Discrimination, théorie bayésienne de la décision

YANQING ZENG GI05
MINH TRI LÊ GI02

5 juin 2017

Introduction

Dans ce Tp, nous allons appliquer des méthodes d'apprentissage supervisé. À partir d'un jeu de données, nous allons donc tenter de trouver un modèle pouvant prédire les classes d'appartenance à des individus.

Le modèle s'entraînera d'abord sur des données d'apprentissage où le résultat est connu. Puis, nous évaluerons la performance du modèle trouvé sur la qualité de la prédiction des données de test.

Les trois méthodes de classification que nous allons utiliser sont le classifieur euclidien, l'algorithme des K plus proches voisins et la règle de Bayes.

1 Classifieur euclidien, K plus proches voisins

1.1 Programmation

1.1.1 Classifieur euclidien

La fonction ceuc.app fait l'apprentissage des paramètres du classifieur euclidien. Autrement dit, il retourne les centres de chaque classe, sous la forme d'une matrice mu de dimensions g x p.

Elle a pour arguments d'entrée : Xapp (de dimensions napp x p) des individus d'apprentissage, zapp (de longueur napp) des étiquettes associées à chacun de ces individus.

```
ceuc.app<-function(Xapp,zapp){

      Xapp<-as.matrix(Xapp);
      p<-dim(Xapp)[2];#la dimension des variables X
      n<-dim(Xapp)[1];#le nombre des individus

      mu<-matrix(rep(0,p*2),2,p);#initialisation de mu

      for (g in 1:2){# g : le nombre de classes. (2 classes dans notre cas)
            x<-rep(0,p);# x stocke la somme des individus appartenant a cette classe
            n_g<-0;#n_g compte le nombre des individus appartenant a cette classe
            for (i in 1:n){
                  if( zapp[i]==g){
                      x<-x+Xapp[i,];
                      n_g<-n_g+1;
                  }
            }
            mu[g,]<-x/n_g;#la moyenne
      }

```

```
    mu;
}
```

La fonction ceuc.val retourne un vecteur (de longueur ntst) d'étiquettes prédites. Elle prend en arguments d'entrée : le tableau individus-variables Xtst (de dimension ntst x p) et la matrice mu des paramètres estimés.

```
ceuc.val<-function(mu,Xtst){

      Xtst<-as.matrix(Xtst);
      mu<-as.matrix(mu);# les centres de chaque classe

      n<-dim(Xtst)[1];# ntst
      z<-rep(0,n);# les etiquettes predites

#distance est une matrice stockant la distance entre les individus et les centres des
    classes. Par exemple, la premiere colonne est la distance entre les individus et la
    classe 1.

      distance<-distXY(Xtst,mu);

      for(i in 1:n){
          z[i]<-which.min(distance[i,]);  ; #on prend l'etiquette z qui selon la classe
              minimise la distance.
      }
      z;
}
```

1.1.2 K plus proches voisins

La fonction kppv.val fait le classement d'un tableau individus-variables Xtst en comptant des classes des K plus proches voisins.

Elle prend en argument les données étiquettes (Xapp, zapp), le nombre de voisins K et l'ensemble à évaluer Xtst.

```
kppv.val<-function(Xapp,zapp,K,Xtst){
 # initialisation
      Xapp<-as.matrix(Xapp);
      Xtst<-as.matrix(Xtst);
      n_tst<-dim(Xtst)[1];
      n_app<-dim(Xapp)[1];

      d_index<-rep(0,n_app);#d_index est un vecteur pour stocker les index des k plus
          proches voisins
      g_count<-rep(0,2);#g_count est un vecteur de longueur 2 pour compter les classes
          pour les K plus proches voisins

      z<-rep(0,n_tst);

      for(i in 1:n_tst){

            distance<-distXY(Xtst,Xapp);# la matrice de distance

            d_index<-order(distance[i,])[1:K];#trier et trouver les K plus proche voisin
                pour l'individu i
```

```
            g_count[1]<-table(zapp[d_index])['1'];
            g_count[2]<-table(zapp[d_index])['2'];
            g_count[is.na(g_count)]<-0;

            if( g_count[1] >g_count[2] ){#la fonction de decision
                  z[i]<-1;
            }
            else{
                  z[i]<-2;
            }
      }
      z;
}
```

La fonction kppv.tune détermine le nombre optimal de voisins K parmi un vecteur nppv. Elle prend en argument : les données étiquettes (Xapp et zapp), les individus-variables Xval, un vecteur zapp pour la validation, et un ensemble de valeurs nppv correspondantes aux différents nombres de voisins à tester.

```
kppv.tune<-function(Xapp,zapp,Xval,zval,nppv){
      Xapp<-as.matrix(Xapp);
      Xval<-as.matrix(Xval);
      ztst<-rep(0,length(zval));
      n_k<-rep(0,length(nppv));#n_k compte le nombre de predictions correctes.

      for(k in length[nppv]){#k est l'iterateur du vecteur nppv
            ztst<-kppv.val(Xapp,zapp,nppv[k],Xval);
            for(i in length(zval)){
                  if (zval[i]==ztst[i]){
                        n_k[k]<-n_k[k]+1;
                  }
            }
      }
      nppv[which.max(n_k)];#on retourne le nombre de K qui maximise le nombre de
          predictions correctes.
}
```

1.2. Évaluation des performances

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

1. Estimation des paramètres μ_k, $\sum_k$ et π_k

	μ_k	$\sum_k$	π_k
Classe 1	$\begin{pmatrix}-0.0316\\1.0920\end{pmatrix}$	$\begin{pmatrix}0.6438 & 0.1127\\0.0.1127 & 0.9566\end{pmatrix}$	0.45
Classe 2	$\begin{pmatrix}-1.8834\\0.1051\end{pmatrix}$	$\begin{pmatrix}1.3127 & 0.3081\\0.3081 & 1.3739\end{pmatrix}$	0.55

(a) Synth1-40 (arrondi à 10^{-4})

	μ_k	$\sum_k$	π_k
Classe 1	$\begin{pmatrix}-0.0257\\0.8153\end{pmatrix}$	$\begin{pmatrix}0.8653 & -0.1288\\-0.1288 & 1.0883\end{pmatrix}$	0.54
Classe 2	$\begin{pmatrix}-1.965422\\-0.1265\end{pmatrix}$	$\begin{pmatrix}0.7405 & -0.0368\\-0.0368 & 0.7445\end{pmatrix}$	0.46

(b) Synth1-100 (arrondi à 10^{-4})

	μ_k	$\sum_k$	π_k
Classe 1	$\begin{pmatrix}0.1316\\0.8797\end{pmatrix}$	$\begin{pmatrix}1.0467 & 0.0520\\0.0520 & 0.9801\end{pmatrix}$	0.528
Classe 2	$\begin{pmatrix}-1.8834\\-0.0848\end{pmatrix}$	$\begin{pmatrix}0.9628 & -0.1104\\-0.1104 & 0.9753\end{pmatrix}$	0.472

(c) Synth1-500 (arrondi à 10^{-4})

	μ_k	$\sum_k$	π_k
Classe 1	$\begin{pmatrix}-0.0128\\0.9157\end{pmatrix}$	$\begin{pmatrix}0.9666 & -0.0651\\-0.651 & 1.0773\end{pmatrix}$	0.496
Classe 2	$\begin{pmatrix}-1.9612\\0.0183\end{pmatrix}$	$\begin{pmatrix}0.9885 & 0.0209\\0.0209 & 0.9394\end{pmatrix}$	0.504

(d) Synth1-1000 (arrondi à 10^{-4})

TABLE 1 – Estimation des paramètres μ_k, $\sum_k$ et π_k pour les jeux de données Synth1-n

2. Calcul du taux d'erreur d'apprentissage ε et de test (N=20 séparations aléatoires) avec le classifieur euclidien

Pour calculer le taux d'erreur ponctuelle $\widehat{\varepsilon}$ d'une prédiction, il faut simplement compter le nombre de prédictions erronées ε_i entre la prédiction faite et les données réelles (d'apprentissage ou de test), puis diviser par le nombre total d'individus n.

On répète cette procédure N fois pour obtenir 20 réalisations du taux d'erreur. Pour obtenir l'estimation du taux d'erreur $\overline{\varepsilon}$, il ne reste plus qu'à faire la moyenne des 20 réalisations.
L'intervalle de confiance à 95% est obtenu comme-ci : $IC = \left[\mu - 1.96 * \frac{s^*}{\sqrt{N}};\ \mu + 1.96 * \frac{s^*}{\sqrt{N}}\right]$.

Avec :

— μ : La moyenne des N réalisations de ε

— s^* : L'écart-type corrigé des N réalisations de ε

— N : Le nombre de réalisations de ε

Données	$\overline{\varepsilon}$	IC
Apprentissage	0.2037	[0.1830; 0.2244]
Test	0.2500	[0.2108; 0.2892]

(a) Synth1-40 (arrondi à 10^{-4})

Données	$\overline{\varepsilon}$	IC
Apprentissage	0.0852	[0.0815; 0.0889]
Test	0.1042	[0.0971; 0.1114]

(b) Synth1-100 (arrondi à 10^{-4})

Données	$\overline{\varepsilon}$	IC
Apprentissage	0.1326	[0.1310; 0.1342]
Test	0.1335	[0.1305; 0.1366]

(c) Synth1-500 (arrondi à 10^{-4})

Données	$\overline{\varepsilon}$	IC
Apprentissage	0.1390	[0.1360; 0.1420]
Test	0.1470	[0.1405; 0.1535]

(d) Synth1-1000 (arrondi à 10^{-4})

TABLE 2 – Calcul du taux d'erreur d'apprentissage ε et de test pour les jeux de données Synth1-n avec le classifieur euclidien

3. Détermination du nombre de K voisins optimaux

Si K=1 voisin est compris dans l'intervalle du nombre de voisins à tester alors le nombre de voisins optimal vaut 1 pour tous les jeux de données **Synth1-*n***.
Dans ce cas, l'algorithme des K plus proches voisin est en surapprentissage. Donc le K obtenu est très précis pour ce jeu de données, mais testé sur d'autres données, ce résultat sera beaucoup moins précis et donc moins robuste dans le cas général.

```
kppv.tune(Xapp, zapp, Xapp, zapp,(1:10)) # Avec K compris dans [1;10]
```

Si K=1 n'est pas compris dans l'intervalle de nombre de voisins à tester alors le nombre de voisins optimal vaut généralement 3 pour tous les jeux de données **Synth1-*n***.

```
kppv.tune(Xapp, zapp, Xapp, zapp,(2:10)) # Avec K compris dans [2;10]
```

4. Calcul du taux d'erreur d'apprentissage ε et de test (N=20 séparations aléatoires) avec les K plus proches voisins

On procède de la même manière qu'en 1 pour calculer le taux d'erreur.
On exclura K=1 des possibilités pour les raisons mentionnées précédemment.

Données	$\bar{\varepsilon}$	IC
Apprentissage	0.1552	[0.1453; 0.1652]
Test	0.2765	[0.2588; 0.2942]

(a) Synth1-40 (arrondi à 10^{-4})

Données	$\bar{\varepsilon}$	IC
Apprentissage	0.0720	[0.0601; 0.0839]
Test	0.1083	[0.0866; 0.1300]

(b) Synth1-100 (arrondi à 10^{-4})

Données	$\bar{\varepsilon}$	IC
Apprentissage	0.1012	[0.0928; 0.1096]
Test	0.1732	[0.1599; 0.1865]

(c) Synth1-500 (arrondi à 10^{-4})

Données	$\bar{\varepsilon}$	IC
Apprentissage	0.1049	[0.1009; 0.1089]
Test	0.1812	[0.1711; 0.1912]

(d) Synth1-1000 (arrondi à 10^{-4})

TABLE 3 – Calcul du taux d'erreur d'apprentissage ε et de test pour les jeux de données Synth1-n avec les K plus proches voisins

En comparant la méthode du classifieur euclidien et celle des K plus proches voisins, on observe tout d'abord que l'algorithme des K plus proches voisins est plus précis que le classifieur euclidien sur toutes les données d'apprentissage.
Cependant, la performance de la prédiction sur les données d'apprentissage et de test pour le classifieur euclidien est assez similaire contrairement à l'algorithme des K plus proches voisins. Il est donc plus robuste que l'algorithme des K plus proches voisins, ce dernier étant moins stable dans ses prédictions.
La classifieur euclidien est ainsi plus précis sur les données de test.

Dans l'ensemble, les résultats pour les deux méthodes tendent à diminuer quand le nombre d'individus augmente, ce qui est cohérent. La performance de ces deux algorithmes est correcte et avoisine les 90% de précision sur les données d'apprentissage (avec *n>=100*. Cela descend jusqu'à 80% de précision pour les données de test.

1.2.2 Jeu de données Synth2-1000

1 Estimer les paramètres μ_k et $\sum_k$ ainsi que les proportions π_k des classes

	μ_k	$\sum_k$	π_k
Classe 1	$\begin{pmatrix} 3.018387510 \\ -0.006382269 \end{pmatrix}$	$\begin{pmatrix} 0.9885089 & 0.1129223 \\ 0.1129223 & 1.0907809 \end{pmatrix}$	0.523
Classe 2	$\begin{pmatrix} -2.14228117 \\ -0.02652448 \end{pmatrix}$	$\begin{pmatrix} 0.9885089 & 0.1129223 \\ 0.1129223 & 1.0907809 \end{pmatrix}$	0.477

TABLE 4 – Estimation des paramètres μ_k, $\sum_k$ et π_k pour les jeux de données Synth2-1000

2 Calculer les estimations (ponctuelles et intervalles de confiance) de ε

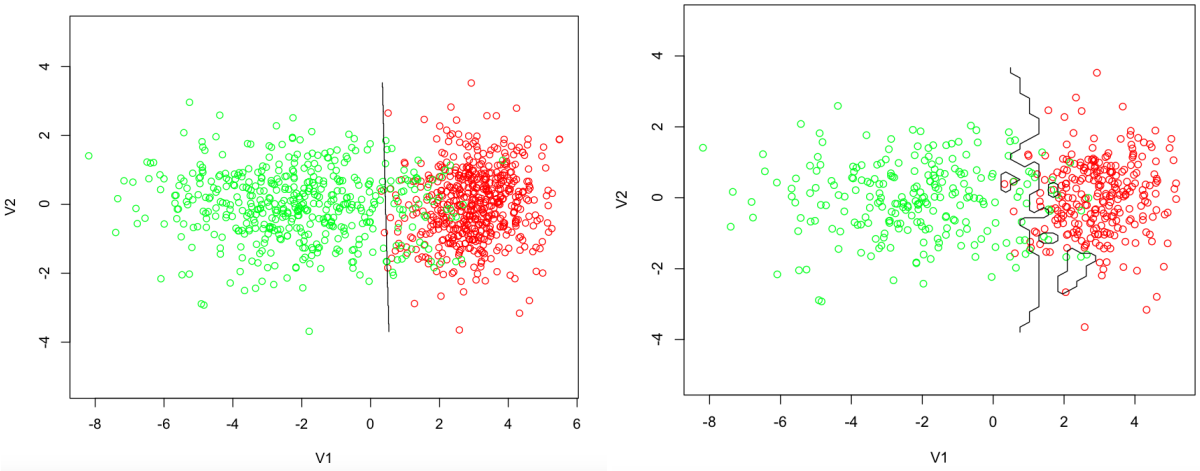

FIGURE 1 – Le classifieur euclidien et le classifieur des k plus proches voisins

Pour les Kppv sur les données d'apprentissage, on a aussi exclu le K=1 voisin pour éviter l'overfitting. En effet, pour obtenir le nombre de K voisins optimaux, on peut appliquer la méthode K-fold validation croisée minimisant l'erreur sans introduire l'overfitting. C'est-à-dire qu'on sépare les données en K ensembles, et à chaque itération, on prend (K-1) parties pour l'apprentissage et une autre pour le test. On prend les paramètres qui minimise l'erreur totale de ces K itérations.

	Erreur ponctuelle	Écart-type de ε	Intervalle de confiance
Classifieur euclidien apprentissage	0.06364318	0.005491928	[0.0528792,0.07440716]
Classifieur euclidien test	0.06261261	0.01064066	[0.0417573,0.08346792]
Kppv apprentissage	0.0488	0.007743248	[0.03362351,0.06397649]
Kppv test	0.0816	0.01313252	[0.05586073,0.1073393]

TABLE 5 – Calcul du taux d'erreur d'apprentissage ε et de test pour les jeux de données Synth2-1000 avec le classifieur euclidien et Kppv

On en déduit que la variance de l'erreur pour les données tests est généralement plus élevée que celui des données d'apprentissage, ainsi que l'erreur ponctuelle. L'erreur d'apprentissage pour

le classifieur vaut 0 quand K=1, parce que les données correspondent parfaitement au modèle. Par contre, l'overfitting introduit une erreur plus élevée sur les données tests.

Pour ce jeu de données, les deux classifieurs ont une très bonne performance (à condition que les proportions des deux classes soient proches), grâce au nombre d'individus d'apprentissage qui est assez grand.

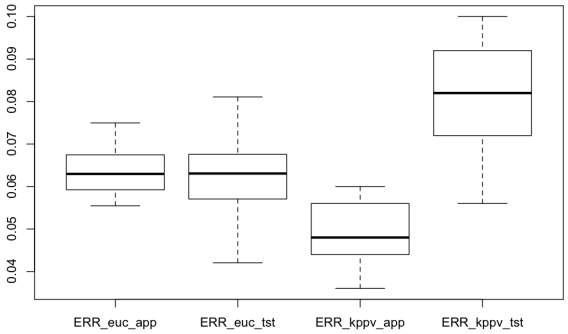

FIGURE 2 – Distribution de l'erreur

1.2.3 Jeux de données réelles

a. Pima

	Données	$\bar{\varepsilon}$	IC
Classifieur euclidien	Apprentissage	0.2461	[0.2430; 0.2491]
	Test	0.2470	[0.2420; 0.2521]
Kppv	Apprentissage	0.1476	[0.1392; 0.1559]
	Test	0.2707	[0.2523; 0.2891]

TABLE 6 – Calcul du taux d'erreur d'apprentissage ε et de test pour le jeu de données Pima avec la méthode du classifieur euclidien et les K plus proches voisins (arrondi à 10^{-4})

Comme avec les précédents jeux de données, le classifieur euclidien permet d'obtenir des résultats très similaires entre les données d'apprentissage et de test, contrairement à l'algorithme des K plus proches voisins.
Cependant, à défaut d'être robuste, la méthode des K plus proches voisins permet d'atteindre une précision plus élevée sur les données d'apprentissage.

Enfin, les résultats sur les données de test sont similaires, avec un léger avantage pour le classifieur euclidien. Ils sont de l'ordre de 75% de précision.

b. Breastcancer

	Données	$\bar{\varepsilon}$	IC
Classifieur euclidien	Apprentissage	0.0391	[0.0384; 0.0399]
	Test	0.0403	[0.0388; 0.0418]
Kppv	Apprentissage	0.0182	[0.0158; 0.0216]
	Test	0.0356	[0.0320; 0.0392]

TABLE 7 – Calcul du taux d'erreur d'apprentissage ε et de test pour le jeu de données Breastcancer avec la méthode du classifieur euclidien et les K plus proches voisins (arrondi à 10^{-4})

Les prédictions faites par les deux méthodes sont très performantes, que ce soit avec les données d'apprentissage ou de test. La précision de la prédiction est supérieure à 96%.
Même remarque que précédemment, les résultats obtenus par les K plus proches voisins varient plus entre les données d'apprentissage et de test comparé au classifieur euclidien.

Enfin, les prédictions sont légèrement plus précises pour l'algorithme des K plus proches voisins qu'avec le classifieur euclidien.

2 Règle de Bayes

1. Distributions marginales des variables $X^1 et X^2$ dans chaque classe

Les individus $n1$ et $n2$ ont été générés dans chaque classe par une loi normale bivariée de paramètres μ_k et $\sum_k$.

μ_k est un vecteur moyenne de la classe k de chaque variable. On note μ_{ki} la moyenne de la variable i de la classe k.
$\sum_k$ est la matrice de covariance de la classe k. On note $\sum_{k_{ij}}$ la covariance des variables i et j dans la classe k.

On a donc : dans la classe k, $X^1 \sim N(\mu_{k1}, \sum_{k_{11}})$ et $X^2 \sim N(\mu_{k2}, \sum_{k_{22}})$. On remarque que les matrices de covariance pour **Synth1-*n*** et **Synth2-1000** sont diagonales donc les variables X^1 et X^2 sont indépendantes l'une de l'autre dans chaque classe.

Enfin, on obtient les lois marginales suivantes :

- **Synth1-*n***
  - Dans la classe $k = 1$:
    - $X^1 \sim N(0, 1)$
    - $X^2 \sim N(1, 1)$
  - Dans la classe $k = 2$:
    - $X^1 \sim N(-2, 1)$
    - $X^2 \sim N(0, 1)$
- **Synth2-1000**
  - Dans la classe $k = 1$:
    - $X^1 \sim N(3, 1)$
    - $X^2 \sim N(0, 1)$
  - Dans la classe $k = 2$:
    - $X^1 \sim N(-2, 5)$
    - $X^2 \sim N(0, 1)$

2. Courbes d'iso-densité

Soit f_k la distribution de la classe k. Comme les variables X^1 et X^2 sont indépendantes alors f_k=$\prod_{i=1}^{n} f_k i$ où f_{ki} est la densité de la variable i de la classe k.

Donc f_1 la distribution de la classe 1 vaut : $f_1 = f_{11} * f_{12}$ où f_{11} est la distribution de la variable 1 dans la classe 1.

Les courbes d'iso densité de chaque classe correspondent aux espaces où les variables X^i ont les mêmes densités.
On pose donc $f_k(x) = k$ où k est une constante.

— **Synth1-*n***

— $f_1(x) = \frac{1}{2\pi}\ exp\left(-\frac{1}{2}(x_1^2 + (x_2-1)^2)\right)$
$f_1(x) = k \Longrightarrow x_1^2 + (x_2-1)^2 = -2ln(2\pi k) \Longrightarrow$ Cercle de centre $(0,1)$ de rayon $\sqrt{-2ln(2\pi k)}$

— $f_2(x) = \frac{1}{2\pi}\ exp\ \left(-\frac{1}{2}((x_1+2)^2 + x_2^2)\right)$
$f_2(x) = k \Longrightarrow (x_1+2)^2 + x_2^2 = -2ln(2\pi k) \Longrightarrow$ Cercle de centre $(-2,0)$ de rayon $\sqrt{-2ln(2\pi k)}$

— **Synth2-1000**

— $f_1(x) = \frac{1}{2\pi}\ exp\ \left(-\frac{1}{2}((x_1-3)^2 + x_2^2)\right)$
$f_1(x) = k \Longrightarrow (x_1-3)^2 + x_2^2 = -2ln(2\pi k) \Longrightarrow$ Cercle de centre $(3,0)$ de rayon $\sqrt{-2ln(2\pi k)}$

— $f_2(x) = \frac{1}{2\pi\sqrt{5}}\ exp\ \left(-\frac{1}{10}((x_1+2)^2 + 5x_2^2)\right)$
$f_2(x) = k \Longrightarrow (x_1+2)^2 + 5x_2^2 = -10ln(\sqrt{5})ln(2\pi k)$
$\Leftrightarrow \frac{(x_1+2)^2}{-10ln(\sqrt{5})ln(2\pi k)} + \frac{x_2^2}{-2ln(\sqrt{5})ln(2\pi k)} = 1 \Longrightarrow$ Ellipse de centre $(-2,0)$ de grand rayon $\sqrt{-10ln(\sqrt{5})ln(2\pi k)}$ et de petit rayon $\sqrt{-2ln(\sqrt{5})ln(2\pi k)}$

3. Règle de Bayes pour les jeux de données Synth1-*n*

Pour $g = 2$ classes, la règle de Bayes est :

$$\delta(x) = \omega_1 \Leftrightarrow \frac{f_1(x)}{f_2(x)} > \frac{\pi_2}{\pi_1}$$

Soit :

$$\frac{f_1(x)}{f_2(x)} = \frac{\frac{1}{2\pi}\ exp\left(-\frac{1}{2}(x_1^2 + (x_2-1)^2)\right)}{\frac{1}{2\pi}\ exp\ \left(-\frac{1}{2}((x_1+2)^2 + x_2^2)\right)} > \frac{\pi_2}{\pi_1}$$

$$\Leftrightarrow exp\left(\frac{1}{2}(-x_1^2 + (x_1+2)^2 - (x_2-1)^2 + x_2^2\right) > \frac{\pi_2}{\pi_1}$$

$$\Leftrightarrow exp\left(2x_1 + x_2 + \frac{3}{2}\right) > \frac{\pi_2}{\pi_1}$$

$$\Leftrightarrow 2x_1 + x_2 > ln(\frac{\pi_2}{\pi_1}) - \frac{3}{2}$$

En posant $k = ln(\frac{\pi_2}{\pi_1}) - \frac{3}{2}$, on a donc :

$$2x_1 + x_2 > k$$

La règle de Bayes est donc bien une fonction linéaire de x_1, x_2 et k.
Comme $\pi_1 = \pi_2 = 0.5$, $k = ln(1) - \frac{3}{2} = -\frac{3}{2}$.
Finalement, la frontière de décision est

$$2x_1 + x_2 > -\frac{3}{2}$$

$$\Leftrightarrow x_2 > -2x_1 - \frac{3}{2}$$

On classifie un individu dans la classe 1 quand $x_2 > -2x_1 - \frac{3}{2}$.

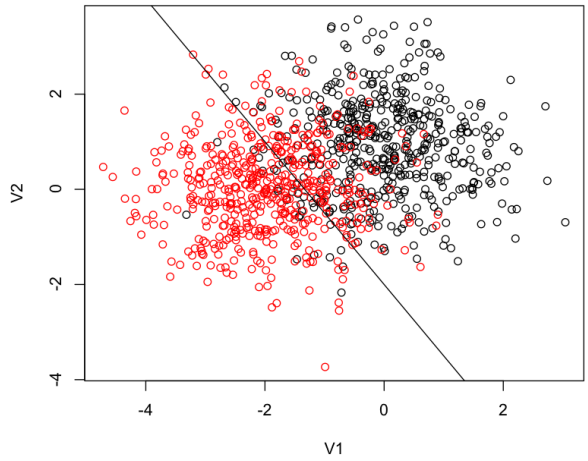

FIGURE 3 – Règle de Bayes pour le jeu de données **Synth1-1000**

4. Règle de Bayes pour le jeu de données Synth2-1000

$$\delta^*(x) = \begin{cases} 1 & si \quad P(w_2|x) < P(w_1|x) \\ 2 & sinon \end{cases}$$

$$
\begin{aligned}
\delta^* = 1 &\Leftrightarrow \frac{f_1(x)}{f_2(x)} > \frac{\pi_2}{\pi_1} \\
&\Leftrightarrow \frac{\frac{1}{2\pi} exp(-\frac{(x_1-3)^2}{2} - \frac{x_2^2}{2})}{\frac{1}{2\pi\sqrt{5}} exp(-\frac{(x_1+2)^2}{10} - \frac{x_2^2}{2})} > \frac{\pi_2}{\pi_1} \\
&\Leftrightarrow \frac{(x_1+2)^2}{10} - \frac{(x_1-3)^2}{2} > ln(\frac{\pi_2}{\pi_1}) - ln\sqrt{5} \\
&\Leftrightarrow 4x_1^2 - 34x_1 + 41 < -10ln(\frac{\pi_2}{\pi_1\sqrt{5}}) \\
&\Leftrightarrow (x_1 - \frac{17}{4})^2 < -40ln(\frac{\pi_2}{\pi_1\sqrt{5}}) + 125 \\
&\Leftrightarrow \frac{17}{4} - \frac{\sqrt{125 - 40ln(\frac{\pi_2}{\pi_1\sqrt{5}})}}{4} < x_1 < \frac{17}{4} + \frac{\sqrt{125 - 40ln(\frac{\pi_2}{\pi_1\sqrt{5}})}}{4}
\end{aligned}
$$

Donc la frontière de décision est bien délimitée par deux constantes $k1$ et $k2$: $k_1 < x_1 < k_2$.

Avec la probabilité à priori $\pi_1 = \pi_2 = 0.5$, on trouve alors $1.115626476 < x_1 < 7.384373524$. La frontière de décision est illustrée par la figure 4.

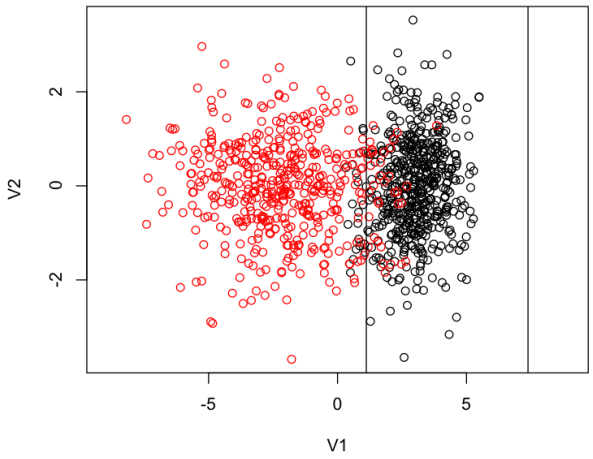

FIGURE 4 – Règle de Bayes pour le jeu de données **Synth2-1000**

5. Expression formelle de l'erreur de Bayes

La probabilité d'erreur de Bayes est la probabilité de mal classifier un individu.
Soit :

$$
\begin{aligned}
\varepsilon(\sigma) &= \int_{R_1} \varepsilon(\delta|x) f(x) dx + \int_{R_2} \varepsilon(\delta|x) f(x) dx \\
&= \int_{R_1} P(w_2|x) f(x) dx + \int_{R_2} P(w_1|x) f(x) dx \\
&= \int_{R_1} \pi_2 f_2(x) dx + \int_{R_2} \pi_1 f_1(x) dx
\end{aligned}
$$

a. Expression formelle de l'erreur de Bayes pour le jeu de données Synth1-n

On rappelle que :

$$\delta^*(x) = \begin{cases} 1 & si \quad x_2 > -2x_1 - k \; avec \; k = ln(\frac{\pi_2}{\pi_1}) - \frac{3}{2} \\ 2 & sinon \end{cases}$$

$$\varepsilon(\sigma) = \int_{R_2} \pi_1 f_1(x)dx + \int_{R_1} \pi_2 f_2(x)dx$$

$$\varepsilon(\sigma) = \int_{-\infty}^{-2x_1-\frac{3}{2}} \int_{-\infty}^{+\infty} \pi_1 f_1(x)dx_1dx_2 + \int_{-2x_1-\frac{3}{2}}^{+\infty} \int_{-\infty}^{+\infty} \pi_2 f_2(x)dx_1dx_2$$

D'une part,

$$\int_{-\infty}^{-2x_1-\frac{3}{2}} \int_{-\infty}^{+\infty} \pi_1 f_1(x)dx_1dx_2$$

$$\pi_1 \int_{-\infty}^{+\infty} \int_{-\infty}^{-2x_1-\frac{3}{2}} \frac{1}{2\pi} exp(-\frac{1}{2}(x_1^2 + (x_2-1)^2))dx_2dx_1$$

$$= \pi_1 \int_{-\infty}^{+\infty} \frac{e^{-\frac{x_1^2}{2}} \; erfc(\frac{4x_1+5}{2\sqrt{2}})}{2\sqrt{2\pi}} \; dx_2 \approx 0.131776 * \pi_1$$

Où *erfc* est la fonction complémentaire de l'erreur (voir Annexe 2).

D'autre part,

$$\int_{-2x_1-\frac{3}{2}}^{+\infty} \int_{-\infty}^{+\infty} \pi_1 f_1(x)dx_1dx_2$$

$$= \pi_2 \int_{-\infty}^{+\infty} \int_{-2x_1-\frac{3}{2}}^{\infty} \frac{1}{2\pi} exp(-\frac{1}{2}((x_1+2)^2 + x_2^2))dx_2dx_1$$

$$= \pi_2 \int_{-\infty}^{+\infty} \frac{e^{-\frac{(x_1+2)^2}{2}}(erf(\frac{4x_1+3}{2\sqrt{2}}+1))}{2\sqrt{2\pi}} \; dx_2 = 0.131776 * \pi_2$$

Donc l'erreur formelle est

$$\varepsilon(\sigma) = 0.131776 * \pi_1 + 0.131776 * \pi_2$$

$$= 0.131776 \; (pour \; \pi_1 = \pi_2 = 0.5)$$

La valeur effective pour Synth1-1000 est proche de l'erreur formelle.
On a $\pi_1 = 0.496$ et $\pi_2 = 0.504$. Soit au total, la même valeur que l'erreur formelle.

b. L'expression formelle de l'erreur de Bayes pour le jeu de données Synth2-1000

$$\delta^*(x) = \begin{cases} 1 & si \quad k_1 < x < k_2 \\ 2 & sinon \end{cases}$$

$$\begin{aligned} \varepsilon(\sigma) &= \int_{R_1} P(w_2|x)f(x)dx + \int_{R_2} P(w_1|x)f(x)dx \\ &= \int_{-\infty}^{k1} \pi_1 f_1(x)dx + \int_{k_1}^{k2} \pi_2 f_2(x)dx + \int_{k2}^{\infty} \pi_1 f_1(x)dx \\ &= \pi_1 * \Phi(k_1 - 3) + \pi_2 * \Phi(\frac{k_2+2}{5}) - \pi_2 * \Phi(\frac{k_1+2}{5}) + \pi_1 * \Phi(-k_2 + 3) \\ &\simeq 0.1330483, \qquad avec \quad k1 = 1.115627, \quad k2 = 7.384374 \end{aligned}$$

On remarque que l'erreur trouvé pour Synth1-1000 et Synth2-1000 sont assez proches (à 10^{-1} près).

La valeur effective pour Synth2-1000 est $\varepsilon(\sigma) \simeq 0.1282971$ avec $\pi_1 = 0.523$ et $\pi_2 = 0.477$.

Les résultats d'erreurs formelles et effectives sont donc cohérents pour les jeux de données Synth1-1000 et Synth2-1000 avec résultats obtenus par la règle de Bayes figure 3 et 4.

Pour Synth1-1000 : En comparant le taux d'erreur de la règle de Bayes avec le classifieur euclidien et K plus proches voisins, on remarque que la performance est proche du classifieur euclidien et des Kppv, mais avec un léger avantage pour la règle de Bayes.

Pour Synth2-1000 : Le classifieur euclidien et les Kppv ont des résultats très performants par rapport à la règle de Bayes.

Conclusion

Ce TP nous a permis de mettre en application les méthodes de bases de l'apprentissage supervisé, de l'apprentissage des paramètres à l'évaluation des performances.

Cela nous a aussi permis de comprendre les avantages et inconvénients de ces différentes méthodes afin de pouvoir choisir par la suite, choisir l'algorithme approprié au problème.

Références

[1] Error function

Annexes

Erfc : Fonction d'erreur complémentaire

$$erfc(x) = 1 - erf(x)$$
$$= \frac{2}{\sqrt{\pi}} \int_x^{+\infty} exp(-r^2)dt$$

Et,

$$erf(x) = 2\phi(x\sqrt{2}) - 1$$

Donc

$$erfc(x) = 2 - 2\phi(x\sqrt{2})$$