pi K_1) \tag{5}$$

On en déduit que la courbe d'iso-densité est un cercle de centre (0,2) et de rayon $\sqrt{-2\ln(2\pi K_1)}$.
De la même façon, pour ω_2, on a $\boldsymbol{x}_1^2 + (\boldsymbol{x}_2 + 1)^2 = -2\ln(2\pi K_2)$. La courbe d'iso-densité est alors un cercle de centre (0,-1), de rayon $\sqrt{-2\ln(2\pi K_2)}$.
Pour le dernier jeu de données, on procède de la même façon. Ainsi, pour ω_1, $\boldsymbol{x}_1^2 + (\boldsymbol{x}_2 - 3)^2 = -2\ln(2\pi K_2)$, on obtient alors un cercle de centre (0,3) de rayon $\sqrt{-2\ln(2\pi K_1)}$. Pour ω_2, $\boldsymbol{x}_1^2 + (\boldsymbol{x}_2 + 5)^2 = -2\ln(50\pi K_2)$, on obtient alors un cercle de centre (0,-5) de rayon $\sqrt{-2\ln(50\pi K_1)}$.

3.3 Règle de Bayes

La règle de Bayes, δ^* a pour but de minimiser la probabilité d'erreur $\epsilon(\delta|\boldsymbol{x})$. Nous avons ici deux classes seulement et pouvons l'exprimer de cette manière :

$$\delta^*(\boldsymbol{x}) = a_1 \Leftrightarrow \mathbb{P}(\omega_1|\boldsymbol{x}) > \mathbb{P}(\omega_2|\boldsymbol{x}) \tag{6}$$

$$\Leftrightarrow \frac{f_1(\boldsymbol{x})\pi_1}{f(\boldsymbol{x})} > \frac{f_2(\boldsymbol{x})\pi_2}{f(\boldsymbol{x})} \tag{7}$$

$$\Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} > \frac{\pi_2}{\pi_1} \tag{8}$$

Or $\pi_1 = \pi_2 = 0,5$ pour les premiers jeux de données, et nous sommes de plus dans un cas d'homoscédasticité. On a donc $\delta^*(\boldsymbol{x}) = \begin{cases} a_1 & \text{si } \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} > \frac{1}{2} \\ a_2 & \text{sinon} \end{cases}$, c'est-à-dire $\delta^*(\boldsymbol{x}) = \begin{cases} a_1 & \text{si } x_2 > \frac{1}{2} \\ a_2 & \text{sinon} \end{cases}$.
Pour le dernier jeu de données, on résoud l'équation :

$$\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} > \frac{1}{2} \tag{9}$$

$$\Leftrightarrow -\frac{1}{2}(\boldsymbol{x}_1^2 + (\boldsymbol{x}_2 - 3)^2) > ln(\frac{1}{20\pi}) - \frac{1}{2}(\boldsymbol{x}_1^2 + (\boldsymbol{x}_2 + 5)^2) \tag{10}$$

$$\Leftrightarrow (x_2 + 5)^2 - (x_2 - 3)^2 > 2\ln(\frac{1}{20\pi}) \tag{11}$$

Ce qui nous donne $\delta^*(\boldsymbol{x}) = \begin{cases} a_1 & \text{si } \boldsymbol{x}_2 > \sqrt{\frac{1}{2}\ln(\frac{1}{20\pi}) - 4} \\ a_2 & \text{sinon} \end{cases}$.
On remarque que pour le jeu Synth-1 on obtient une frontière de décision linéaire, alors qu'elle est quadratique pour le jeu Synth-2.

3.4 Frontières de décision

Nous avons tracé avec R les frontières de décision pour les quatre premiers jeux, grâce à a fonction *abline*. D'après les résultats obtenus grâce à la règle de Bayes, la frontière de décision entre ω_1 et ω_2 est une droite se trouvant à mi-chemin entre les deux classes, c'est-à-dire répondant à l'équation $X2 = 0.5$.

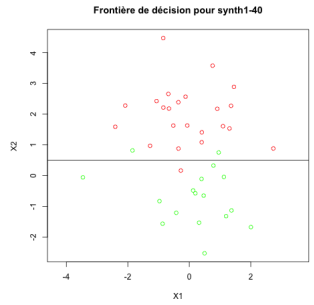

FIGURE 3 – Frontière de décision : synth1-500

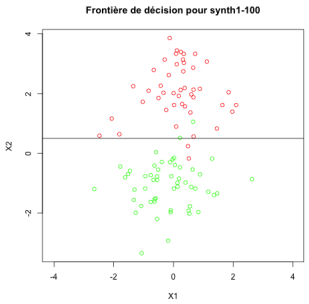

FIGURE 4 – Frontière de décision : synth1-1000

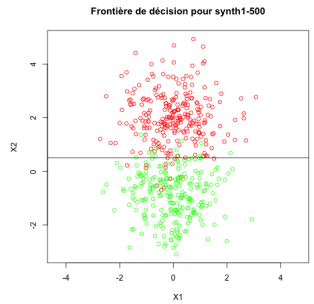

FIGURE 5 – Frontière de décision : synth1-500

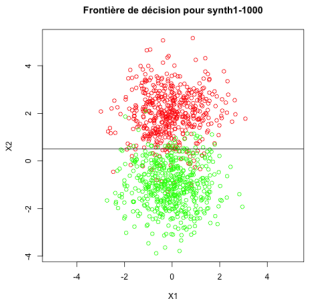

FIGURE 6 – Frontière de décision : synth1-1000

3.5 Erreur de Bayes

La probabilité d'erreur de Bayes est la plus petite erreur possible que peut atteindre une règle de décision utilisant uniquement la variable explicative X. Nous sommes ici dans un cas d'homoscédasticité. De plus, $\pi_1 = \pi_2$. L'erreur de Bayes s'écrit alors $\epsilon^* = \Phi(-\frac{\Delta}{2})$, sachant que $\Delta^2 = (\mu_2 - \mu_1)^T \Sigma^{-1} (\mu_2 - \mu_1)$ et que Φ est la fonction de répartition de la loi normale centrée réduite.
Dans notre cas, $\Delta^2 = 9$ donc $\epsilon^* = \Phi(-\frac{3}{2}) = 0.0668$.
On obtient une erreur relativement similaire à celles que nous avons pu trouver précédemment pour les premiers jeux de données, pour les ensembles de test. On note ici que les taux d'erreurs avec l'ensemble d'apprentissage sont cependant bien plus faibles, ce qui confirme notre intuition selon laquelle utiliser les individus d'apprentissage pour effectuer les tests est une méthode biaisée. Ces résultats sont "trop parfaits" au vu du taux d'erreur de Bayes.

4 Conclusion

Ce TP nous a permis dans un premier temps de nous familiariser avec deux classifieurs simples : le classifieur euclidien et la règle des K plus proches voisins (PPV). Ceux-ci nous ont servi à effectuer la discrimination de nouveaux individus à partir d'un groupe d'individu déjà connus et étiquetés. Nous avons ensuite calculé les différents taux d'erreur afin de voir si un des classifieurs se révèle plus fiable que l'autre. Nous avons vu que dans notre cas la règle des PPV s'avère plus fiable que le classifieur euclidien lorsqu'il s'agit d'un jeu de données important.
Nous avons ensuite pu faire le parallèle entre ces résultats expérimentaux et la théorie grâce à la règle de Bayes. Nous avons alors vu l'importance du découpage des données en individus d'apprentisage pour construire la règle de décision, différents des individus de test qui servent à l'évaluer, en comparant nos résultats avec la probabilité d'erreur de Bayes.

A Annexes

A.1 Classifieur euclidien

```
ceuc.app <- function(Xapp, zapp){#on calcule le centre de gravite des classes
#Xapp : napp*p, tableau individu-variables des individus d'apprentissage
#zapp : napp, étiquettes associées aux individus
#mu : g*p, params estimés du classifieur euclidien
mu <-matrix(,nrow=length(unique(zapp)),ncol(Xapp))
for (i in 1:(length(unique(zapp)))){
mu[i,]<-apply(Xapp[which(zapp==i),],2,mean)
}
mu
}
-----------------------------------------------
ceuc.val <- function(mu, Xtst) {
etiquette<-matrix(,nrow=nrow(Xtst),1)
distTab <- distXY(mu, Xtst)
for (i in 1:(nrow(Xtst))){
etiquette[i,] <- which.min(distTab[,i])
}
etiquette
}
```

A.2 K plus proches voisins

```
kppv.tune <- function(Xapp, zapp, Xval, zval, nppv) {
Xapp <- as.matrix(Xapp)
Xval <- as.matrix(Xval)
zapp <- as.vector(zapp)
zval <- as.vector(zval)
res = c(1:length(nppv)) #contiendra le taux d'erreur pour chaque valeur de ppv
for(i in 1:length(nppv)) { #Pour chaque valeur de ppv
ztst <- kppv.val(Xapp, zapp, i, Xval) #On cherche le vecteur des étiquettes
res[i] <- mean(abs(zval-ztst)) #On soustrait les deux vecteurs et on fait la moyenne de la val abs
}
res
}
-------------------------------------------
kppv.val<-function(Xapp,zapp,K,Xtst){
Xapp<-as.matrix(Xapp)
Xtst<-as.matrix(Xtst)
zapp <- as.vector(zapp)
napp <- nrow(Xapp) #nb indiv app
ntst <- nrow(Xtst) #nb indiv test

distance<-distXY(Xtst,Xapp) #distance indiv app-test
distance2<-t(apply(distance,1,sort)) #distance triée
```

```
dk <- distance2[,K] #kième distance

distk <- NULL #matrice kième distance
for(i in 1:napp) {
distk <- cbind(distk, dk)
}
comp <- distance<=distk #matrice booléenne des k indivs les + proches de chaque indiv test

matzapp <- NULL #matrice des étiquettes des indiv d'apprentissage
for(i in 1:ntst) {
matzapp <- rbind(matzapp, zapp)
}

ztst <- comp*matzapp #Ne contient plus que les étiquettes des k plus près indiv d'apprentissage
z <- NULL #retient l'étiquette en plus grand nombre
for(i in 1:ntst) {
if(length(which(ztst[i,]==1))< length(which(ztst[i,]==2)))
z[i] <- 2
else
z[i] <- 1
}
z
}
```