TP3 - Discrimination, théorie bayésienne de la décision

Chang QI (chang.qi@etu.utc.fr)

Valentin MONTUPET (valentin.montupet@etu.utc.fr)

UV : **SY09**

Semestre : **P17**

1. Classifieur euclidien, K plus proches voisins

Dans ce chapitre, nous souhaitons étudier la performance de deux classifieurs sur différents jeux de données. Pour chacun d'entre eux, les individus seront répartis dans 2 classes.

1.1 Programmation

1.1.1 Classifieur euclidien

Le fonctionnement de ce classifieur est simple : Étant donnés un nombre d'individus N_{app} décrit chacun par p **variables aléatoires**, et le **vecteur zapp composé des étiquettes des individus**, il s'agit d'apprendre de ces données afin d'obtenir une frontière de décision dans le but de prédire l'étiquette d'un éventuel nouvel individu.

Pour ce faire, il s'agit de calcul le centre d'inertie dans l'espace R^p de tous les points appartenant à chaque classe. C'est ce que fait la fonction *ceuc.app* : En prenant en entrée le tableau individus variables **Xapp** et le vecteur étiquettes **zapp**, la fonction nous retourne les centres d'inerties des deux classes dans une matrice *mu*. Ceci va nous permettre de partitionner l'espace de représentation en deux régions de décisions, séparées par une frontière de décision qui est l'hyperplan médiateur du segment joignant les centres de gravité des deux classes.

Dès lors, la fonction *ceuc.val* se charge de prédire les étiquettes des individus test **Xtst** étant donné la matrice *mu*. Pour chaque individu X, il s'agit de calculer sa distance aux deux de gravité $g1$ et $g2$. Si le vecteur X est plus proche de $g1$ alors il est affecté à la classe 1, autrement il est affecté à la classe 2.

Connaissant à l'avance la véritable étiquette de ces individus, différents des individus **Xapp**, on pourra juger de la performance du classifieur en évaluant sa précision et son rappel. Si le classifieur est performant, on pourra ainsi prédire efficacement l'étiquette d'un nouvel individu.

La frontière de décision obtenue sur le jeu de données Synth1-40 est exposée figure 1.1 et peut être assimilée à une droite composée des points à équidistance des centres de gravité des deux classes.

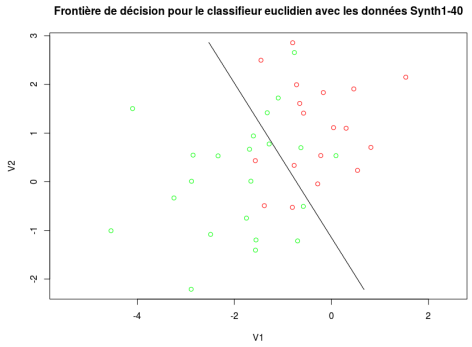

FIGURE 1.1

1.1.2 Cas du classifieur K plus proches voisins

La discrimination en K plus proche voisins consiste à affecter le vecteur X à la classe la plus représentée parmi celles de ses K plus proches voisins. En effet, on calcul la distance euclidienne de X par rapport à ses K plus proches voisins, et on détermine ainsi sa classe en analysant la classe la plus représentée parmi ces plus proches voisins. Pour se faire nous devons créer deux fonctions *kppv.tune*, qui déterminera le nombre K optimal qui minimise l'erreur et *kppv.val* qui permettra de classer l'ensemble de test **Xtst** composé d'individus-variables.

La fonction *kppv.val* prend en entrée l'ensemble d'apprentissage **Xapp** et leurs étiquettes associées **zapp**, une valeur de K et l'ensemble de test **Xtst**. Pour chaque X dans **Xtst**, la fonction extrait les classes de ses K plus proches voisins et attribue comme étiquette à X la classe la plus représentée parmi ses voisins. A la fin, la fonction retourne un vecteur de taille N_{tst} correspondant aux étiquettes prédites des individus de **Xtst**.

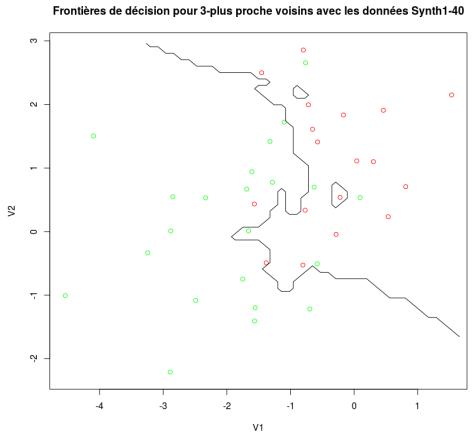

FIGURE 1.2

La fonction *kppv.tune* prend en entrée l'ensemble d'apprentissage **Xapp** et leurs étiquettes associées **zapp**, l'ensemble de validation croisée **Xval** et leurs étiquettes **zval** et une liste de valeur de K à tester **nppv**. Pour chaque valeur K dans **nppv**, la fonction fait appel à *kppv.val* avec l'ensemble d'apprentissage **Xapp**, la valeur K est l'ensemble de validation croisée **Xval** afin de récupérer les étiquettes prédites pour K=i $mu[i]$. Il suffit alors, pour chaque vecteur i d'estimer l'erreur entre $mu[i]$ et **zval**, et de garder la valeur de K qui minimise cette erreur. Le code des 4 fonctions peut être trouvé en Annexe.

1.2 Évaluation des performances

Lors de l'implémentation d'un algorithme d'apprentissage supervisé, il est nécessaire de bien découper l'ensemble du jeu de données en plusieurs parties distinctes aléatoires : l'ensemble d'apprentissage, l'ensemble de test et l'ensemble de validation croisée.

- **L'ensemble d'apprentissage** est composé d'individus aléatoirement sélectionnés dans le dataset original où l'on connaît à l'avance l'étiquette associée. Il constitue la base de connaissance pour apprendre les paramètres du modèle,
- **L'ensemble de validation croisée** sert à estimer les hyper-paramètres du modèle, comme par exemple le nombre K de voisins à prendre en compte, afin de prévenir les soucis de suraprentissage par exemple,
- **L'ensemble de test** est utilisé pour vérifier les performances d'un classifieur. Il n'est jamais utilisé pendant le processus d'apprentissage de manière à avoir une estimation non biaisée de l'erreur sur l'ensemble de test

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

Pour commencer, calculons pour chaque jeu de données $\boldsymbol{\mu}_k$ et $\boldsymbol{\Sigma}_k$ qui sont respectivement la moyenne et la variance de la classe k, ainsi que les proportions π_k de chaque classe. Les résultats sont exposés dans le tableau ci-dessous.

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
$\boldsymbol{\mu}_1$	(-0.316 1.092)	(0.026 0.815)	(0.132 0.880)	(-0.013 0.916)
$\boldsymbol{\mu}_2$	(-1.883 0.105)	(-1.965 -0.127)	(-1.883 -0.085)	(-1.961 0.018)
$\boldsymbol{\Sigma}_1$	$\begin{bmatrix} 0.682 & 0.119 \\ 0.119 & 1.012 \end{bmatrix}$	$\begin{bmatrix} 0.882 & -0.131 \\ -0.131 & 1.109 \end{bmatrix}$	$\begin{bmatrix} 1.050 & 0.052 \\ 0.052 & 0.984 \end{bmatrix}$	$\begin{bmatrix} 0.969 & -0.065 \\ -0.065 & 1.079 \end{bmatrix}$
$\boldsymbol{\Sigma}_2$	$\begin{bmatrix} 1.375 & 0.323 \\ 0.323 & 1.439 \end{bmatrix}$	$\begin{bmatrix} 0.757 & -0.038 \\ -0.038 & 0.761 \end{bmatrix}$	$\begin{bmatrix} 0.967 & -0.111 \\ -0.111 & 0.979 \end{bmatrix}$	$\begin{bmatrix} 0.990 & 0.021 \\ 0.021 & 0.941 \end{bmatrix}$
π_1	0.45	0.54	0.53	0.50
π_2	0.55	0.46	0.47	0.50

TABLE 1.1 – Estimations des paramètres $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ et π_k pour les différents jeux de données

D'après le tableau 1.1, on peut constater que les proportions π_k tournent autour de 50% pour chaque classe, $\boldsymbol{\mu}_1$ semble être proche de $\begin{pmatrix} 0 & 1 \end{pmatrix}$, $\boldsymbol{\mu}_2$ proche de $\begin{pmatrix} -2 & 0 \end{pmatrix}$ et que les variances $\boldsymbol{\Sigma}_k$ sont presque diagonales : on peut donc affirmer que les nuages de points sont de même volume et plus ou moins sphériques. De plus, en observant les $\boldsymbol{\mu}_k$ qui sont les centres d'inerties des classes, on constate que ces derniers sont relativement proches l'un de l'autre : on peut imaginer que certains points seront alors difficiles à classer pour les deux classifieurs, le taux d'erreur aura ainsi tendance à être plus élevé.

Afin d'évaluer les performances des deux classifieurs, on veut estimer le taux d'erreur ϵ générés par ces derniers. Pour ce faire, on répète N = 20 fois une séparation aléatoire du jeu de données pour former un ensemble d'apprentissage et un ensemble de test et de manière optionnelle, un ensemble de validation. On peut ensuite, pour chaque séparation, apprendre les paramètres du modèle sur l'ensemble d'apprentissage formé et calculer le taux d'erreur obtenu sur l'ensemble de test. On peut ainsi recueillir un échantillon de 20 estimations du taux d'erreur et alors déterminer une estimation ponctuelle de ϵ (moyenne) et un intervalle de confiance sur ϵ.

L'intervalle de confiance sur ϵ est construit selon la formule suivante, supposant que l'on se trouve avec une population suivant une loi normale :

$$]\bar{x} - 1,96\frac{\boldsymbol{\Sigma}(X)}{\sqrt{n}}, \bar{x} + 1,96\frac{\boldsymbol{\Sigma}(X)}{\sqrt{n}}[$$

pour un intervalle de confiance à 95%

Nous remarquons que les résultats sont similaires pour l'ensemble d'apprentissage et de test. De même, les tendances des résultats sont identiques pour les deux classifieurs : plus il y a de données, plus les classifieurs semblent fiables. Cela s'explique par un plus grand nombre d'individus d'apprentissage et donc une meilleure compréhension du modèle et un meilleur calcul des paramètres. Remarquons également que pour de faibles volumes de données (inférieurs à 100), le classifieur euclidien est un peu moins performant. En effet, ce dernier est performant pour des groupes homogènes et sphériques, cette dernière condition n'étant pas totalement vraie pour ces données.

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
ϵ ensemble d'apprentissage	0.207	0.084	0.131	0.142
IC apprentissage ($\alpha = 0,95$)	[0.192, 0.230]	[0.082, 0.102]	[0.127, 0.135]	[0.140, 0.148]
ϵ ensemble de test	0.227	0.100	0.138	0.140
IC test ($\alpha = 0,95$)	[0.167, 0.264]	[0.074, 0.108]	[0.127,0.142]	[0.135,0.152]

TABLE 1.2 – Estimations ponctuelles du taux d'erreur ϵ et intervalle de confiance pour l'ensemble d'apprentissage et de test avec le classifieur euclidien

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
ϵ ensemble d'apprentissage	0.550	0.044	0.111	0.128
IC apprentissage ($\alpha = 0,95$)	[0.365, 0.735]	[0.026, 0.062]	[0.097, 0.124]	[0.122, 0.133]
ϵ ensemble de test	0.505	0.188	0.160	0.162
IC test ($\alpha = 0,95$)	[0.354, 0.656]	[0.099, 0.276]	[0.149, 0.171]	[0.148, 0.176]

TABLE 1.3 – Estimations ponctuelles du taux d'erreur ϵ et intervalle de confiance pour l'ensemble d'apprentissage et de test avec le classifieur KPPV

Si l'on effectue une séparation aléatoire des données en un ensemble d'apprentissage et un ensemble de test, et que l'on souhaite déterminer le nombre optimal de voisins à l'aide de la fonction *kppv.tune*, en utilisant l'ensemble d'apprentissage comme ensemble de validation, le **nombre optimal de voisin déterminé est 1**. Ce résultat n'est pas surprenant : en effet le modèle a été appris sur les données Xapp, le tester avec les mêmes données résulte à choisir le plus proche voisin à savoir lui-même est obtenir dès lors 100% de précision, mais ce résultat ne signifie rien : c'est le **suraprentissage** ! En d'autres termes, cela revient à donner un ensemble de questions à des étudiants, leur faire apprendre les réponses par coeur, et à reposer exactement les mêmes questions le jour de l'examen. C'est pourquoi il est nécessaire de différencier les données d'apprentissage des données de test pour évaluer la performance d'un classifieur, et d'avoir parfois recours à plusieurs procédés pour éviter le surapprentissage comme la validation croisée ou bien la régularisation.

1.2.2 Jeu de données Synth2-1000

Nous considérons désormais le jeu de données $Synth2-1000$ qui a une distribution différente des autres jeux précédents. Afin d'estimer la performance des classifieurs sur ce jeu de données, estimons au préalable les valeurs $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ et π_k. On peut constater plusieurs différences table 1.4 vis à vis des jeux de données précédents :

Tout d'abord, de par l'analyse des vecteurs $\boldsymbol{\mu}_1$ et $\boldsymbol{\mu}_2$, les centres d'inertie sont beaucoup plus éloignés l'un de l'autre que dans la distribution précédente. De plus, alors que le nuage de points de la classe 1 semble relativement sphérique, celui de la classe 2 semble plus allongé, comme en témoigne $\boldsymbol{\Sigma}_2$, qui n'est pas de la forme $\left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$.

Comme nous pouvons le constater avec la table 1.9, les résultats obtenus sont meilleurs qu'avec les différents jeux de la distribution précédente : l'estimation ponctuelle du taux d'erreur ϵ est très faible

	Classe 1	Classe 2
$\boldsymbol{\mu}_k$	(3.018 -0.006)	(-2.142 -0.026)
$\boldsymbol{\Sigma}_k$	$\begin{bmatrix} 0.990 & 0.113 \\ 0.113 & 1.093 \end{bmatrix}$	$\begin{bmatrix} 4.435 & -0.154 \\ -0.154 & 1.031 \end{bmatrix}$
π_k	0.52	0.48

TABLE 1.4 – Estimations des paramètres $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ et π_k pour le jeu de données Synth2-1000

	Euclidien	KPPV
ϵ ensemble d'apprentissage	0.063	0.048
IC apprentissage ($\alpha = 0,95$)	[0.061, 0.065]	[0.042, 0.054]
ϵ ensemble de test	0.061	0.066
IC test ($\alpha = 0,95$)	[0.056, 0.067]	[0.060, 0.071]

TABLE 1.5 – Estimations ponctuelles du taux d'erreur ϵ et intervalle de confiance pour l'ensemble d'apprentissage et de test sur Synth2-1000

et les intervalles de confiance sur ϵ sont petits. Ce résultat n'est pas surprenant si l'on se réfère aux constatations précédentes : les nuages de points des deux classes étant davantage éloignés, les classifieurs euclidien et KPPV réussissent à classer les individus dans la classe correspondante.

1.2.3 Jeux de données réelles : Pima & Breastcancer

Pima

	Pima
$\boldsymbol{\mu}_1$	(2.927 110.017 69.913 27.290 31.430 0.446 29.222)
$\boldsymbol{\mu}_2$	(4.700 143.119 74.700 32.977 35.820 0.617 36.412)
$\boldsymbol{\Sigma}_1$	7x7 matrix
$\boldsymbol{\Sigma}_2$	7x7 matrix
π_1	0,67
π_2	0,33

TABLE 1.6 – Estimations des paramètres $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ et π_k pour les données Pima

En réalisant les mêmes étapes que précédemment, nous pouvons déduire certaines choses : tout d'abord, les proportions des classes sont très différentes : la classe 1 représente quasiment 70% de l'effectif total du jeu de données. Cela peut influencer négativement sur la performance des classifieurs car le nuage de point de la classe 1 sera beaucoup plus gros que celui de la classe 2. Dans le cas du classifieur euclidien, la frontière de décision pourrait couper le nuage de point de la classe 1. Ensuite, on constate que les coefficients non diagonaux des matrices de covariances ne sont plus nuls mais les valeurs sur les diagonales représentent respectivement 59% et 78% pour $\boldsymbol{\Sigma}_1$ et $\boldsymbol{\Sigma}_2$. De plus, certaines valeurs diagonales sont très élevées, traduisant d'une variance forte pour les variables correspondantes.

	Euclidien	KPPV
ϵ ensemble d'apprentissage	0.240	0.185
IC apprentissage ($\alpha = 0,95$)	[0.233, 0.246]	[0.167, 0.203]
ϵ ensemble de test	0.254	0.249
IC test ($\alpha = 0,95$)	[0.242, 0.264]	[0.239, 0.259]

TABLE 1.7 – Estimations ponctuelles du taux d'erreur ϵ et intervalle de confiance pour l'ensemble d'apprentissage et de test sur Pima.

Ainsi, on obtient des résultats similaires pour les deux classifieurs qui sont peu satisfaisants : l'estimation ponctuelle du taux d'erreur est de l'autre de 25% pour l'ensemble de test. Cela était prévisible au vu de la complexité des données accrues par rapport aux jeux de données précédents.

Breastcancer

	Breastcancer
$\boldsymbol{\mu}_1$	(2.96 1.31 1.41 1.35 2.11 2.41 2.08 1.26 1.06)
$\boldsymbol{\mu}_2$	(7.19 6.58 6.56 5.59 5.33 4.71 5.97 5.86 2.60)
$\boldsymbol{\Sigma}_1$	9x9 matrix
$\boldsymbol{\Sigma}_2$	9x9 matrix
π_1	0,65
π_2	0,35

TABLE 1.8 – Estimations des paramètres $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ et π_k pour les données Breastcancer

	Euclidien	KPPV
ϵ ensemble d'apprentissage	0.038	0.014
IC apprentissage ($\alpha = 0,95$)	[0.036, 0.040]	[0.010, 0.019]
ϵ ensemble de test	0.039	0.043
IC test ($\alpha = 0,95$)	[0.035, 0.042]	[0.038, 0.049]

TABLE 1.9 – Estimations ponctuelles du taux d'erreur ϵ et intervalle de confiance pour l'ensemble d'apprentissage et de test sur Breastcancer.

Là encore, les proportions des classes sont très différentes : 65% de l'effectif total du jeu de données pour la classe 1 contre 35% pour la classe 2. Cela peut avoir la même conséquence sur la frontière de décision que pour Pima. En revanche, les centroïdes des deux classes semblent beaucoup plus éloignés que pour Pima, « facilitant la tâche » du classifieur Ensuite, les matrices de covariances ne sont pas diagonales et les valeurs sur les diagonales représentent respectivement 36% et 43% pour $\boldsymbol{\Sigma}_1$ et $\boldsymbol{\Sigma}_2$.

On obtient pour ce jeu de données d'excellents résultats avec un taux d'erreur estimé à environ 4% pour l'ensemble de test pour les deux classifieurs et même 1% pour l'ensemble d'apprentissage avec le classifieur Kppv ! Les intervalles de confiance sur ϵ sont eux-aussi très petits, confortant notre excellent

résultat. Celà peut s'expliquer par le fait que les classes sont très clairement distinctes, avec pour preuve des centroïdes de classes lointains entre eux.

1.3 Règle de Bayes

Nous savons d'après l'énoncé que la classe ω_k suit une loi normale bivariée de paramètres $\boldsymbol{\mu}_k = (\boldsymbol{\mu}_{k1}, \boldsymbol{\mu}_{k2})^T$ et $\boldsymbol{\Sigma}_k = diag(\boldsymbol{\Sigma}^2_{k1}, \boldsymbol{\Sigma}^2_{k2})$ pour k = 1, 2. De plus, les $\boldsymbol{\Sigma}_k$ sont définies positives, donc inversibles, et diagonales, donc les variables sont supposées indépendantes.

1.3.1 Distributions marginales

Nous cherchons les distributions marginales des variables $X^1 et X^2$. Tout d'abord, la densité de probabilité de la loi normale bivariée s'écrit :

$$f(x) = \frac{1}{(2\pi)^{N/2} det(\boldsymbol{\Sigma})^{1/2}} e^{-\frac{1}{2}(x-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})}$$

On sait aussi que tout sous-vecteur d'un vecteur aléatoire gaussien suit une loi gaussienne, et qu'en particulier ses composantes sont toutes gaussiennes. Par conséquent :

$$f_1(x) = \frac{1}{(2\pi)^{N/2} det(\boldsymbol{\Sigma}_1)^{1/2}} e^{-\frac{1}{2}(x-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1}(x-\boldsymbol{\mu}_1)}$$

$$f_1(x) = \frac{1}{2\pi} e^{-\frac{1}{2}(x_1^2 + (x_2-1)^2)} \tag{1.1}$$

$$f_2(x) = \frac{1}{(2\pi)^{N/2} det(\boldsymbol{\Sigma}_2)^{1/2}} e^{-\frac{1}{2}(x-\boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1}(x-\boldsymbol{\mu}_2)}$$

$$f_2(x) = \frac{1}{2\pi} e^{-\frac{1}{2}((x_1+2)^2 + x_2^2)} \tag{1.2}$$

Dès lors, on a $X^1_{\omega 1} \sim N(0,1)$, $X^1_{\omega 2} \sim N(1,1)$, $X^2_{\omega 1} \sim N(-2,1)$, $X^2_{\omega 2} \sim N(0,1)$

De la même manière pour **Synth2-1000**, on obtient :

$$f_1(x) = \frac{1}{2\pi} e^{-\frac{1}{2}((x_1-3)^2 + x_2^2)} \tag{1.3}$$

$$f_2(x) = \frac{1}{(2\pi)\sqrt{5}} e^{-\frac{1}{2}(\frac{(x_1+2)^2}{5} + x_2^2)} \tag{1.4}$$

D'où $X^1_{\omega 1} \sim N(3,1)$, $X^1_{\omega 2} \sim N(0,1)$, $X^2_{\omega 1} \sim N(-2,5)$, $X^2_{\omega 2} \sim N(0,1)$

1.3.2 Courbes d'iso-densité

Les courbes d'iso-densité sont définies telles que la densité est égale en tout point de chaque courbe. Autrement dit, on a $f_k(x) = c_k$ pour k=1,2.

$$\frac{1}{(2\pi)^{N/2} det(\boldsymbol{\Sigma}_k)^{1/2}} e^{-\frac{1}{2}(x-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(x-\boldsymbol{\mu}_k)} = c_k$$

On remarque que $(x-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1}(x-\boldsymbol{\mu}_1) = (k_1)^2$ et
$(x-\boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1}(x-\boldsymbol{\mu}_2) = (k_2)^2$ avec k_1, k_2 des constantes telles que $k_i = -2ln(2\pi c_i)$ pour i=1,2.

Pour la distribution **Synth1** on en déduit :

$$(x - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1} (x - \boldsymbol{\mu}_1) = (k_1)^2$$

$$x_1^2 + (x_2 - 1)^2 = (k_1)^2 \tag{1.5}$$

$$(x - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (x - \boldsymbol{\mu}_2) = (k_2)^2$$

$$(x_1 + 2)^2 + x_2^2 = (k_2)^2 \tag{1.6}$$

Pour la classe ω_1 les courbes d'iso-densité sont les cercles (ou hypersphères de dimension 2) ayant pour centre $(0, 1)$ et pour rayon k_1, avec k_1 constante. Pour la classe ω_2 les courbes d'iso-densité sont les cercles ayant pour centre $(-2, 0)$ et pour rayon k_2, avec k_2 constante.

Pour la distribution **Synth2**, de la même manière :

$$(x_1 - 3)^2 + (x_2)^2 = (k_1)^2 \tag{1.7}$$

$$\frac{(x_1 + 2)^2}{5} + x_2^2 = (k_2)^2 \tag{1.8}$$

Pour la classe ω_1 les courbes d'iso-densité sont les cercles ayant pour centre $(3, 0)$ et pour rayon k_1, avec $k_1 = -2ln(2\pi c_i)$ constante. Pour la classe ω_2 les courbes d'iso-densité sont les ellipsoïdes ayant pour centre $(-2, 0)$ et pour rayon k_2, avec k_2 constante.

1.3.3 Expression de la règle de Bayes frontières de décision

Par définition, la règle de Bayes peut s'écrire

$$\delta^* = \begin{cases} \omega_1 & si \frac{f_1(x)}{f_2(x)} > \frac{\pi_2}{\pi_1} \\ \omega_2 & sinon \end{cases}$$

Or, nous sommes dans le cas ou $\pi_1 = \pi_2 = 0,5$, le rapport $\frac{\pi_2}{\pi_1}$ est donc égal à 1 Pour les données **Synth1-n**, on en déduit :

$$\begin{aligned} & \frac{\frac{1}{2\pi} e^{-\frac{1}{2}(x_1^2 + (x_2-1)^2)}}{\frac{1}{2\pi} e^{-\frac{1}{2}((x_1+2)^2 + x_2^2)}} > 1 \\ \iff & \frac{1}{2\pi} e^{-\frac{1}{2}(x_1^2 + (x_2-1)^2)} > \frac{1}{2\pi} e^{-\frac{1}{2}((x_1+2)^2 + x_2^2)} \\ \iff & x_1^2 + (x_2 - 1)^2 > (x_1 + 2)^2 + x_2^2 \\ \iff & 2x_1 + x_2 + \frac{3}{2} < 0 \end{aligned}$$

On obtient donc la règle de Bayes suivante :

$$\delta^* = \begin{cases} \omega_1 & si\ 2x_1 + x_2 + \frac{3}{2} < 0 \\ \omega_2 & sinon \end{cases}$$

En vérifiant avec des individus issus de la distribution **Synth1**, on s'aperçoit que la règle de Bayes semble cohérente et on a ω_1 la classe des individus étiquetés 2 et ω_2 la classe des individus étiquetés 1. Les données Synth1-1000 sont représentées avec la frontière de décision obtenue figure 1.3

De la même façon, pour **Synth2** :

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

$$4x_1^2 - 34x_1 + 41 - 5ln(5) < 0$$

D'où la règle de Bayes :

$$\delta^* = \begin{cases} \omega_1 & si\ 4x_1^2 - 34x_1 + 41 - 5ln(5) < 0 \\ \omega_2 & sinon \end{cases}$$

En vérifiant avec les données, on observe que ω_1 est cette fois l'ensemble des individus étiquetés 1, ω_2 l'ensemble des individus étiquetés 2. La frontière de décision ainsi que le jeu de données **Synth2-1000** sont représentés figure 1.4

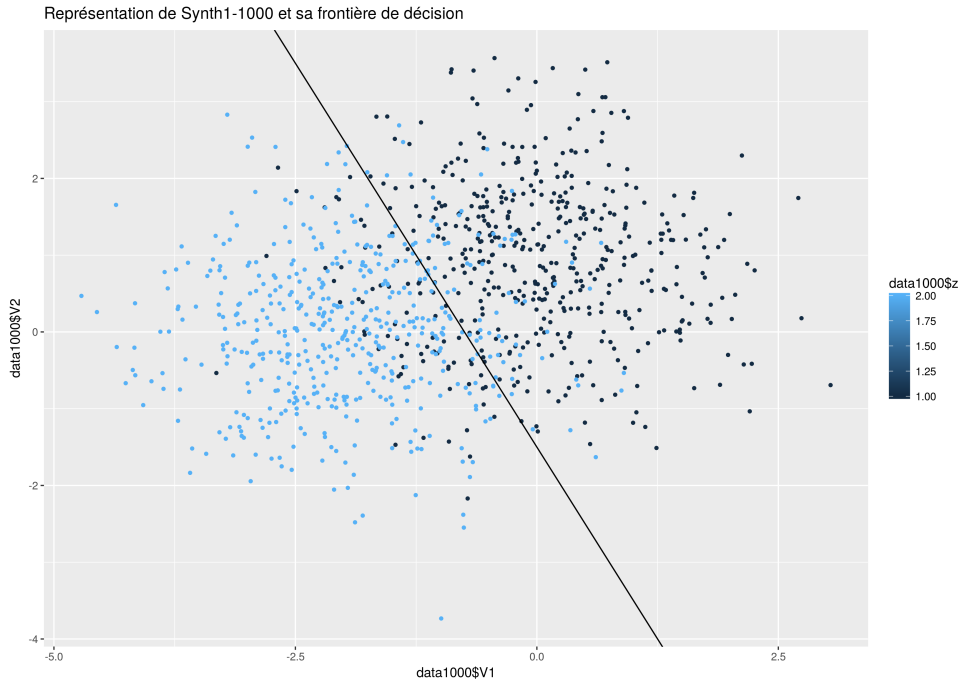

FIGURE 1.3 – Représentation de la frontière de décision $2x_1 + x_2 + \frac{3}{2} = 0$ et des données Synth1-1000

1.3.4 Erreur de Bayes

Dans le cas de **Synth1-n**, nous sommes dans les conditions d'homoscédasticité et $\pi_1 = \pi_2 = 0,5$. L'erreur de Bayes s'écrit donc :

$$\epsilon^* = \phi(-\frac{\Delta}{2})$$

avec $\Delta = \sqrt{(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)}$

Ainsi :

$$\epsilon^* = \phi(-\frac{\sqrt{5}}{2}) \simeq 1 - 0,8686 \simeq 0.1314$$

On obtient alors un taux d'erreur du même ordre que ceux obtenus avec les classifieurs euclidien et KPPV. Ceci conforte donc nos résultats précédents.

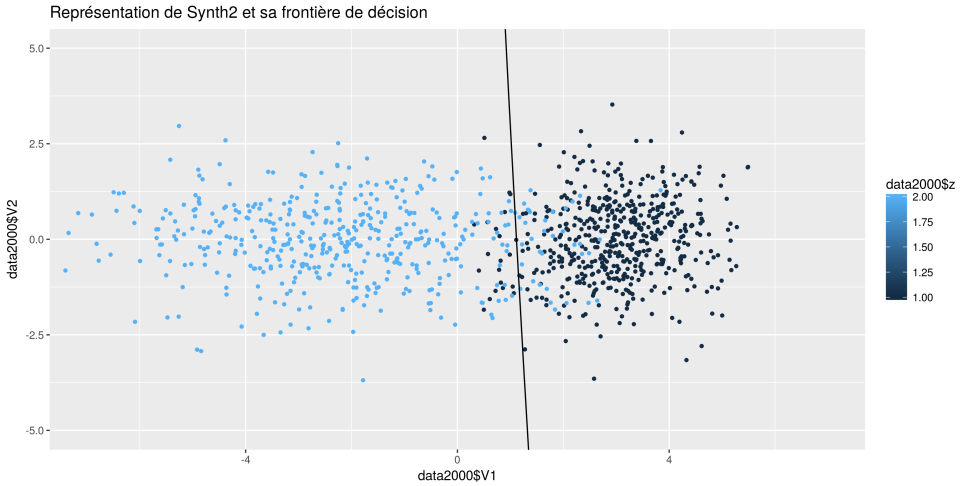

FIGURE 1.4 – Représentation de la frontière de décision $4x_1^2 - 34x_1 + 41 - 5ln(5) = 0$ et des données Synth2-1000

Dans le cas de **Synth2-1000**, nous ne pouvons pas exprimer analytiquement l'erreur de Bayes car l'homoscédasticité n'est pas respectée. Nous pouvons seulement en donner une borne maximale, appelée borne de Bhattacharyya :

$$\epsilon^* \leq \sqrt{\pi_1 \pi_2} e^{-\Delta_B^2}$$

avec $\Delta_B^2 = \frac{1}{8}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T (\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2})^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) + \frac{1}{2} ln(\frac{det \frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}}{\sqrt{det \boldsymbol{\Sigma}_1 det \boldsymbol{\Sigma}_2}}) \simeq 1,189$

Ainsi, on obtient $\epsilon^* \leq 0.1522$, ce qui est un peu plus élevé que ce que nous avions trouvé précédemment.

.1 Implémentation du classifieur euclidien

```
ceuc.app <- function(Xapp, zapp){
  #Getting classes
  classes <- unique(zapp)
  g <- length(classes)

  #Getting Xapp size
  n<-dim(Xapp)[1]
  p<-dim(Xapp)[2]

  # classAppartenance(i,j) = 1 if Xapp[i] is in class g, 0 otherwise
  classAppartenance <- matrix(0,n,g)
  for(i in 1:n){
    classAppartenance[i, zapp[i]] <- 1
  }

  mu <- matrix(0,g,p)
  for(k in 1:g){
    center<- apply((Xapp*classAppartenance[,k]),2,sum)
    center <- center/sum(classAppartenance[,k])
    mu[k,]<-center
  }
  mu
}
```

.2 Implémentation du classifieur KPPV

```
kppv.val <- function(Xapp, zapp, K, Xtst){
  count <- rep(0,K)
  ntst <- dim(Xtst)[1]
  predicted <- rep(0,ntst) #Predicted labels to be returned
  neighbours <- matrix(0,ntst,K) # for storing the K nearest neighbours of X

  dist <- distXY(as.matrix(Xtst),as.matrix(Xapp))

  classement <- t(apply(dist, 1, rank))

  for(i in 1:ntst){
    neighbours[i,] <- sort.int(classement[i,], index.return = TRUE)$ix[1:K]
  }
  for(i in 1:ntst){
    for(j in 1:K){
```

```
      count[zapp[neighbours[i,j]]] =  count[zapp[neighbours[i,j]]] + 1
    }
    predicted[i] <- which.max(count)
    count <- rep(0,K)
  }
  predicted
}

kppv.tune <- function(Xapp, zapp, Xval, zval, nppv)
{
  nb_test<-length(zval)
  nb_k <- length(nppv)
  resultat <- matrix(0, nb_k, nb_test)
  error<-array(0,nb_k)
  for(i in nppv){
    resultat[i,] <- kppv.val(Xapp, zapp, i, Xval)
  }

  for (i in nppv){
    for (j in 1:nb_test){
      if(zval[j] != resultat[i,j]){
        error[i] = error[i] + 1
      }
    }
  }
  which.min(error)
}
```