SY09 - Analyse des données et Data-Mining

Printemps 2017 (P17)

TP3

Discrimination, théorie bayésienne de la décision

Maxime Churin GI04 - Louis Denoyelle GI04

Introduction

Après les TP sur les méthodes non-supervisées, nous poursuivons ici avec une introduction aux méthodes supervisées. Ce TP aborde les concepts de discrimination de classes ainsi que la théorie bayésienne de la décision. La première partie nous fait découvrir une utilisation plus poussée du logiciel `R` avec l'utilisation de fonctions dans le but de simuler des critères de décision puis de les analyser sur des jeux de données abstraits et réels. La deuxième partie est une application de la règle de Bayes sur un cas linéaire et un cas quadratique.

1 Classifieur euclidien, *K* plus proches voisins

1.1 Programmation

Dans cette première partie, nous avons programmé les fonctions permettant l'étude du classifieur euclidien et des K plus proches voisins. Nous avons décidé de ne pas limiter l'utilisation de nos fonctions aux jeux de données binaires en les rendant valides pour un nombre g de classes. Nous détaillerons, ci-dessous, leur fonctionnement général et le code, plus longuement annoté, est donné dans le fichier *functions-1-1.R* joint au rapport. Pour ces deux classifieurs, on émet l'hypothèse que le vecteur d'étiquettes associé à chaque jeu de données contient des valeurs entières successives de 1 à g. Le maximum d'un tel vecteur correspond donc au nombre de classes du jeu de données.

1.1.1 Classifieur euclidien

La fonction *ceuc.app* prend en paramètre les données d'apprentissage sous la forme d'un tableau individus-variables et d'un vecteur associant à chaque individu sa classe. Pour retourner la matrice de dimension $g \cdot p$ contenant les paramètres estimés du classifieur euclidien, on calcule la moyenne de chaque variable pour chaque classe du jeu de données.

La seconde, *ceuc.val*, prend en paramètre la matrice des paramètres estimés, qui a pu être obtenue via la fonction précédente, ainsi qu'un ensemble de test sous la forme d'un tableau individus-variables. Son rôle va être d'associer à chaque individu de l'ensemble de test une classe déterminée par le critère de la distance euclidienne minimale au vu des paramètres estimés. Elle retourne donc un vecteur associant à chaque individu de l'ensemble de test une classe prédite par ce critère.

1.1.2 *K* plus proches voisins

La fonction *kppv.val* prend en paramètre les données d'apprentissage sous la forme d'un tableau individus-variables et d'un vecteur associant à chaque individu sa classe, le nombre de voisins K à utiliser pour le critère de décision et, enfin, l'ensemble de test sous la forme d'un tableau individus-variables. Elle va être en charge d'associer à chaque individu i de l'ensemble de test la classe la plus représentée parmi ses K plus proches voisins : K individus de l'ensemble d'apprentissage avec la distance euclidienne la plus faible par rapport à i. Elle retourne donc, un vecteur associant à chaque individu de l'ensemble de test une classe prédite par ce critère.

La seconde, *kppv.tune*, prend en paramètre les données d'apprentissage et de validation sous la forme d'un tableau individus-variables et d'un vecteur associant à chaque individu sa classe et, enfin, un ensemble de nombre de voisins à tester. En utilisant la fonction précédente sur l'ensemble de validation pour chaque valeur de K à tester, elle stocke la similarité entre le vecteur de classes prédites et celui de validation. Elle est alors capable de retourner le nombre de voisins optimal à utiliser pour réaliser le moins d'erreur de prédiction de classe sur l'ensemble de validation.

1.1.3 Test des fonctions

On utilise alors le script *script-1-1.R* joint au rapport pour vérifier nos fonctions. On observe alors qu'elles nous donnent des frontières de décision identiques à celles données dans le sujet du TP sur le jeu de données *Synth1-40*.

1.2 Évaluation des performances

Pour chaque jeu de données étudié, le taux d'erreur ε pour un classifieur euclidien ou selon la méthode des K plus proches voisins sera analysé. Pour estimer ε, 20 répétitions seront effectuées où le jeu de données sera séparé en un ensemble d'apprentissage, un ensemble de test et également un ensemble de validation pour la méthode des K plus proches voisins. À la fin de ces répétitions, une estimation ponctuelle $\widehat{\varepsilon}$ de ce taux d'erreur sera déterminée ainsi qu'un intervalle de confiance.

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

Les premiers jeux de données étudiés sont issus de `Synth-1` et suivent une loi normale bivariée qui ne diffèrent que par l'effectif. On estime pour chaque jeu de données les paramètres π_k, μ_k et $\sum_k$ à l'aide d'une fonction nommée `estimation`

Paramètre	Classe 1	Classe 2
π_k	0.45	0.55
μ_k	$\begin{pmatrix} -0.31648 & 1.09195 \end{pmatrix}$	$\begin{pmatrix} -1.88338 & 0.10512 \end{pmatrix}$
$\sum_k$	$\begin{pmatrix} 0.68163 & 0.11936 \\ 0.11936 & 1.01290 \end{pmatrix}$	$\begin{pmatrix} 1.37521 & 0.32273 \\ 0.32273 & 1.43933 \end{pmatrix}$

Table 1.1 – Estimation des paramètres pour le jeu de données Synth1-40.

Paramètre	Classe 1	Classe 2
π_k	0.54	0.46
μ_k	$\begin{pmatrix} 0.02572 & 0.81533 \end{pmatrix}$	$\begin{pmatrix} -1.96542 & -0.12650 \end{pmatrix}$
$\sum_k$	$\begin{pmatrix} 0.88164 & -0.13126 \\ -0.13126 & 1.10884 \end{pmatrix}$	$\begin{pmatrix} 0.75698 & -0.03763 \\ -0.03763 & 0.76107 \end{pmatrix}$

Table 1.2 – Estimation des paramètres pour le jeu de données Synth1-100.

Paramètre	Classe 1	Classe 2
π_k	0.528	0.472
μ_k	$\begin{pmatrix} 0.13164 & 0.87971 \end{pmatrix}$	$\begin{pmatrix} -1.88345 & -0.08475 \end{pmatrix}$
$\sum_k$	$\begin{pmatrix} 1.05015 & 0.05222 \\ 0.05222 & 0.98388 \end{pmatrix}$	$\begin{pmatrix} 0.96687 & -0.11089 \\ -0.11089 & 0.97940 \end{pmatrix}$

Table 1.3 – Estimation des paramètres pour le jeu de données Synth1-500.

Paramètre	Classe 1	Classe 2
π_k	0.496	0.504
μ_k	$\begin{pmatrix} -0.01279 & 0.91568 \end{pmatrix}$	$\begin{pmatrix} -1.96120 & 0.01834 \end{pmatrix}$
$\sum_k$	$\begin{pmatrix} 0.96857 & -0.06521 \\ -0.06521 & 1.07943 \end{pmatrix}$	$\begin{pmatrix} 0.99042 & 0.02095 \\ 0.02095 & 0.94129 \end{pmatrix}$

Table 1.4 – Estimation des paramètres pour le jeu de données Synth1-1000.

L'estimation de ces paramètres nous permet d'observer que les effectifs sont presque à chaque fois équitablement répartis et les matrices $\sum_k$ très proches de la matrice unité. Exception faite pour `Synth1-40` où l'effectif plus faible est moins représentatif. En effet, les valeurs extrêmes ont plus de poids sur la moyenne et sur la variance. Cela indique que les nuages de points doivent ressembler à des sphères distinctes avec les mêmes effectifs.

On estime à présent les taux d'erreurs pour le classifieur euclidien et la méthode des K plus proches voisins. L'estimation du taux d'erreur s'effectue en comptant les différences entre la classe prédite et la classe réelle. Ce taux est calculé sur 20 répétitions et son estimation $\widehat{\varepsilon}$ correspond à la moyenne sur ces 20 répétitions. On décide de calculer un intervalle de confiance fiable à 95%, car il s'agit d'une valeur de référence qui laisse une faible marge d'erreur tout en restant suffisamment restreint. L'intervalle de confiance à 95% est déterminé dans le cas gaussien à l'aide de la formule suivante : $[\bar{X_n} - 1.96\frac{\sigma}{\sqrt{n}}, \bar{X_n} + 1.96\frac{\sigma}{\sqrt{n}}]$

Les fonctions `R` permettant de calculer les taux d'erreurs sont les fonctions `evalutationCeuc` pour le classifieur euclidien et `evaluationKppv` pour la méthode des K plus proches voisins. Ces fonctions renvoient un tableau avec le taux d'erreur pour chaque répétition sur les données d'apprentissage et de test. Le taux d'erreur ainsi que l'intervalle de confiance sont ensuite déterminés à l'aide de la fonction `estimationTauxErreur` qui prend en entrée le tableau des erreurs sur les données d'apprentissage et de test récupéré précédemment.

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
$\widehat{\varepsilon}$ apprentissage	0.20740	0.08955	0.12882	0.14085
IC apprentissage	[0.18674 ;0.22807]	[0.07873 ;0.10037]	[0.12429 ;0.13336]	[0.13596 ;0.14574]
$\widehat{\varepsilon}$ test	0.23461	0.09242	0.13892	0.14639
IC test	[0.19021 ;0.27901]	[0.07245 ;0.1123]	[0.13033 ;0.14750]	[0.13666 ;0.15613]

Table 1.5 – Estimation ponctuelle du taux d'erreur $\widehat{\varepsilon}$ avec un intervalle de confiance à 95% pour le classifieur euclidien.

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
$\widehat{\varepsilon}$ apprentissage	0.115	0.066	0.1312	0.1349
IC apprentissage	[0.05942 ;0.17057]	[0.04648 ;0.08551]	[0.12538 ;0.13701]	[0.12885 ;0.14094]
$\widehat{\varepsilon}$ test	0.34	0.10625	0.1356	0.1516
IC test	[0.27900 ;0.40099]	[0.08618 ;0.12631]	[0.12375 ;0.14744]	[0.14117 ;0.16202]

Table 1.6 – Estimation ponctuelle du taux d'erreur $\widehat{\varepsilon}$ avec un intervalle de confiance à 95 % pour la méthode des K plus proches voisins.

On remarque que le taux d'erreur étant relativement bas le classifieur euclidien et les K plus proches voisins sont d'assez bons prédicateurs de classe. La différence la plus flagrante entre les deux méthodes se retrouve dans le jeu de données `Synth1-40`, ce qui est normal car il s'agit

de l'effectif le plus bas donc le plus sujet à des variations importantes. On remarque que le calcul du taux d'erreur donne de meilleurs résultats pour les données d'apprentissage que pour les données de test mais que les deux taux d'erreur restent tout de même relativement proches, surtout pour les jeux de données avec le plus grand effectif. Les intervalles de confiance étant assez resserrés, on constate une constance dans le calcul de $\widehat{\varepsilon}$. Les intervalles de confiance sont plus resserrés dans le cas des données d'apprentissage car le calcul se fait sur un plus grand nombre de données. Cela majoritairement dû à la méthode de séparation qui attribut un plus grand nombre de données aux données d'apprentissage. On remarque également que le taux d'erreur pour la méthode des K plus proches voisins est souvent plus faible pour les données d'apprentissage alors que celui du classifieur euclidien est plus faible pour les données de test. Cependant, cette différence n'est ni systématique ni significative pour en tirer des conclusions.

Enfin, en utilisant la fonction `kppv.tune` avec l'ensemble d'apprentissage comme ensemble de test, la fonction renvoie un nombre de voisins optimal égal à 1. C'est tout à fait logique car il trouvera, comme voisin le plus proche de ce point, le point lui-même. La fiabilité de l'algorithme serait de 100%.

1.2.2 Jeu de données Synth2-1000

Paramètre	Classe 1	Classe 2
π_k	0.523	0.477
μ_k	$\begin{pmatrix} 3.01838 & -0.00638 \end{pmatrix}$	$\begin{pmatrix} -2.14228 & -0.02652 \end{pmatrix}$
$\sum_k$	$\begin{pmatrix} 0.99040 & 0.11314 \\ 0.11314 & 1.09287 \end{pmatrix}$	$\begin{pmatrix} 4.43478 & -0.15440 \\ -0.15440 & 1.03086 \end{pmatrix}$

Table 1.7 – Estimation des paramètres pour le jeu de données Synth2-1000.

Nous remarquons, avec les valeurs de π_k, que ce jeu de données est, tout comme `Synth1-N`, assez équitablement réparti. Les valeurs de μ_k et de $\sum_k$ nous montrent qu'il s'agit également de nuages de points de forme sphérique et plutôt distincts.

	Synth2-1000
$\widehat{\varepsilon}$ apprentissage	0.06169
IC apprentissage	[0.05927 ;0.06411]
$\widehat{\varepsilon}$ test	0.06591
IC test	[0.06103 ;0.07079]

Table 1.8 – Estimation ponctuelle du taux d'erreur $\widehat{\varepsilon}$ avec un intervalle de confiance à 95% pour le classifieur euclidien.

	Synth2-1000
$\widehat{\varepsilon}$ apprentissage	0.0524
IC apprentissage	[0.04615 ;0.05864]
$\widehat{\varepsilon}$ test	0.0586
IC test	[0.05151 ;0.06568]

Table 1.9 – Estimation ponctuelle du taux d'erreur $\widehat{\varepsilon}$ avec un intervalle de confiance pour la méthode des K plus proches voisins.

Les résultats du calcul du taux d'erreur nous rappellent celui du jeu de données `Synth1-N`, avec un taux d'erreur assez proche pour les deux méthodes (autour de 0.06) et un intervalle assez resserré. On remarque que ce taux d'erreur est assez bas, ce qui montre que les deux méthodes appliquées ici sont plutôt efficaces et prouve que la répartition des classes est marquée.

1.2.3 Jeux de données réelles

Les jeux de données analysés ici sont ceux de femmes atteintes de diabète (`Pima`) et de femmes atteintes de cancer du sein (`Breastcancer`). On évalue encore une fois le taux d'erreur $\widehat{\varepsilon}$ pour le classifieur euclidien et les K plus proches voisins :

	Pima	Breastcancer
$\widehat{\varepsilon}$ apprentissage	0.24577	0.03714
IC apprentissage	[0.24014 ;0.25141]	[0.03483 ;0.03946]
$\widehat{\varepsilon}$ test	0.24802	0.04144
IC test	[0.23829 ;0.25776]	[0.03617 ;0.04672]

Table 1.10 – Estimation ponctuelle du taux d'erreur $\widehat{\varepsilon}$ avec un intervalle de confiance à 95 % pour le classifieur euclidien.

	Pima	Breastcancer
$\widehat{\varepsilon}$ apprentissage	0.21184	0.02587
IC apprentissage	[0.20256 ;0.22112]	[0.01969 ;0.03206]
$\widehat{\varepsilon}$ test	0.23496	0.03618
IC test	[0.21871 ;0.25120]	[0.02902 ;0.04334]

Table 1.11 – Estimation ponctuelle du taux d'erreur $\widehat{\varepsilon}$ avec un intervalle de confiance à 95% pour la méthode des K plus proches voisins.

On remarque que le taux d'erreur $\widehat{\varepsilon}$ est beaucoup plus important dans les données `Pima` que dans les données `Breastcancer` où le taux d'erreur est très faible. Ces données réelles nous montrent qu'il est rare de se tromper dans le diagnostic d'un cancer du sein et beaucoup moins dans le cas du diabète. Cependant, ce taux d'erreur n'est tout de même pas nul dans le cas du cancer du sein, et peut être extrêmement dommageable. C'est le cas d'un diagnostic d'un « *faux négatif* », cas où un diagnostic se révèle négatif alors que la personne est quand même atteinte d'un cancer. Le cas d'un « *faux positif* » l'est beaucoup moins car des analyses plus poussées pourraient permettre de lever le doute sur ce diagnostic.

2 Règle de Bayes

Après avoir utilisé des critères basés sur des distances, nous allons maintenant analyser une règle de décision minimisant la probabilité d'erreur d'affectation d'un individus à une classe : la règle de Bayes dans le cas de deux classes.

Les jeux de données *Synth1-N* et *Synth2-1000* possède un vecteur de caractéristiques $x = (x_1, x_2)^T$ et chaque classe k suit une loi normale multidimensionnelle d'espérance μ_k et de variance Σ_k.

2.1 Distributions marginales

Les jeux de données *Synth1-N* validant l'hypothèse d'homoscédasticité : $\Sigma_k = \Sigma$, on se place dans le modèle d'une analyse discriminante linéaire (ADL). On écrit donc la loi jointe :

$$f_k(x) = \frac{1}{(2\pi)^{p/2}(\det \Sigma)^{1/2}} \exp(-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k))$$

En injectant les paramètres, on obtient $f_1(x)$ pour la classe 1 et $f_2(x)$ pour la classe 2 :

$$f_1(x) = \frac{1}{2\pi} \exp(-\frac{1}{2}(x_1^2 + (x_2-1)^2) \tag{2.1}$$

$$f_2(x) = \frac{1}{2\pi} \exp(-\frac{1}{2}((x_1+2)^2 + x_2^2) \tag{2.2}$$

Les variables étant indépendantes, on peut écrire la loi jointe comme le produit des lois marginales :

$$f_1(x_1) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x_1^2) \qquad f_1(x_2) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x_2-1)^2)$$

$$f_2(x_1) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x_1+2)^2) \qquad f_2(x_2) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x_2^2)$$

Sachant que la densité de probabilité de la loi normale est donnée par :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2)$$

On remarque ainsi que pour les données *Synth1-N* les variables de chaque classe suivent une loi normale $N(\mu, \sigma^2)$ telle que pour la classe 1 : $X^1 \sim N(0,1)$ et $X^2 \sim N(1,1)$ et pour la classe 2 : $X^1 \sim N(-2,1)$ et $X^2 \sim N(0,1)$.

Le jeu de données *Synth2-1000* caractérisé par des paramètres μ_k et Σ_k différents, on se place dans le modèle d'une analyse discriminante quadratique (ADQ). On écrit donc la loi jointe :

$$f_k(x) = \frac{1}{(2\pi)^{p/2}(\det \Sigma_k)^{1/2}} \exp(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k))$$

$$f_1(x) = \frac{1}{2\pi} \exp(-\frac{1}{2}((x_1-3)^2 + x_2^2) \tag{2.3}$$

$$f_2(x) = \frac{1}{\sqrt{5} \cdot 2\pi} \exp(-\frac{1}{2}((\frac{x_1 + 2}{\sqrt{5}})^2 + x_2^2) \tag{2.4}$$

En appliquant le même raisonnement que sur le précédent jeu de données, on obtient pour la classe 1 $X^1 \sim N(3, 1)$ et $X^2 \sim N(0, 1)$ et pour la classe 2 $X^1 \sim N(-2, 5)$ et $X^2 \sim N(0, 1)$.

2.2 Coursbes d'iso-densités

Les courbes d'iso-densités sont telles que $f_1(x) = K_1$ et $f_2(x) = K_2$ pour chaque jeu de données.

Pour *Synth1-N*, on obtient d'après les équations (2.1) et (2.2) les courbes suivantes :

$$(2.1) = K_1 \qquad\qquad (2.2) = K_2$$
$$x_1^2 + (x_2 - \mu_{12})^2 = -2\ln(2\pi K_1) \qquad\qquad (x_1 - \mu_{21})^2 + x_2^2 = -2\ln(2\pi K_2)$$

Ces deux courbes correspondent à des équations de cercle telles que la courbe issue de $f_1(x)$ a pour rayon $\sqrt{-2 \cdot \ln(2\pi K_1)}$ et pour centre $(0, \mu_{12} = 1)$ et la courbe issue de $f_2(x)$ a pour rayon $\sqrt{-2 \cdot \ln(2\pi K_2)}$ et pour centre $(\mu_{21} = -2, 0)$.

Pour *Synth2-1000*, on obtient d'après les équations (2.3) et (2.4) les courbes suivantes :

$$(2.3) = K_1 \qquad\qquad (2.4) = K_2$$
$$(x_1 - \mu_{11})^2 + x_2^2 = -2\ln(2\pi K_1) \qquad\qquad (\frac{x_1 - \mu_{21}}{\sigma_{21}})^2 + x_2^2 = -2\ln(\sigma_{21} 2\pi K_2)$$
$$(\frac{x_1 - \mu_{21}}{\sigma_{21} \cdot \sqrt{-2\ln(\sigma_{21} 2\pi K_2)}})^2 + (\frac{x_2}{\sqrt{-2\ln(\sigma_{21} 2\pi K_2)}})^2 = 1$$

Ces deux courbes correspondent respectivement à une équation de cercle et une équation d'ellipse tels que la courbe issue de $f_1(x)$ a pour rayon $\sqrt{-2 \cdot \ln(2\pi K_1)}$ et pour centre $(\mu_{11} = 3, 0)$ et la courbe issue de $f_2(x)$ a pour rayon $r_{x_1} = \sigma_{21}\sqrt{-2\ln(\sigma_{21} 2\pi K_2)}$, $r_{x_2} = \sqrt{-2\ln(\sigma_{21} 2\pi K_2)}$ et pour centre $(\mu_{21} = -2, 0)$.

2.3 Règle de Bayes et frontière de décision

Dans le cas de jeux de données à deux classes la règle de Bayes s'exprime en fonction du rapport de vraisemblance $f_1(x)/f_2(x)$ avec a_k représentant l'affectation de x à la classe ω_k :

$$\delta^*(x) = a_1 \Leftrightarrow \mathbb{P}(\omega_1|x) > \mathbb{P}(\omega_2|x) \Leftrightarrow \frac{f_1(x)\pi_1}{f(x)} > \frac{f_2(x)\pi_2}{f(x)} \Leftrightarrow \frac{f_1(x)}{f_2(x)} > \frac{\pi_2}{\pi_1}$$

Or nos jeux de données étant obtenus avec $\pi_1 = \pi_2$, on a :

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

Pour chaque jeu de données issu de *Synth1*, il est évident que la règle de Bayes est une fonction linéaire : l'hypothèse d'homoscédasticité permettant d'annuler les coefficients de degré 2. On peut donc réécrire la règle de Bayes grâce aux équations (2.1) et (2.2) :

$$\delta^*(x) = \begin{cases} a_1 & si \quad x_2 < -2x_1 - k \quad avec \quad k = -1.5 \\ a_2 & sinon \end{cases}$$

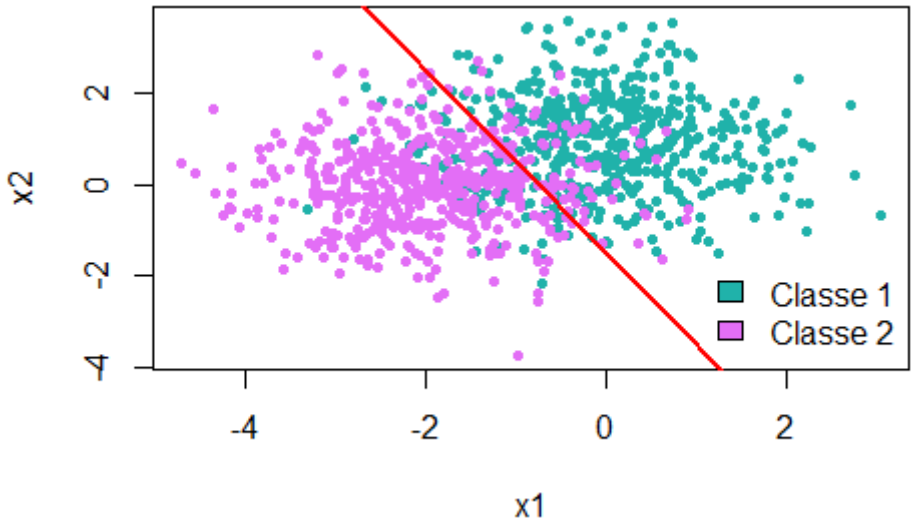

Figure 2.1 – Frontière de décision $x2 = -2x1 - 1.5$ dans le plan formé par les variables $x1$ et $x2$ avec les données de Synth1-1000.

On constate que la zone au dessus de cette droite détermine la région de décision pour la classe 1 et que la zone en dessous de cette droite détermine la région de décision pour la classe 2. Graphiquement, on observe que de nombreux points autour de cette droite n'ont pas forcément la classe à laquelle on pourrait s'attendre, ainsi, un nouvel examen de ces point par cette règle de décision impliquerait une erreur d'affectation de classe. Cette frontière est valable pour les différents jeux de données *Synth1* car ils suivent la même distribution. Une autre méthode pour déterminer cette frontière aurait été d'utiliser les fonctions discriminantes linéaires telles que $h_1(x) = h_2(x)$.

Pour le jeu de données *Synth2-1000*, on peut réécrire la règle de Bayes grâce aux équations (2.3) et (2.4) :

$$\begin{aligned} & (2.3) > (2.4) \\ (x_1 - 3)^2 - 2\ln(\sqrt{5}) &< (\frac{x_1 + 2}{\sqrt{5}})^2 \\ 0.8x_1^2 - 6.8x_1 + 6.59 &< 0 \end{aligned}$$

$$\delta^*(x) = \begin{cases} a_1 & si \quad x_1 > k_1 = 1.1155 \quad et \quad x_1 < k_2 = 7.38 \\ a_2 & sinon \end{cases}$$

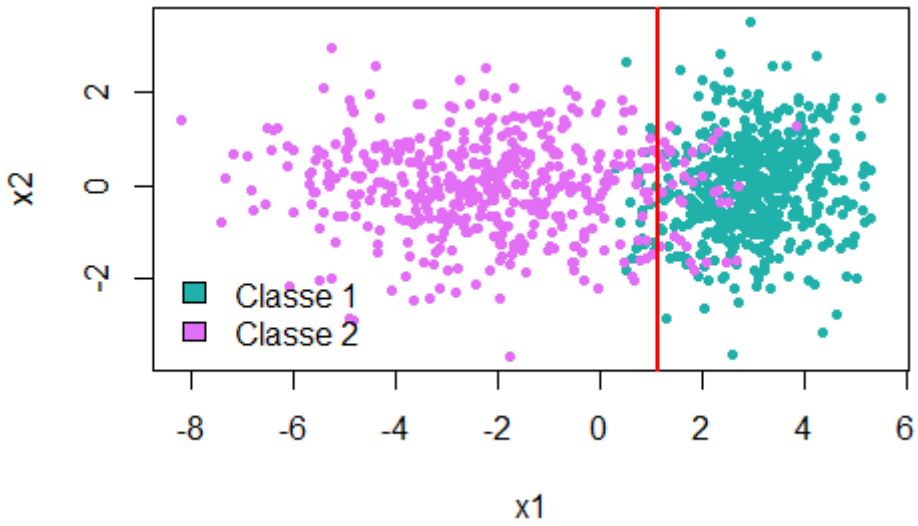

Figure 2.2 – Frontière de décision pour les classes 1 et 2 dans le plan formé par les variables x_1 et x_2 avec les données de Synth2-1000.

On constate que la zone à gauche de la droite détermine la région de décision pour la classe 2 et que la zone à droite de cette droite détermine la région de décision pour la classe 1. On peut réaliser la même observation graphique que sur la Figure 2.1. Une autre méthode pour déterminer cette frontière aurait été d'utiliser les fonctions discriminantes quadratiques telles que $g_1(x) = g_2(x)$.

2.4 Erreur de Bayes

Les jeux de de données *Synth1-N* validant l'hypothèse d'homoscédasticité et possédant seulement 2 classes avec $\pi_1 = \pi_2$, on écrit la probabilité d'erreur de Bayes suivante :

$$\begin{aligned} \varepsilon^* &= \phi(-\frac{\Delta}{2}) \quad \text{avec} \quad \Delta = \sqrt{(\mu_2 - \mu_1)^T \Sigma^{-1} (\mu_2 - \mu_1)} \\ &= \phi(-1.12) = 1 - \phi(1.12) = 0.13 \end{aligned}$$

Le taux d'erreur calculé ici par la règle de Bayes est fidèle aux taux d'erreur calculés précédemment pour `Synth1-1000` avec le classifieur euclidien et les K plus proches voisins. On ne peut donc pas conclure sur une règle de décision plus performante car les différences minimes avec le taux d'erreur $\widehat{\varepsilon}$ estimé sont probablement dues aux arrondis et estimations de part et d'autres.

Pour le jeu de de données *Synth2-1000* on ne peut utiliser la formule précédente. On calcule alors une borne supérieur de l'erreur de Bayes :

$$\begin{aligned} & \varepsilon^* \leq \sqrt{\pi_1 \pi_2} \cdot \exp(-\Delta_B^2) \\ \text{avec} \quad & \Delta_B^2 = \frac{1}{8}(\mu_2 - \mu_1)^T (\frac{\Sigma_1 + \Sigma_2}{2})^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \ln \frac{\det \frac{\Sigma_1 + \Sigma_2}{2}}{\sqrt{\det \Sigma_1 \det \Sigma_2}} \\ & \Delta_B^2 = \frac{1}{8} \cdot \frac{25}{3} + \frac{1}{2} \ln(\frac{3}{\sqrt{5}}) = 1.19 \\ & \varepsilon^* \leq \sqrt{0.25} \exp(-1.19) = 0.152 \end{aligned}$$

Le calcul du taux d'erreur de Bayes pour un cas quadratique nécessite l'utilisation de la borne de Bhattacharyya qui donne une borne supérieure du le taux d'erreur ε sur le jeu de données. Cette borne est approximativement deux fois et demi plus grande que le taux d'erreur $\widehat{\varepsilon}$ estimé précédemment (voir Table 1.8 et Table 1.9). Bien que cette borne soit cohérente en majorant nos estimations précédentes, elle ne nous donne qu'un résultat très approximatif. Par conséquent, la probabilité d'erreur de Bayes dans ce cas quadratique ne nous permet pas, à elle seul, de conclure efficacement sur le taux d'erreur de ce jeu de données.

Conclusion

Ce TP nous a permis d'écrire et d'utiliser des scripts `R` pouvant estimer des paramètres, classifier des données et estimer un taux d'erreur pour ce classifieur. Ces scripts ont pu être testés et vérifiés à travers des jeux de données abstraits (`Synth1-N`, `Synth2-1000`) ou réels (`Pima` et `Beastcancer`). L'estimation des taux d'erreurs pour le classifieur euclidien et les K plus proches voisins nous a notamment montré que les résultats étaient meilleurs pour les données d'apprentissage que pour pour les données de test. L'application de la règle de Bayes à travers les jeux de données `Synth1-N` et `Synth2-1000` nous a permis de déterminer et tracer les équations des frontières entre chaque classe des jeux de données. Enfin, le calcul du taux d'erreur de Bayes s'est avéré concluant pour le cas linéaire et, au contraire, très approximatif dans le cas quadratique.