SY09 TP3
Discrimination, théorie bayésienne de la décision

Thibault Subreville & Yuting Chen

P17

L'objectif de ce TP est d'apprendre à manipuler le classifieur euclidien et le classifieur des K plus proches voisins sur différentes données binaires. Nous allons ensuite étudier les performances théoriques des outils utilisés en utilisant la règle de Bayes. L'outil d'analyse statistique utilisé est le logiciel R.
N.B.: Toutes les valeurs données dans ce rapport seront arrondies à 10^{-2} près et les intervalles de confiance seront donnés au seuil de 95% sauf mention contraire.

1 Classifieur euclidien, K plus proches voisins

Dans cette partie, on souhaite étudier la performance du classifieur euclidien et des K plus proches voisins sur un jeu de données binaires, i.e. sur une population constituée de 2 classes (g=2).

1.1 Programmation

1.1.1 Classifieur euclidien

On construit les fonctions **ceuc.app** qui réalise l'apprentissage des paramètres du classifieur euclidien et **ceuc.val** qui réalise le classement d'un tableau individus-variables et retourner un vecteur d'étiquettes prédites.
N.B.: Les fonctions suivantes fonctionnent pour un nombre quelconques de classes ($g \geq 2$).

```
#Nous supposerons pour la fonction suivante que
#les classes sont décrites par une variable numérique commençant par 1
ceuc.app <- function(Xapp, zapp){
  Xapp=as.matrix(Xapp)
  zapp=as.vector(zapp)

  g=unique(zapp)
  mu=matrix(nrow=length(g), ncol=dim(Xapp)[2])

  for(i in g){
    mu[i,]=apply(Xapp[which(zapp==i),],2,mean)
  }
  mu
}
```

Figure 1: La fonction **ceuc.app**

```
ceuc.val <- function(mu, Xtst){
  Xtst=as.matrix(Xtst)
  dist=distXY(mu,Xtst)
  etiquette=apply(dist,2,which.min)
  etiquette
}
```

Figure 2: La fonction **ceuc.val**

1.1.2 K plus proches voisins

On construit les fonctions **kppv.val()** qui réalise le classement d'un tableau individus-variables et retourne un vecteur d'étiquettes prédites et **kppv.tune()** qui détermine détermine le nombre "optimal" de voisins K_{opt}.
N.B.: Les fonctions suivantes fonctionnent pour un nombre quelconques de classes ($g \geq 2$).

```
kppv.val = function(Xapp, zapp, k, Xtst) {
  Xapp = as.matrix(Xapp);
  Xtst = as.matrix(Xtst);
  zapp = as.matrix(zapp);
  napp=dim(Xtst)[1]

  predit = matrix(0, napp, 1);
  dist = distXY(Xtst, Xapp);
  resultat = matrix(0, napp, k);
  for (i in 1:napp) {
    voisin = order(dist[i,])[1:k];
    resultat[i,] = zapp[voisin,1];
    temp = sort(table(resultat[i,]),decreasing=TRUE);
    predit[i,] = as.numeric(names(temp)[1]);
  }
  predit<-as.vector(predit)
  predit
}
```

Figure 3: La fonction **kppv.val**

```
kppv.tune = function(Xapp, zapp, Xval, zval, nppv) {
  error = matrix(0,2,length(nppv));
  nb = 1;
  for (k in nppv) {
    predit = kppv.val(Xapp,zapp,k,Xval);
    error[1,nb] = k;
    error[2,nb] = mean(predit!=zval);
    nb = nb+1;
  }
  t=min(which(error[2,] == min(error[2,])))
  return(error[1,t]);
}
```

Figure 4: La fonction **kppv.tune**

1.1.3 Tests des fonctions

Les figures suivantes permettent de visualiser la capacité des classifieurs à bien classer les données. La méthode est de construire un classifieur à l'aide de l'ensemble d'apprentissage pour prédire la classe de chaque individu de l'ensemble de test/validation en utilisant le modèle construit, puis afficher l'ensemble de test avec la frontière de décision.
Pour le test des fonctions, nous avons utiliser le jeu de données disponible dans le fichier **Synth1-40.csv**.
En utilisant les deux classifieurs programmés précédemment, nous obtenons les figures suivantes:

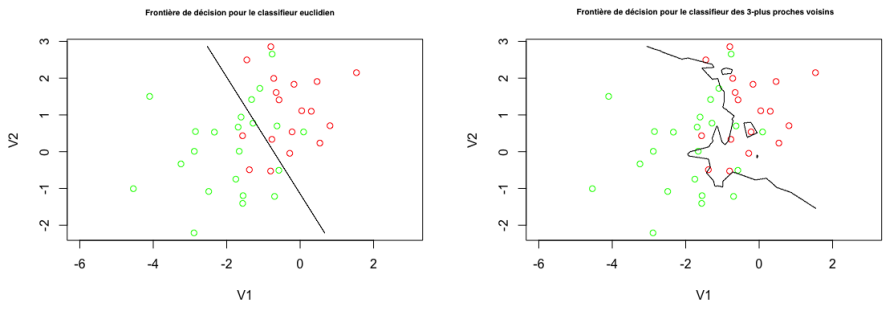

Figure 5: Frontières de décision avec le jeu de donnée *Synth1-40.csv*

Nous obtenons les mêmes figures que dans la partie 1.3 de l'énoncé.

1.2 Évaluation des performances

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

1. Pour chacun des jeux de données, nous allons estimer les paramètres μ_k (espérance) et Σ_k (covariance) des distributions conditionnelles, ainsi que les proportions π_k des deux classes.
En effectuant cela sur les jeux de données, nous obtenons les résultats suivants pour la classe 1:

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
Espérance μ_k	(-0.32,1.09)	(0.03,0.82)	(0.13,0.88)	(-0.01,0.92)
Covariances Σ_k	$\begin{pmatrix} 0.68 & 0.12 \\ 0.12 & 1.01 \end{pmatrix}$	$\begin{pmatrix} 0.88 & -0.13 \\ -0.13 & 1.11 \end{pmatrix}$	$\begin{pmatrix} 1.05 & 0.05 \\ 0.05 & 0.98 \end{pmatrix}$	$\begin{pmatrix} 0.97 & -0.07 \\ -0.07 & 1.08 \end{pmatrix}$
Proportions π_k	0.45	0.54	0.528	0.496

Et les résultats suivants pour la classe 2 :

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
Espérance μ_k	(-1.88,0.11)	(-1.97,-0.12)	(-1.88,-0.08)	(-1.96,0.02)
Covariances Σ_k	$\begin{pmatrix} 1.38 & 0.32 \\ 0.32 & 1.44 \end{pmatrix}$	$\begin{pmatrix} 0.76 & 0.04 \\ 0.04 & 0.76 \end{pmatrix}$	$\begin{pmatrix} 0.97 & -0.11 \\ -0.11 & 0.98 \end{pmatrix}$	$\begin{pmatrix} 0.99 & 0.02 \\ 0.02 & 0.94 \end{pmatrix}$
Proportions π_k	0.55	0.46	0.472	0.504

2. Pour évaluer les performances des deux méthodes, nous allons programmer des scripts permettant l'application répétée des fonctions et nous allons calculer le taux d'erreur.
Voici le script pour le classifieur euclidien :

```
TauxError.ceuc <- function(X, z, N=20)
{
  error.app = vector("numeric", N)
  error.test = vector("numeric", N)

  for (i in 1:N)
  {
    donn.sep = separ1(X, z)
    Xapp = donn.sep$Xapp
    zapp = donn.sep$zapp
    Xtst = donn.sep$Xtst
    ztst = donn.sep$ztst

    mu = ceuc.app(Xapp, zapp)
    predit.app = ceuc.val(mu, Xapp)
    predit.test = ceuc.val(mu, Xtst)

    error.app[i] = mean(predit.app!=zapp) #Apprentissage
    error.test[i] = mean(predit.test!=ztst) #Test
  }

  error = NULL
  error$app = mean(error.app)
  error$test = mean(error.test)
  error$ICapp = IC(error$app,dim(Xapp)[1])
  error$ICtest = IC(error$test,dim(Xtst)[1])

  return (error)
}

#intervalle de confiance// binomial (moyenne):erreur à 0.95%
IC <- function(erreur,n) {
  tmp=1.96*sqrt(erreur*(1-erreur)/(n))
  IC1 = erreur-tmp;
  IC2= erreur+tmp;
  intervalle=c(IC1,IC2)
  return (intervalle)
}
```

Figure 6: La fonction **TauxErreur** et **Intervalle de confiance** avec euclidien

Pour calculer l'intervalle de confiance de l'erreur ϵ, on cherche à estimer la proportion π d'individus qui se situent dans la mauvaise classe. Nous supposons que cette dernière suit une loi de Bernoulli ϵ_i qui vaut 1 si le classifieur se trompe de classe et 0 sinon. On estime π la moyenne empirique des erreurs par $\overline{\epsilon} = 1/n \sum_{i=1}^{n} \epsilon_i$. Les v.a.r ϵ_i étant de Bernouilli on peut utiliser l'approximation donné par le TLC.
Partant de cette hypothèse, l'intervalle de confiance s'écrit sous la forme :

$$[\overline{\epsilon} - z_{1-\alpha/2}\sqrt{\frac{\overline{\epsilon}(1-\overline{\epsilon})}{n}}, \overline{\epsilon} + z_{1-\alpha/2}\sqrt{\frac{\overline{\epsilon}(1-\overline{\epsilon})}{n}}]$$

Pour $\alpha = 5\%$, on lit dans les tables $z_{1-\alpha/2} = z_{97,5\%} = 1.96$
Pour caculer l'intervalle de confiance, il nous suffit alors de remplacer dans les calculs la moyenne empirique aléatoire par son estimation ponctuelle.

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
Moyenne des erreurs sur l'apprentissage	0.23	0.09	0.14	0.13
Moyenne des erreurs sur le test	0.21	0.09	0.13	0.14
Intervalle de confiance d'apprentissage	[0.059;0.370]	[0.022;0.161]	[0.098;0.172]	[0.093;0.166]
Intervalle de confiance de test	[-0.010;0.434]	[-0.009;0.183]	[0.077;0.179]	[0.083,0.188]

On remarque que certaines bornes inférieurs sont négatives, ceci n'est pas une "erreur de calcul", mais les hypothèses de départ ne sont pas toutes à fait vrai. En effet, l'hypothèse énoncée précédemment est basée sur une aproximation qui est d'autant plus vrai que n est grand. De plus, l'indépendance des données n'est totalement pas vérifiée.

3. Nous effectuons à présent une séparation aléatoire des données avec la fonction separ1 en un ensemble d'apprentissage et un ensemble de test. Puis nous appliquons la fonction kppv.tune ayant le même ensemble d'apprentissage et de validation, le k optimal est k=1. Cela n'est guère étonnant car l'erreur dans ce cas présent est nulle; ce phénomène peut s'expliquer par le fait que le classifieur détermine la classe du point de test par rapport aux résultats d'apprentissage. Or, le point le plus proche du point de test est lui-même, donc le classifieur est sûr de ne jamais se tromper. C'est typiquement un cas de sur-apprentissage des données.

4.Voici le script de performance pour les K plus proches voisins :

```
performance.kppv = function(X, z, N)
{
  kopt = vector("numeric", N)
  error.app = vector("numeric", N)
  error.test = vector("numeric", N)

  for (i in 1:N)
  {
    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;

    k = kppv.tune(Xapp, zapp, Xval, zval, 2*(1:6)-1);
    predit.app = kppv.val(Xapp, zapp, k, Xapp);
    predit.test = kppv.val(Xapp, zapp, k, Xtst);
    kopt[i] = k;
    error.app[i] = mean(predit.app!=zapp);
    error.test[i] = mean(predit.test!=ztst);
  }

  res = NULL
  res$kopt = kopt
  res$error.app = mean(error.app)
  res$error.test = mean(error.test)
  res$error.ICapp = IC(res$error.app,dim(Xapp)[1])
  res$error.ICtest = IC(res$error.test,dim(Xtst)[1])
  return(res)
}
```

Figure 7: La fonction **TauxErreur** et **Intervalle de confiance** avec K-ppv

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
Moyenne des erreurs sur l'apprentissage	0.16	0.18	0.23	0.13
Moyenne des erreurs sur le test	0.34	0.43	0.49	0.15
Intervalle de confiance d'apprentissage	[0.003;0.332]	[0.071;0.284]	[0.188;0.295]	[0.099;0.158]
Intervalle de confiance de test	[0.046;0.644]	[0.229;0.625]	[0.406;0.582]	[0.106,0.196]

La constat que l'on tire des résultats est le fait que plus le classifieur possède des données pour apprendre, plus l'erreur d'apprentissage diminue (tend vers une valeur constante). De plus, l'erreur de test semble tendre vers l'erreur d'apprentissage (lorsqu'on réalise plusieurs fois le classifieur) preuve de la robustesse du classifieur pour étudier des données du même type. Si nous comparons les résultats avec le classifieur euclidien et le classifieur des KPPV, on peut vraisemblablement dire qu'ils sont de même performance pour ces jeux de données.

1.2.2 Jeu de données Synth2-1000

On considère maintenant le jeu de données **Synth2-1000**. En réalisant de manière analogue à la sous-section précédente, nous obtenons les estimations suivantes:

	Classe 1	Classe 2
Espérance μ_k	(3.02,-0.01)	(-2.14,-0.03)
Covariances Σ_k	$\begin{pmatrix} 0.99 & 0.11 \\ 0.11 & 1.09 \end{pmatrix}$	$\begin{pmatrix} 4.43 & -0.15 \\ -0.15 & 1.03 \end{pmatrix}$
Proportions π_k	0.523	0.477

Pour le calcul des intervalles de confiance, nous réalisons aussi de manière analogue.
Dans le cas du classifieur Euclidien, nous obtenons les valeurs suivantes:

	Données d'apprentissage	Données de tests
Moyenne des erreurs sur 20 échantillons à 10^{-3} de $\overline{\epsilon}$	0.064	0.059
Intervalle de confiance à 10^{-3}	[0.045;0.083]	[0.034;0.085]

On remarque directement que les estimations de ϵ sont presque identiques entre les données d'apprentissage et de tests, preuve de la robustesse du classifieur pour ce jeu de données.
Dans le cas du classifieur des K plus proches voisins, nous obtenons les valeurs suivantes:

	Données d'apprentissage	Données de tests
Moyenne des erreurs sur 20 échantillons à 10^{-3} de $\overline{\epsilon}$	0.048	0.058
Intervalle de confiance à 10^{-3}	[0.029;0.068]	[0.029;0.087]

On remarque comme précédemment une erreur d'apprentissage similaire à l'erreur de test. On peut donc conclure qu'au vue des résultats que ce classifieur est robuste et pertinent pour ce jeu de données.

1.2.3 Jeux de données réelles

Pour le calcul/estimation suivantes, nous procédons de la même façon que la sous-section précédente.

- Données **Pima**
  Pour la classifieur euclidien:

	Données d'apprentissage	Données de tests
Moyenne des erreurs sur 20 échantillons à 10^{-3} de $\overline{\epsilon}$	0.246	0.244
Intervalle de confiance à 10^{-3}	[0.201;0.292]	[0.180;0.307]

  Pour le classifeur des KPPV:

	Données d'apprentissage	Données de tests
Moyenne des erreurs sur 20 échantillons à 10^{-3} de $\overline{\epsilon}$	0.195	0.257
Intervalle de confiance à 10^{-3}	[0.147;0.243]	[0.182;0.331]

  Nous distinguons que les deux classifieurs ont des performances moindres par rapport aux jeux de données précédent. Notons tout de même que les 2 classes sont relativement bien représentées ($\frac{1}{3}$ pour la classe 2, et $\frac{2}{3}$ pour la classe 1). Il serait pertinent de choisir une autre règle de décision pour ce jeu de données ou de réaliser un "prétraitement" des données.

- Données **Breastcancer**
  Pour le classifieur euclidien:

	Données d'apprentissage	Données de tests
Moyenne des erreurs sur 20 échantillons à 10^{-3} de $\overline{\epsilon}$	0.038	0.041
Intervalle de confiance à 10^{-3}	[0.020;0.056]	[0.015;0.067]

  Pour le classifieur des KPPV:

	Données d'apprentissage	Données de tests
Moyenne des erreurs sur 20 échantillons à 10^{-3} de $\overline{\epsilon}$	0.021	0.040
Intervalle de confiance à 10^{-3}	[0.006;0.037]	[0.010;0.070]

Pour ce jeu de données on remarque une différence significative entre les deux classifieurs pour les données d'apprentissage. En revanche, pour les données de test le taux d'erreur est équivalent. On peut donc dire que ces deux classifieurs sont aussi performants par rapport au jeu de données Pima pour le jeu de données Breastcancer.

2 Règle de Bayes

N.B.: Pour tous les graphiques présents dans cette partie, nous utiliserons le code couleur suivant: ω_1: rouge, ω_2: vert.

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

L'énoncé nous donne pour les jeux de données **Synth1-40**, **Synth1-100**, **Synth1-500** et **Synth1-1000**: $\mu_1 = \begin{pmatrix}\mu_{11}\\ \mu_{12}\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix}$, $\mu_2 = \begin{pmatrix}\mu_{21}\\ \mu_{22}\end{pmatrix} = \begin{pmatrix}-2\\ 0\end{pmatrix}$,
$\sum_1 = \sum_2 = \begin{pmatrix}\sigma_{11}^2 & 0\\ 0 & \sigma_{22}^2\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$
Les distributions d'une loi continue s'expriment via leurs densités marginales:

$$f_1(x) = \frac{1}{(2\pi)^{p/2}(det\sum_k)^{1/2}}\exp(-\frac{1}{2}(x-\mu_k)^T\sum_k^{-1}(x-\mu_k) \quad avec \quad p=2 \quad et \quad k=1$$

soit

$$f_1(x) = \frac{1}{(2\pi)\sigma_{11}\sigma_{22}}\exp[-\frac{1}{2}((\frac{x_1-\mu_{11}}{\sigma_{11}})^2 + (\frac{x_2-\mu_{12}}{\sigma_{22}})^2)]$$

Cela nous donne numériquement:

$$f_1(x) = \frac{1}{2\pi}\exp[-\frac{1}{2}((x_1)^2 + (x_2-1)^2)] \quad \text{et} \quad f_2(x) = \frac{1}{2\pi}\exp[-\frac{1}{2}((x_1+2)^2 + (x_2)^2)]$$

Comme $f_1(x)$ et $f_2(x)$ sont des vecteurs gaussiens et suivent une loi normale, nous pouvons donc écrire d'après les fonctions de densité précédentes: $X^1_{\omega_1} \sim \mathcal{N}(0,1); X^1_{\omega_2} \sim \mathcal{N}(-2,1)$;
et $X^2_{\omega_1} \sim \mathcal{N}(1,1); X^2_{\omega_2} \sim \mathcal{N}(0,1)$

On applique le même raisonnement au jeu de données **Synth2-1000** et nous obtenons:

$$f_1(x) = \frac{1}{2\pi}\exp[-\frac{1}{2}((x_1-3)^2 + (x_2)^2)] \quad \text{et} \quad f_2(x) = \frac{1}{2\pi\sqrt{5}}\exp[-\frac{1}{2}(\frac{(x_1+2)^2}{5} + (x_2)^2)]$$

et on a: $X^1_{\omega_1} \sim \mathcal{N}(3,1); X^1_{\omega_2} \sim \mathcal{N}(-2,5)$;
et $X^2_{\omega_1} \sim \mathcal{N}(0,1); X^2_{\omega_2} \sim \mathcal{N}(0,1)$

2.2 Détermination des courbes d'iso-densité dans chacune des classes

Les courbes d'iso-densité sont telles que $f_1(x) = C_1$ et $f_2(x) = C_2$ où C_1 et C_2 sont des constantes. Nous obtenons ainsi l'équation suivante pour les jeux de données **Synth1-40**, **Synth1-100**, **Synth1-500** et **Synth1-1000** :

$$f_1(x) = \frac{1}{(2\pi)\sigma_{11}\sigma_{22}}\exp[-\frac{1}{2}((\frac{x_1-\mu_{11}}{\sigma_{11}})^2 + (\frac{x_2-\mu_{12}}{\sigma_{22}})^2)] = C_1 \Leftrightarrow$$

$$-\frac{1}{2}((\frac{x_1-\mu_{11}}{\sigma_{11}})^2 + (\frac{x_2-\mu_{12}}{\sigma_{22}})^2) = \ln(2\pi\sigma_{11}\sigma_{22}C_1) \Leftrightarrow$$

$$(\frac{x_1-\mu_{11}}{\sigma_{11}})^2+(\frac{x_2-\mu_{12}}{\sigma_{22}})^2=-2\ln(2\pi\sigma_{11}\sigma_{22}C_1)$$

soit numériquement: $(x_1)^2+(x_2-1)^2=-2\ln(2\pi C_1)$
On remarque que cette équation est l'équation caractéristique d'un cercle de centre (0,1) et de rayon $r=\sqrt{-2\ln(2\pi C_1)}$ avec $0<C_1\leq\frac{1}{2\pi}$

On applique le même raisonnement à $f_2(x)$ et on obtient:

$$(\frac{x_1-\mu_{21}}{\sigma_{11}})^2+(\frac{x_2-\mu_{22}}{\sigma_{22}})^2=-2\ln(2\pi\sigma_{11}\sigma_{22}C_2)$$

soit numériquement: $(x_1+2)^2+(x_2)^2=-2\ln(2\pi C_2)$
On remarque que cette équation est l'équation caractéristique d'un cercle de centre (-2,0) et de rayon $r=\sqrt{-2\ln(2\pi C_2)}$ avec $0<C_2\leq\frac{1}{2\pi}$

Pour le jeu de données **Synth2-1000**:
On a pour $f_1(x)$:

$$(\frac{x_1-\mu_{11}}{\sigma_{11}})^2+(\frac{x_2-\mu_{12}}{\sigma_{22}})^2=-2\ln(2\pi\sigma_{11}\sigma_{22}C_3)$$

soit numériquement: $(x_1-3)^2+(x_2)^2=-2\ln(2\pi C_3)$
C'est un cercle de centre (3,0) et de rayon $r=\sqrt{-2\ln(2\pi C_3)}$ avec $0<C_3\leq\frac{1}{2\pi}$

On a pour $f_2(x)$:

$$(\frac{x_1-\mu_{21}}{\sigma_{11}})^2+(\frac{x_2-\mu_{22}}{\sigma_{22}})^2=-2\ln(2\pi\sigma_{11}\sigma_{22}C_4)$$

soit numériquement:

$$\frac{(x_1+2)^2}{5}+(x_2)^2=-2\ln(2\pi\sqrt{5}C_4)$$

ou

$$\frac{(x_1+2)^2}{-10\ln(2\pi\sqrt{5}C_4)}+\frac{(x_2)^2}{-2\ln(2\pi\sqrt{5}C_4)}=1$$

Cette équation représente une ellipse de centre (-2,0) et de rayon $\sqrt{-2\ln(2\pi\sqrt{5}C_4)}$ par rapport à y et $\sqrt{-10\ln(2\pi\sqrt{5}C_4)}$ par rapport à x avec $0<C_4\leq\frac{1}{2\pi\sqrt{5}}$.

2.3 Détermination de la règle de Bayes

Pour le problème de discrimination ω_1 et ω_2, nous déterminons la règle de Bayes:

$$\delta^*(x)=\begin{cases} a_1 & \text{si} \quad P(\omega_2|x)<P(\omega_1|x) \\ a_2 & \text{sinon.}\end{cases}$$

que l'on peut exprimer avec le rapport de vraisemblance:

$$\delta^*(x)=\begin{cases} a_1 & \text{si} \quad \frac{f_1(x)}{f_2(x)}>\frac{\pi_2}{\pi_1} \\ a_2 & \text{sinon.}\end{cases}$$

où π_2 et π_1 sont les probabilités à priori de la classe 2 et de la classe 1, respectivement.

Pour les quatre premiers jeux de données, les deux classes suivent des distributions de même Σ. Nous avons donc d'après ce qui précède:

$$\frac{f_1(x)}{f_2(x)}>\frac{\pi_2}{\pi_1}\Leftrightarrow\frac{\frac{1}{(2\pi)}\exp[-\frac{1}{2}((x_1)^2+(x_2-1)^2)]}{\frac{1}{(2\pi)}\exp[-\frac{1}{2}((x_1+2)^2+(x_2)^2)]}>1\Leftrightarrow$$

$$\exp[-\frac{1}{2}((x_1)^2+(x_2-1)^2)+\frac{1}{2}((x_1+2)^2+(x_2)^2)]]>1 \Leftrightarrow$$

$$-\frac{1}{2}((x_1)^2+(x_2-1)^2)+\frac{1}{2}((x_1+2)^2+(x_2)^2)]>\ln(1)=0 \Leftrightarrow$$

$$2x_1+\frac{x_2}{2}>-\frac{3}{2} \quad \text{ou} \quad x_2>-4x_1-3$$

Nous sommes donc dans le cas de l'analyse discriminante linéaire où $k=-3$; ce qui est concordant avec le résultat de linéarité du cours.

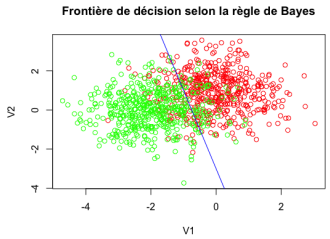

Figure 8: Frontière de décision correspondant à la règle de Bayes avec le jeu de donnée *Synth1-1000.csv*

2.4 Calcul l'expression de la règle de Bayes pour le jeu de données Synth2-1000

Pour le jeu de données *Synth2-1000*, on a l'inégalité suivante:

$$\frac{f_1(x)}{f_2(x)}>\frac{\pi_2}{\pi_1} \Leftrightarrow \frac{\frac{1}{2\pi}\exp[-\frac{1}{2}((x_1-3)^2+(x_2)^2)]}{\frac{1}{2\pi\sqrt{5}}\exp[-\frac{1}{2}(\frac{(x_1+2)^2}{5}+(x_2)^2)]}>1 \Leftrightarrow$$

$$\sqrt{5}\exp[-\frac{1}{2}((x_1-3)^2+(x_2)^2)+\frac{1}{2}(\frac{(x_1+2)^2}{5}+(x_2)^2)]>1 \Leftrightarrow$$

$$-\frac{1}{2}((x_1-3)^2+(x_2)^2)+\frac{1}{2}(\frac{(x_1+2)^2}{5}+(x_2)^2)>\ln(\frac{1}{\sqrt{5}}) \Leftrightarrow$$

$$-4x_1^2+40x_1>10\ln(\frac{1}{\sqrt{5}})+41 \quad \text{ou} \quad x_1^2<10x_1-4[10\ln(\frac{1}{\sqrt{5}})+41]$$

On a alors $k_1=10$ et $k_2=-4[10\ln(\frac{1}{\sqrt{5}})+41]$
La représentation de cette courbe, nous donne la figure suivante:

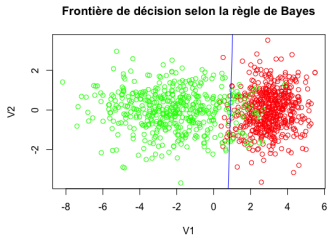

Figure 9: Frontière de décision correspondant à la règle de Bayes avec le jeu de donnée *Synth2-1000.csv*

2.5 Calcul l'expression formelle de l'erreur de Bayes en fonction de k, ou k1 et k2

Pour les données **Synth1-n**, nous sommes en présence de deux classes avec une même matrice de covariance Σ, et $\pi_1 = \pi_2$ (probabilité à priori égal). La probabilité d'erreur de Bayes s'écrit donc sous la forme:

$$\epsilon^* = \Phi(-\frac{\Delta}{2}) \quad \text{avec} \quad \Delta^2 = (\mu_2 - \mu_1)^T \Sigma^{-1} (\mu_2 - \mu_1)$$

Soit numériquement :

$$\Delta = \sqrt{5} \quad \text{et} \quad \epsilon^* = \Phi(-1.12) = 1 - \Phi(1.12) = 1 - 0.8686 = 0.1314$$

Le taux d'erreur de Bayes est donc de 0.13 pour les quatre jeux de données Synth1-n. On retrouve le même résultat avec les 2 classifieurs dans le cas de Synth1-1000, preuve que les deux classifieurs précédents sont "adaptés" pour ce jeu de données.

Pour les données **Synth2-1000**, nous sommes en présence de deux classes avec une matrice de covariance Σ différente, et $\pi_1 = \pi_2$ (probabilité à priori égal). La probabilité d'erreur de Bayes s'écrit donc sous la forme:

$$\epsilon^* \leq \sqrt{\pi_1 \pi_2} e^{-\Delta_B^2} \quad \text{où} \quad \Delta_B^2 \quad \text{est la distance de Bhattacharyya}$$

Numériquement, nous obtenons :

$$\Delta_B^2 = 1.54 \quad \text{et} \quad \epsilon^* \leq 0.11$$

Le taux d'erreur de Bayes donc inférieurs à 0.11 (à 10^{-2} près). Par ailleurs, les résultats que l'on trouve pour les 2 classifeurs sont inférieurs à cette borne.

3 Conclusion du TP

Ce TP nous a permis d'appliquer une partie des méthodes supervisées via l'outil statistique R. Dans un premier temps, nous avons étudié le classifieur euclidien et le classifieur des K plus proches voisins (KKPV) tant en évaluant leur fiabilité qu'en calculant les probabilités d'erreur et intervalle de confiance. Ensuite, nous avons appliqué la règle de Bayes en calculant les frontières de décisions sur les jeux de données précédents, nous permettant de comparer nos résultats avec la probabilité d'erreur de Bayes.