Compte-rendu du TP03 de SY09
Discrimination, théorie bayésienne de la décision

NGO Sy-Toan, Elliot BARTHOLME

2 juin 2017

1 Introduction

Ce nouveau Travail Pratique présentera des concepts d'apprentissage supervisé, visant à construire des systèmes ou modèles de prédiction à partir d'observations. Il s'agit de déterminer une variable à expliquer (groupe ou classe d'individus) en observant des variables explicatives sur un autre ensemble.

Pour cela, nous étudierons deux classifieurs simples, le classifieur euclidien et la méthode des K plus proches voisins, qui permettent de prédire les étiquettes d'individus grâce à l'observation de caractéristiques dans un groupe d'apprentissage. Nous étudierons ensuite les performances respectives de ces classifieurs sur différentes données, puis terminerons par une application de la règle de Bayes à notre étude.

2 Classifieur euclidien, K plus proches voisins

Dans cette partie nous allons étudier deux classifieurs. Nous allons programmer des fonctions en R permettant d'utiliser ces derniers.

4 algorithmes seront définis, afin de passer outre toute lecture de code source, et en anglais. Un ensemble de commentaires se trouveront en dessous de chacun pour faciliter la compréhension du lecteur.

2.1 Classifieur euclidien

Le classifieur euclidien affecte le vecteur x au groupe dont le centre de gravité est le plus proche. La règle de décision est donc la suivante :

$$\delta(x) = \begin{cases} a_1 & si\ d(x, \overline{x_1}) \leq d(x, \overline{x_2}) \\ a_2 & sinon, \end{cases} \tag{1}$$

Pour calculer $\overline{x_k}$ le centre du nuage correspondant à la classe w_k :

$$\overline{x_k} = \frac{1}{n_k} \sum_{i=1}^{n} z_i k * x_i$$

où $z_i k = 1$ si $x_i \in w_k, z_i k = 0$ sinon. Voici l'algorithme défini pour apprendre les paramètres du classifieur euclidien (centres de gravité) :

Algorithm 1: Learning of euclidean classifier parameters

function ceuc.app(X_{app}, z_{app});
Input : X_{app} : matrix $n_{app} \times p$ of learning individuals group
z_{app} : vector of length n_{app} containing each label (class) of those individuals
Output: Estimated parameters from euclidean classifier : matrix $g \times p$ of gravity centers for each class' scatter plot
g = nb different classes (indices 1..g)
μ = matrix $g \times p$ with p = dimension of each indivdual (variables count)
for $i \leftarrow 1$ *to* g **do**
  $\omega_i \leftarrow i^{th}$ class from unique classes index
  $X_i \leftarrow$ individuals from X_{app} for which $z_{app_i} = i$ $(\omega(x_i) = \omega_i)$
  $\mu = rowbind\Big[\mu,\ mean(X_i)$ for each column (coordinates of the gravity center for class $\omega_i)\Big]$
end
return μ

Le règle de décision $\sigma(\mathbf{x})$, qui va étiquetter un individu X si avec la classe du centre de gravité le plus proche est définie par l'algorithme suivant :

Algorithm 2: Result of ranking with euclidean classifier

function ceuc.val(μ, X_{tst})
Input : μ : matrix of euclidean parameters (gravity centers)
X_{tst} : matrix $n \times p$ of test individuals group
Output: z_{tst} : vector (length n_{tst}) of test individuals classes
$distfromgravitycenters \leftarrow distXY(\mu, X_{tst})$ [a]
$z_{tst} < -index(min(distfromgravitycenters))$ for each column [b]
return z_{tst}

a. $distXY(X, Y)$ calculates square of euclidean distances between groups X and Y therefore $distfromgravitycenters$ is a matrix of dim $n_{classes} \times n_{indiv}$ of distances between each individual and the euclidean parameters
b. Gets the class associated to the nearest gravity center

Résultat obtenu par avec le classifieur euclidien avec les données "Synth1-40" en utilisant la totalité des données pour l'apprentissage du modèle (voir énoncé Figure 1) :

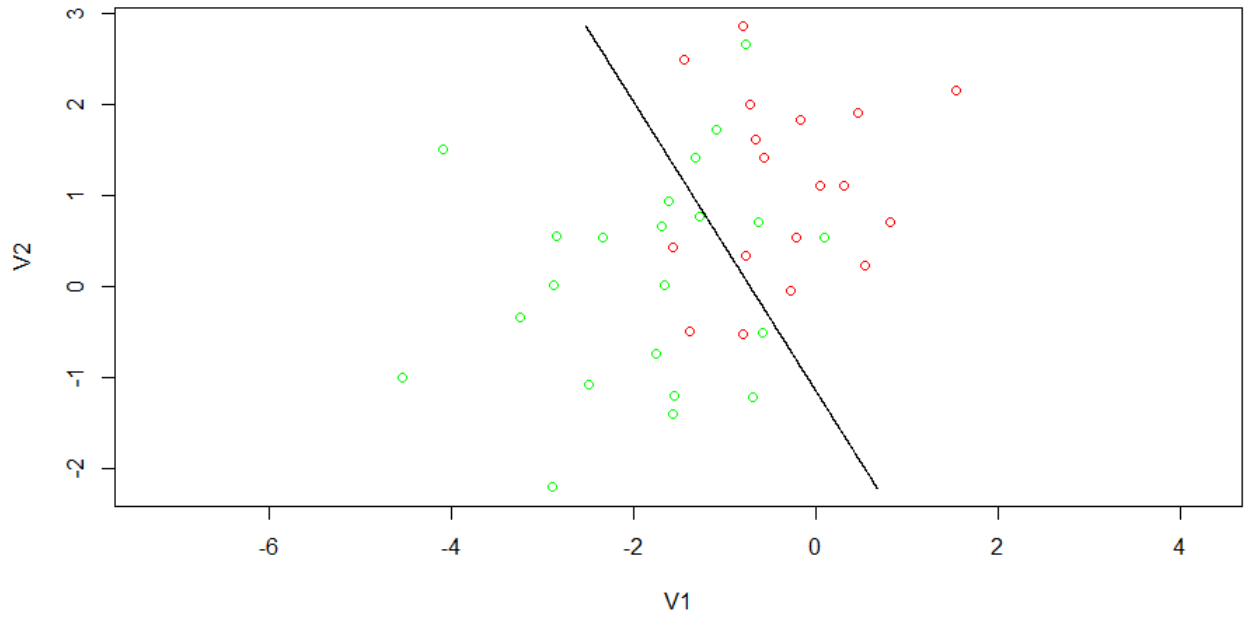

FIGURE 1 – Les données "Synth1-40" séparées par le classifieur euclidien

2.2 K plus proches voisins

Cette règle de décision est différente. Elle consiste à affecter le vecteur x à la classe la plus représentée parmi les K plus proches voisins de x dans l'ensemble d'apprentissage ; la proximité peut notamment être la distance euclidienne. L'ensemble de ces K plus proches voisins étant noté $N_K(x)$, on a la règle de décision suivante :

$$\delta(x) = arg_{k=1,\dots,g} max \quad \frac{1}{K} \sum_{x_i \in N_K(x)} z_{ik} \tag{2}$$

Voici l'algorithme que nous avons fait pour classer un tableau individus-variables X_{tst} grâce à la méthode des K plus proches voisins.

Algorithm 3: Result of ranking with K nearest neighbours method

1 function kppv.val(X_{app}, z_{app}, K, X_{tst})
Input : Labeled data : X_{app} and z_{app}
K : number of neighbours used by the algorithm
X_{tst} : the individuals (group) to evaluate
Output: n_{tst} : vector of labels predicted by the algorithm for the X_{tst} group
2 $distanceTestApp \leftarrow distXY(X_{tst}, X_{app})$ [a]
3 $classes \leftarrow$ indices of classes
4 $indices \leftarrow orderbyrow(distanceTestApp)$
5 $indices \leftarrow Kfirstvaluesfromindices$ [b]
6 $indicesclasses \leftarrow matrix(z_{app}[indices], nrow = nrow(indices))$ [c]
7 $ratio \leftarrow vector$
8 **for** *cl in classes* **do**
9 $presence\_class\_cl \leftarrow sumbycolumn(indicesclasses == cl)/K$
10 $ratio \leftarrow vector(ratio, presence\_class\_cl)$ [d]
11 **end**
12 $ratio \leftarrow matrixfilledbyrow(ratio, nrow = length(classes))$ [e]
13 $labels \leftarrow index(max(ratio))$ for each column [f]
14 **return** *labels*

a. distanceTestApp is a matrix $n_{tst} \times n_{app}$ of squared euclidean distances between test and learning groups
b. These lines 4 and 5 order the matrix distanceTestApp by row and take the K first values so we obtain the K nearest neighbours for each individual from X_{tst} in a matrix $K \times n_{tst}$, naturally considering the euclidean distance
c. Retrieves the classes index of the K nearest neighbours found precedently
d. We sum (indicesclasses matrix by columns) the occurences of each class in the K nearest neighbour of each individual to test
e. Only a R utility : organize the ratios in a matrix rather than vector so we have a dimension $n_{classes} \times n_{tst}$ with the ratio of each class in the K nearest neighbour of each test individual
f. We take the class (label) with the higher ratio for each test individual and return the vector of those labels

L'algorithme suivant va lui déterminer le nombre optimal de voisins à utiliser pour la méthode, en évaluant les performances du classifieur sur un ensemble de données de validation, avec un K différent à chaque fois.

Algorithm 4: Finding the optimal number of neighbours for the method

1 function kppv.tune(X_{app}, z_{app}, X_{val}, z_{val}, $nppv$)
Input : Learning data used for ranking : X_{app} and z_{app}
Validation data used for validation : X_{val} and z_{val}
$nppv$: A set of values corresponding to the different neighbours numbers to try
Output: K_{opt} : value chosen in $nppv$ which gives the best results on the validation set X_{val}
2 $similarities \leftarrow vector$
3 $K_{opt} \leftarrow 0$
4 **for** *K in nppv* **do**
5 $labels \leftarrow kppv.val(X_{app}, z_{app}, K, X_{val})$ [a]
6 $similarities \leftarrow vector(similarities, sum(z_{val} == labels))$ [b]
7 **end**
8 $K_{opt} \leftarrow nppv[index(max(similarities))]$ [c]
9 **return** K_{opt}

a. We use the other algorithm to know the labels that it has determined
b. We store in a vector the correct values labeled by *kppv.val*
c. Gets the smallest index of the max value of similarities vector, then retrieves the K optimal value in nppv list with the index

Résultat obtenu avec la méthode des K plus proches voisins avec les données "Synth1-40" et $K_opt = 1$ en utilisant la totalité des données pour l'apprentissage :

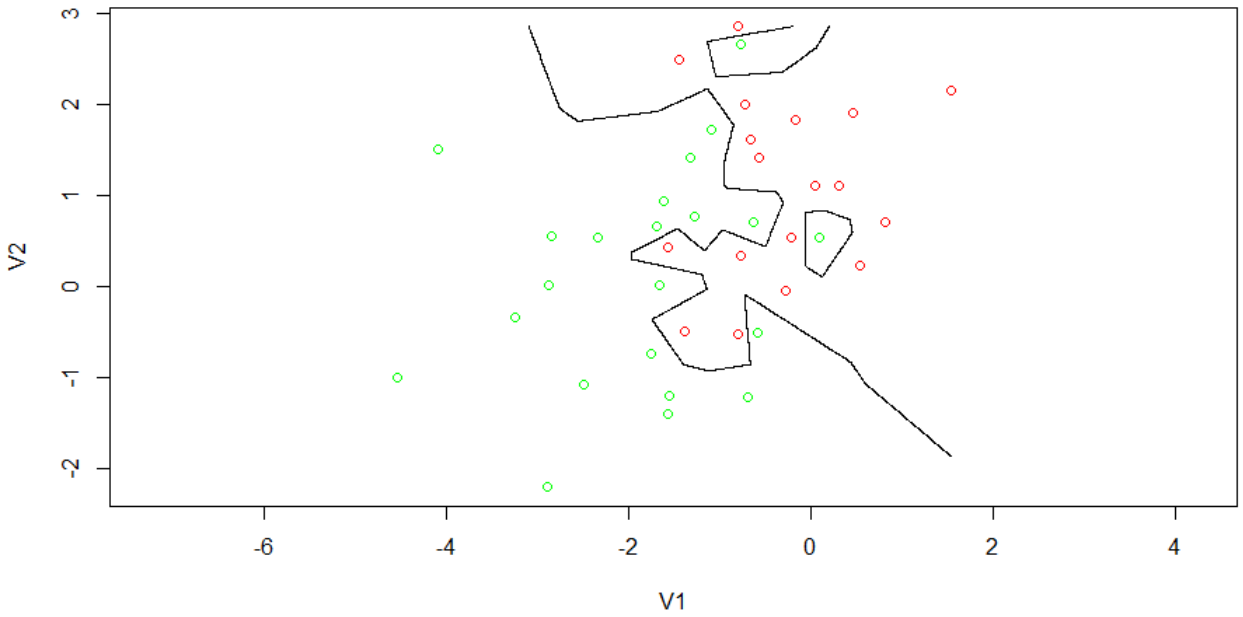

FIGURE 2 – Les données "Synth1-40" classées par la méthode des K plus proches voisins et $K = 1$

Un autre résultat avec $K = 3$:

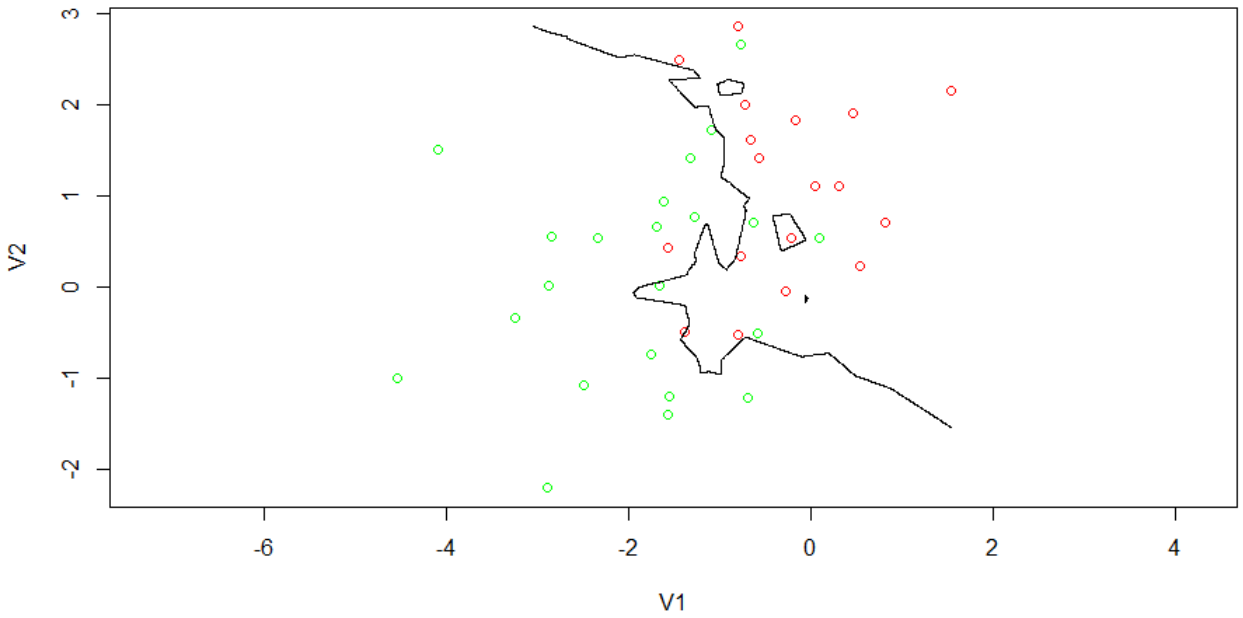

FIGURE 3 – Les données "Synth1-40" classées par la méthode des K plus proches voisins et $K = 3$

3 Évaluation des performances

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

Les classes des différentes observations des données sont distribuées selon une loi normale bivariée.

Si $X_1, ..., X_n$ est un échantillon indépendant et identiquement distribué (*iid*) de vecteur aléatoire parent $X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ alors les estimateurs du maximum de vraisemblance $\hat{\boldsymbol{\mu}}$ et $\hat{\Sigma}$ de $\boldsymbol{\mu}$ et de Σ sont le vecteur moyenne empirique $\overline{\boldsymbol{X}}$ et la matrice de variance empirique V ; on a de plus $\hat{\boldsymbol{\mu}} \sim \mathcal{N}(\boldsymbol{\mu}, \frac{1}{n}\Sigma)$.

Pour chaque jeu de données, nous allons donc estimer les paramètres μ_k et Σ_k des distributions conditionnelles, et la proportion π_k de chacune des classes. On déterminera ensuite le taux d'erreur ε d'apprentissage et de test du classifieur euclidien sur des découpages des données différents.

Pour estimer les paramètres, une fonction *estimationParametres()* a été créée. Celle-ci prend en argument un tableau X individus-variables ainsi qu'un vecteur z d'étiquettes. Elle renvoie une liste de variables correspondant aux paramètres de distribution, ce qui permet d'accéder facilement aux résultats : *resultat$mu, resultat$sigma, resultat$pi*.

Voici les résultats obtenus pour chacun des jeux de données :

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
π_1	0.45	0.54	0.528	0.496
π_2	0.55	0.46	0.472	0.504
μ_1	$\begin{pmatrix} -0.32 \\ 1.09 \end{pmatrix}$	$\begin{pmatrix} 0.03 \\ 0.82 \end{pmatrix}$	$\begin{pmatrix} 0.13 \\ 0.88 \end{pmatrix}$	$\begin{pmatrix} -0.01 \\ 0.92 \end{pmatrix}$
μ_2	$\begin{pmatrix} -1.88 \\ 0.11 \end{pmatrix}$	$\begin{pmatrix} -1.97 \\ -0.13 \end{pmatrix}$	$\begin{pmatrix} -1.88 \\ -0.08 \end{pmatrix}$	$\begin{pmatrix} -1.96 \\ 0.02 \end{pmatrix}$
Σ_1	$\begin{pmatrix} 0.68 & 0.12 \\ 0.12 & 1.01 \end{pmatrix}$	$\begin{pmatrix} 0.88 & -0.13 \\ -0.13 & 1.11 \end{pmatrix}$	$\begin{pmatrix} 1.05 & 0.05 \\ 0.05 & 0.98 \end{pmatrix}$	$\begin{pmatrix} 0.97 & -0.07 \\ -0.07 & 1.08 \end{pmatrix}$
Σ_2	$\begin{pmatrix} 1.38 & 0.32 \\ 0.32 & 1.44 \end{pmatrix}$	$\begin{pmatrix} 0.76 & -0.04 \\ -0.04 & 0.76 \end{pmatrix}$	$\begin{pmatrix} 0.97 & -0.11 \\ -0.11 & 0.98 \end{pmatrix}$	$\begin{pmatrix} 0.99 & 0.02 \\ 0.02 & 0.94 \end{pmatrix}$

TABLE 1 – Estimations des paramètres de distribution conditionnelle de chacune des classes de chaque jeu de données Synth1

On voit qu'on obtient des distributions qui semblent tendre vers les lois de paramètres suivants :

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

3.2 Estimation du taux d'erreur des classifieurs

Nous voulons à présent tester l'efficacité du classifieur euclidien sur les données. Pour cela, nous allons effectuer de multiples séparations du jeu en un ensemble d'apprentissage et un ensemble de test (voir un ensemble de validation pour les K plus proches voisins). On apprendra le modèle (détermination des centres de gravité par exemple) sur l'ensemble d'apprentissage, mais on le testera à la fois sur lui-même et sur l'ensemble de test.

Estimation du taux d'erreur

Lorsque l'on détermine N séparations aléatoires des données en un ensemble d'apprentissage et un ensemble de test, il est possible de recueillir un échantillon de N estimations notées $E_1, ..., E_N$ du taux d'erreur (généralement effectuées sur l'ensemble de test) du classifieur. On peut alors déterminer une estimation ponctuelle $\hat{\varepsilon}$ de ε (moyenne) et un intervalle de confiance sur ε.

Intervalle de confiance

Le nombre d'erreur du classifieur sur un échantillon de données peut s'écrire :

$$X_i = \sum_{i=1}^{n} \mathbb{1}_{\omega(x_i) \neq \omega_i} \tag{3}$$

n étant le nombre d'individus dans l'ensemble testé, $\omega(x_i)$ la classe attribuée par le classifieur à l'individu i et ω_i la classe réelle de l'individu i.

Par conséquent, le taux ponctuel d'erreur d'un échantillon peut s'écrire :

$$E_i = \frac{1}{n} X_i \tag{4}$$

Il est donc intéressant de le voir comme la somme de n expériences aléatoires de Bernoulli avec pour issue une classification correcte ou non par le classifieur. Le nombre d'erreurs X_i d'un échantillon suit donc une loi binomiale :

$$X_i \sim B(n, \varepsilon) \tag{5}$$

avec ε le taux d'erreur obtenu par le classifieur sur une expérience. Idéalement ce taux d'erreur est un paramètre et donc ne change pas. En réalité il faut estimer cette valeur avec nos réalisations car on ne le connaît pas.

Lorsque l'on répète N fois l'expérience et que l'on recueille notre échantillon $E_1, ..., E_N$, on obtient N estimations du taux d'erreur.

Alors, la moyenne empirique $\overline{E} = \frac{E_1 + ... + E_N}{N}$ est un estimateur convergent et sans biais, que l'on peut noter $\hat{\varepsilon}$, de la moyenne des E_i et donc de ε. On a donc :

$$\overline{E} = \hat{\varepsilon} \approx \varepsilon \tag{6}$$

Cet estimateur tend en espérance (en moyenne) vers le paramètre ε (sans biais) :

$$E(\overline{E}) = E(\hat{\varepsilon}) = \varepsilon \tag{7}$$

De plus, d'après le théorème central limite (TLC), on peut approximer cette loi binomiale par une loi normale centrée réduite lorsque N tend l'infini, si les variables E_i sont indépendantes et identiquement distribuées :

$$\sqrt{N} \frac{\overline{E} - \mu}{\sigma} \xrightarrow{L} \mathcal{N}(0, 1) \tag{8}$$

ou encore

$$\overline{E} \xrightarrow{L} \mathcal{N}(\mu, \frac{\sigma}{\sqrt{N}}) \tag{9}$$

et que l'on note finalement dans notre cas (puisque l'espérance de $\overline{E}$ est ε) :

$$\overline{E} \xrightarrow{L} \mathcal{N}(\varepsilon, \varepsilon(1 - \varepsilon)/N) \tag{10}$$

En réalité, l'échantillonnage n'est pas indépendant puisque des mêmes données peuvent apparaître dans deux échantillons (les ensembles obtenus par redécoupage ne sont pas indépendants). L'on en fera cependant abstraction dans nos calculs.

Graphiquement, on voit bien que la loi binomiale donne une distribution gaussienne du taux d'erreur :

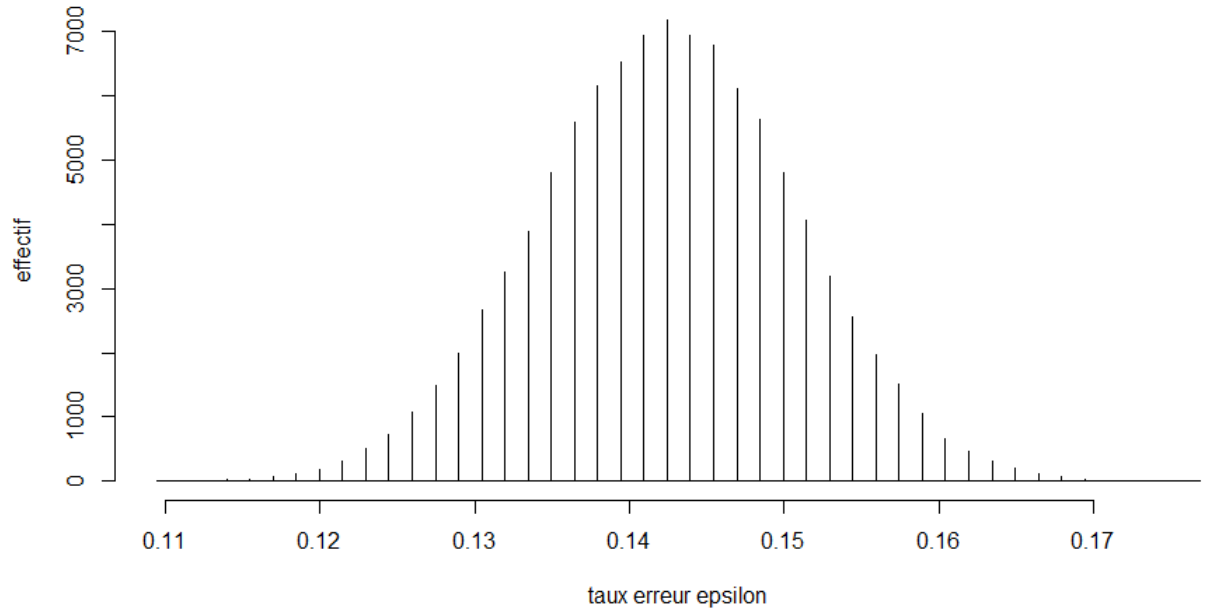

FIGURE 4 – Représentation en histogramme du taux d'erreur ε du classifieur euclidien sur les données Synth1-500 pour 100 000 estimations

On peut donc utiliser cette approximation pour trouver un intervalle de confiance [1] dans lequel est contenu l'espérance de $\overline{E}$ c'est-à-dire notre taux d'erreur ε :

$$I_c = \left[\overline{E} - t_{1-\alpha/2}\frac{s^*}{\sqrt{N}}, \overline{E} + t_{1-\alpha/2}\frac{s^*}{\sqrt{N}}\right] \tag{11}$$

avec N la taille de l'échantillon (ici 20) et s^* l'estimateur non biaisé de la variance [2] qui est obtenu avec la formule :

$$s^* = \frac{1}{N-1} * \sum_{i=1}^{N}(E_i - \overline{E}) \tag{12}$$

avec $t_{1-\alpha/2} = 2.093$

Classifieur Euclidien

Les résultats obtenus à partir des tests avec N = 20 et un niveau de confiance de 5% sont les suivants (les valeurs de variances ne sont pas demandées mais elles ont été ajoutées afin d'expliciter les inconnues du calcul) :

1. En réalité biaisé car l'estimation de la variance suppose l'indépendance entre les estimations
2. non biaisé dans le cas d'indépendance des expériences, tirage avec remise, mais ce n'est pas le cas ici (voir note 1)

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
Variance				
Apprentissage	0.0024	0.0006	0.000089	0.0000073
Test	0.0094	0.0018	0.00045	0.0000023
Estimation de ε				
Apprentissage	0.2	0.0896	0.13	0.1437
Test	0.2538	0.0894	0.1362	0.1407
Intervalle de confiance				
Apprentissage	[0.1989, 0.2011]	[0.0893, 0.0898]	[0.1299882, 0.1300718] [3]	[0.14367, 0.14374]
Test	[0.2494, 0.2582]	[0.0886, 0.0902]	[0.1360, 0.1364]	[0.1406, 0.1408]

TABLE 2 – Variance moyenne estimée, estimation du taux d'erreur et intervalle de confiance de ε, avec le classifieur euclidien

Nos estimations du taux d'erreur sont dans nos intervalles de confiance, ce qui est cohérent.

Interprétation

À partir d'une observation de variance, on peut conclure que le taux d'erreur de test est plus dispersé que les taux d'erreur d'apprentissage, et aussi que généralement la valeur d'erreur d'apprentissage est plus grande que la valeur d'erreur de test. Le modèle étant appris sur l'ensemble d'apprentissage, il commet moins d'erreurs sur lui-même que sur l'ensemble de test puisque les paramètres ont été déterminés avec lui. Cette différence n'est cependant pas toujours très significative et fluctue d'une estimation à une autre.

Ce que l'on observe cependant, c'est qu'entre les jeux de données de 40 individus et de 100 individus le taux d'erreur chute drastiquement de plus de moitié. Après cela, il remonte lorsque l'on augmente le nombre d'observations. On peut alors supposer que l'on est dans une situation de sur-apprentissage ou *overfitting*.

Le sur-apprentissage consiste en un mauvais dimensionnement des données ou de la structure utilisée pour classifier. En effet, un modèle d'apprentissage est efficace s'il peut généraliser un comportement pour prédire des données nouvelles. Si le modèle apprend tout *par coeur*, c'es-à-dire qu'il échantillone finalement toutes les possibilités et stocke trop d'informations, il ne pourra plus généraliser des caractéristiques et perdra sa capacité de prédiction.

Cependant, une réelle situation de sur-apprentissage est caractérisée par une constante baisse des erreurs sur l'ensemble d'apprentissage avec une bifurcation vers le haut uniquement pour l'ensemble à tester ou valider :

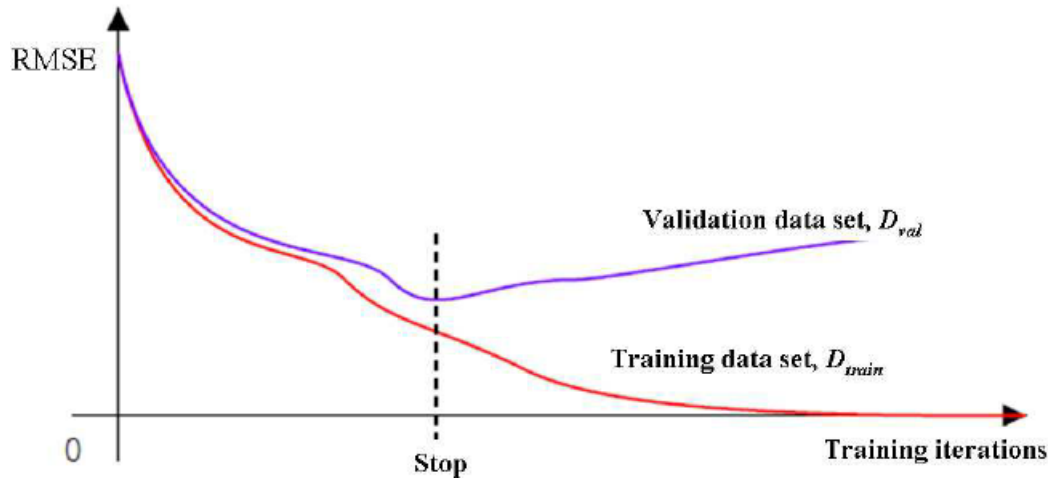

FIGURE 5 – Exemple de situation de sur-apprentissage. Stop marque le point à partir duquel la complexité du modèle est optimale. Il ne faut alors plus augmenter la taille de l'ensemble d'apprentissage.

3. Augmentation des chiffres significatifs pour plus de clarté

Ce n'est pas le cas ici puisqu'il augmente aussi.

L'hypothèse la plus probable est finalement que ce classifieur n'est pas adapté pour les jeux de données *extrêmes* de *Synth1*. En effet, plus l'on augmente le nombre de points, plus la représentation des points forme un nuage de plus en plus compact et mélangé de points des deux classes. Une séparation euclidienne (hyperplan médiateur) sera donc moins performante pour un ensemble de test.

À l'inverse, si l'on a pas assez d'individus, le modèle ne sera pas assez *entraîné* et n'aura pas de bonnes capacités de prédiction (40 individus retranchés d'un tiers lors de la séparation ne constitue pas un ensemble d'apprentissage suffisant, qui sera fortement perturbable lors de l'arrivée de nouveaux individus). Cela se traduit notamment par l'absence de nuage de points remarquables lors de la représentation des données.

Le bon compromis dans notre cas se situe autour des 100 observations, où l'on a un nuage bien séparable en deux groupes par une droite :

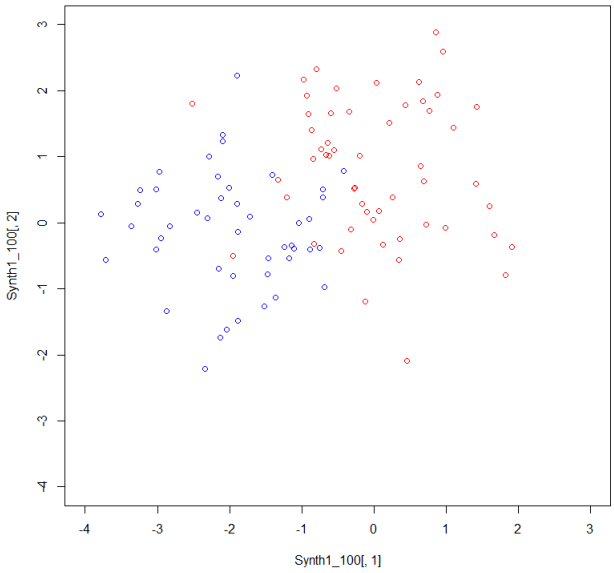

FIGURE 6 – Représentation des données Synth1-100

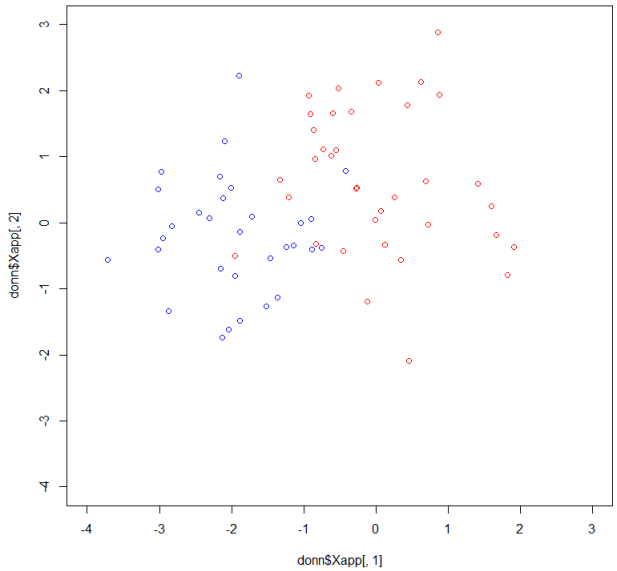

FIGURE 7 – Représentation des données Synth1-100 d'apprentissage (2/3 des données de gauche)

Même en coupant en un ensemble d'apprentissage et de test, il y a assez de données pour que l'apprentissage du modèle soit bon et qu'il soit performant.

K plus proches voisins

Notons tout d'abord que lors d'une séparation aléatoire de l'ensemble de données en un ensemble d'apprentissage, un ensemble de test et un ensemble de validation, si l'on utilise l'ensemble d'apprentissage comme ensemble de validation pour déterminer le nombre optimal de voisins à utiliser, alors on obtient $K_{opt} = 1$.

C'est assez logique puisque l'algorithme va essayer tous les K parmi ceux fournis en argument (*nppv*). Ainsi, lorsqu'il essaiera avec $K = 1$, il prendra la classe du plus proche voisin de chaque élément de l'ensemble de validation. Comme l'ensemble de validation utilisé ici est celui d'apprentissage, alors ce voisin sera lui-même (distance euclidienne nulle). Il prendra donc sa propre classe, aucune erreur ne sera donc faite par le classifieur.

On fait à nouveau des séparations (mais avec un ensemble de validation afin de déterminer l'hyper-paramètre K_{opt}) et on fait les calculs avec la méthode K plus proches voisins. Les différentes valeurs de K_{opt} obtenues avec $N = 20$ sont les suivantes :

	K optimal
Synth1-40	[7 3 7 7 3 7 5 5 7 5 5 3 3 1 3 1 9 11 1 7]
Synth1-100	[1 1 1 1 1 3 3 1 3 5 7 3 1 1 7 5 3 1 5 5]
Synth1-500	[9 5 5 5 11 7 5 7 5 9 7 7 3 3 5 9 9 7 11 9]
Synth1-1000	[9 11 9 9 7 7 9 11 3 7 7 7 9 5 9 5 9 5 5 7]

TABLE 3 – Valeurs de K_{opt} obtenues pour chaque séparation des données faite pour estimer le taux d'erreur

Les valeurs de K sont différentes parce que la division des données en ensembles d'apprentissage, de test, de validation, est complètement aléatoire à chaque itération.

Les résultats obtenus sont résumés à nouveau dans le tableau suivant :

	Synth1-40	Synth1-100	Synth1-500	Synth1-1000
Variance				
Apprentissage	0.0115	0.0028	0.00027	0.00027
Test	0.0194	0.0033	0.00052	0.00092
Estimation de ε				
Apprentissage	0.1625	0.051	0.1176	0.126
Test	0.255	0.1208	0.1536	0.158
Intervalle de confiance				
Apprentissage	[0.1571, 0.1679]	[0.0497, 0.0523]	[0.1175, 0.1177]	[0.1259, 0.1261]
Test	[0.2459, 0.2641]	[0.1193, 0.1224]	[0.1534, 0.1538]	[0.1576, 0.1584]

TABLE 4 – Variance moyenne estimée, estimation du taux d'erreur et intervalle de confiance de ε, avec la méthode des K plus proches voisins

De nouveau, nos estimations du taux d'erreur sont dans nos intervalles de confiance, ce qui est cohérent.

Interprétation

Cette fois, l'estimation du taux d'erreur d'apprentissage est significativement plus petite (surtout pour *Synth1-1000*) que l'estimation de test. Cette méthode semble donc plus adaptée à la distribution des données, ce que confirme une légère infériorité en moyenne de l'estimation du taux d'erreur d'apprentissage sur tous les jeux.

Du reste, on observe la même chose dans les fluctuations du taux en fonction du nombre de données, les remarques précédentes s'appliquent donc de manière analogue ici.

3.3 Jeu de données Synth2-1000

	Synth2-1000
π_1	0.523
π_2	0.477
μ_1	$\begin{pmatrix} 3.02 \\ -0.006 \end{pmatrix}$
μ_2	$\begin{pmatrix} -2.14 \\ -0.03 \end{pmatrix}$
Σ_1	$\begin{pmatrix} 0.99 & 0.11 \\ 0.11 & 1.09 \end{pmatrix}$
Σ_2	$\begin{pmatrix} 4.43 & -0.15 \\ -0.15 & 1.03 \end{pmatrix}$

TABLE 5 – Estimations des paramètres de distribution conditionnelle de chacune des classes du jeu de données *Synth2-1000* dont la distribution des classes ne suit pas la même loi que les données *Synth1*

On voit qu'on obtient des distributions qui semblent tendre vers les lois de paramètres suivants :

$$\mu_1 = \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \quad \Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix}$$

De la même manière que pour les données précédentes, nous calculons les taux d'erreurs moyens sur l'ensemble d'apprentissage et de test ce qui nous permet d'obtenir un estimateur ponctuel du taux d'erreur de chaque classifieur.

Les résultats obtenus avec le classifieur euclidien, $N = 20$ et $\alpha = 5\%$ sont les suivants :

	Variance	Estimation de ε	Intervalle de confiance
Apprentissage	0.00002	0.0646	[0.0646, 0.0646]
Test	0.00010	0.0574	[0.0573, 0.0574]

TABLE 6 – Variance moyenne estimée, estimation du taux d'erreur et intervalle de confiance de ε, avec le classifieur euclidien

Les résultats obtenus avec la méthode des *KPPV*, $N = 20$ et $\alpha = 5\%$ sont les suivants :

	Variance	Estimation de ε	Intervalle de confiance
Apprentissage	0.0002	0.0477	[0.0476, 0.0478]
Test	0.0003	0.0602	[0.0601, 0.0603]

TABLE 7 – Variance moyenne estimée, estimation du taux d'erreur et intervalle de confiance de ε, avec la méthode des *KPPV*

Interprétation

La méthode des KPPV semble légèrement produire de meilleurs résultats que le classifieur euclidien. Les résultats en général sont tout de mêmes plutôt bons, voici la représentation des données :

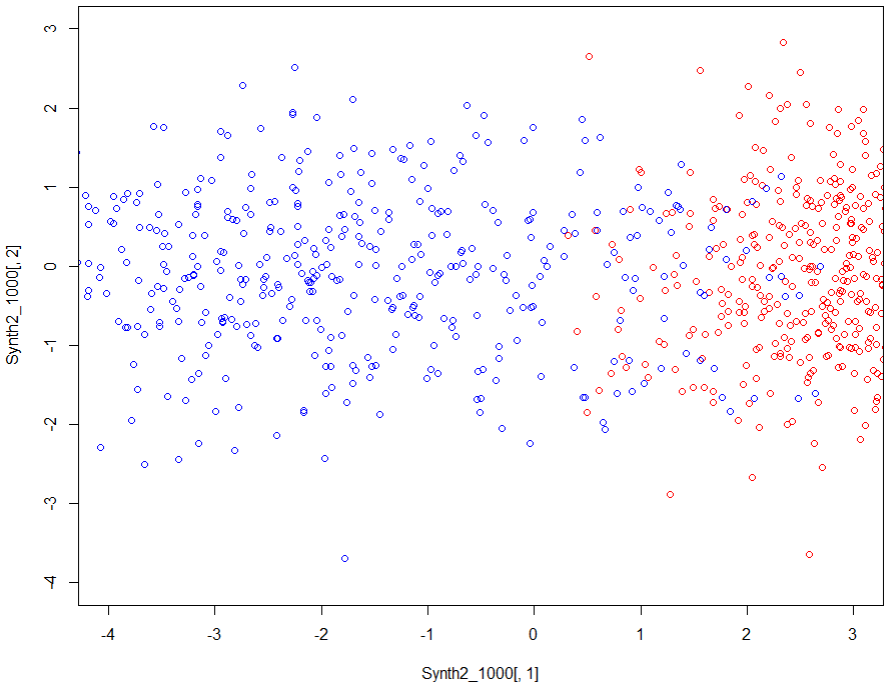

FIGURE 8 – Représentation des données *Synth2-1000* dans le plan

La représentation confirme les bonnes performances des classifieurs :
— La très bonne séparation verticale des deux groupes assure un bon apprentissage des paramètres euclidien et une bonne performance de ce classifieur lors de la prédiction de nouveaux individus
— La linéarité horizontale des deux groupes assure une propagation des voisins les plus proches de l'ensemble de test qui sera satisfaisante surtout aux extrémités droite et gauche du nuage.

3.4 Jeux de données réelles

Nous allons à présent utiliser des procédés tout à fait similaires, pour estimer le taux d'erreur des classifieurs avec des jeux de données réelles, c'est-à-dire dont la distribution des classes ne suit pas de loi de probabilité définie *manuellement*[4].

3.4.1 Pima

Taux d'erreur sur Pima :

Les résultats obtenus avec le classifieur euclidien, $N = 20$ et $\alpha = 5\%$ sont les suivants :

	Estimation de ε	Intervalle de confiance
Apprentissage	0.2411	[0.2408, 0.2415]
Test	0.2523	[0.2519, 0.2526]

TABLE 8 – Estimation du taux d'erreur et intervalle de confiance de ε, avec le classifieur euclidien

Les résultats obtenus avec la méthode des *KPPV*, $N = 20$ et $\alpha = 5\%$ sont les suivants :

	Estimation de ε	Intervalle de confiance
Apprentissage	0.1914	[0.1910, 0.1917]
Test	0.2564	[0.2561, 0.2567]

TABLE 9 – Estimation du taux d'erreur et intervalle de confiance de ε, avec la méthode des *KPPV*

Interprétation

Les résultats sont assez mauvais pour les deux classifieurs. La distribution des classes dans les données ne permet donc pas un bon apprentissage des paramètres de chacun des modèles.

3.4.2 Breastcancer

Taux d'erreur sur Breastcancer :

Les résultats obtenus avec le classifieur euclidien, $N = 20$ et $\alpha = 5\%$ sont les suivants : A FAIRE

	Estimation de ε	Intervalle de confiance
Apprentissage	0.03747	[0.03740, 0.03754]
Test	0.04320	[0.04313, 0.04327]

TABLE 10 – Variance moyenne estimée, estimation du taux d'erreur et intervalle de confiance de ε, avec le classifieur euclidien

Les résultats obtenus avec la méthode des *KPPV*, $N = 20$ et $\alpha = 5\%$ sont les suivants :

4. L'étiquette z de diabète est recueillie par statistique, non par distribution probabilistique, même s'il se peut que dans une population donnée cette distribution suive une loi...

	Estimation de ε	Intervalle de confiance
Apprentissage	0.0243	[0.0242, 0.0244]
Test	0.04205	[0.04196, 0.04216]

TABLE 11 – Variance moyenne estimée, estimation du taux d'erreur et intervalle de confiance de ε, avec la méthode des *KPPV*

Interprétation

Ici les résultats sont très bons. Le taux d'erreur estimé est très faible pour les deux classifieurs, entre 2% et 5%. la distribution des classes semble s'accorder aux deux classifieurs pour l'estimation de leurs paramètres. Le classifieur euclidien et la méthode des K plus proches voisins sont performants pour les prédictions sur ce *dataset*

4 Règle de Bayes

1.

Les distributions marginales des variables X^1 et X^2 suivant chacune une loi normale bivariée de paramètres μ_k et $\sum_k$ sont la projection dans chaque dimension de la loi globale, c'est-à-dire :

Pour le jeu de données *Synth1* :

— Classe ω_1 :
loi marginale selon x_1 : $x_1 \sim N(0,1)$
Loi marginale selon x_2 : $x_2 \sim N(1,1)$

— Classe ω_2 :
loi marginale selon x_1 : $x_1 \sim N(-2,1)$
Loi marginale selon x_2 : $x_2 \sim N(0,1)$

Jeu de données *Synth2* :

— Classe ω_1 :
loi marginale selon x_1 : $x_1 \sim N(3,5)$
Loi marginale selon x_2 : $x_2 \sim N(0,1)$

— Classe ω_2 :
loi marginale selon x_1 : $x_1 \sim N(-2,5)$
Loi marginale selon x_2 : $x_2 \sim N(0,1)$

Alors, les distributions marginales des variables X^1 et X^2 avec la répartition donnée par la fonction de densité de probabilité sont exprimées pour chaque loi par :

$$f_k(x_i) = \frac{1}{\sigma_k\sqrt{2\pi}} \times e^{-\frac{(x_i-\mu_k)^2}{2\sigma_k^2}} \tag{13}$$

2.

La densité de probabilité de la classe ω_k selon X est donnée par la formule :

$$f(x_1,x_2) = f_1(x_1) * f_2(x_2) = \frac{1}{2\pi\sigma_1\sigma_2} \times e^{-\frac{1}{2}[(\frac{x_1-\mu_1}{\sigma_1})^2+(\frac{x_2-\mu_2}{\sigma_2})^2} \tag{14}$$

L'équation des courbes d'isodensité est donnée par :
$f(x_1,x_2) = Cste$

$$\Rightarrow (\frac{x_1-\mu_1}{\sigma_1})^2 + (\frac{x_2-\mu_2}{\sigma_2})^2 = k^2 \tag{15}$$

Lorsque Σ est diagonale, ces courbes d'isodensité sont des ellipsoïdes de centre μ. On a donc des ellipses de centres (μ_1,μ_2) et de demi-axes $(k\sigma_1, k\sigma_2)$, que l'on appelle ellipses d'ordre k.

Elles correspondent, dans l'espace $\mathbb{R}^3$ à l'intersection du plan z avec la surface de densité $f(x_1, x_2)$. Ces courbes d'isodensité, aussi appelées courbes de niveau, représentent la dispersion des individus X dans le plan (x_1, x_2) et pour chaque ellipse ε_k d'ordre k on a :

$$P(X \in \varepsilon_k) = 1 - e^{-\frac{k^2}{2}}$$

3.

La règle de Bayes (stratégie de décision) s'exprime pour $g = 2$:

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

avec $f(x) = \pi_1 f_1(x) + \pi_2 f_2(x)$ la densité de mélange. Finalement on a la règle suivante :

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

soit de manière générale :

$$\delta^*(x) = \begin{cases} a_1 & \text{si } \frac{f_1(x)}{f_2(x)} > \frac{\pi_2}{\pi_1} \\ a_2 & \text{sinon} \end{cases} \tag{16}$$

Dans le cas d'une loi normale bivariée nous avons :

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

De plus dans le jeu de données *Synth1* nous avons des conditions d'homoscédasticité avec

$$\Sigma_1 = \Sigma_2 = I_2$$

Ainsi l'équation 16 de la règle de Bayes devient :

$$\frac{e^{-\frac{1}{2}(x-\mu_1)^T(x-\mu_1)}}{e^{-\frac{1}{2}(x-\mu_2)^T(x-\mu_2)}} > \frac{\pi_2}{\pi_1}$$

$$\Rightarrow e^{-\frac{1}{2}(x_1-\mu_{11})^2+(x_2-\mu_{12})^2-(x_1-\mu_{21})^2-(x_2-\mu_{22})^2} > \frac{\pi_2}{\pi_1} \tag{17}$$

En remplaçant les valeurs des μ_k et π_k par les données de l'énoncé pour *Synth1* nous obtenons :

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

$$\Rightarrow -\frac{1}{2}[-4x_1 - 2x_2 - 3] > 0$$

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

En représentant dans le plan (x_1, x_2) les données *Synth1_ 1000* on obtient :

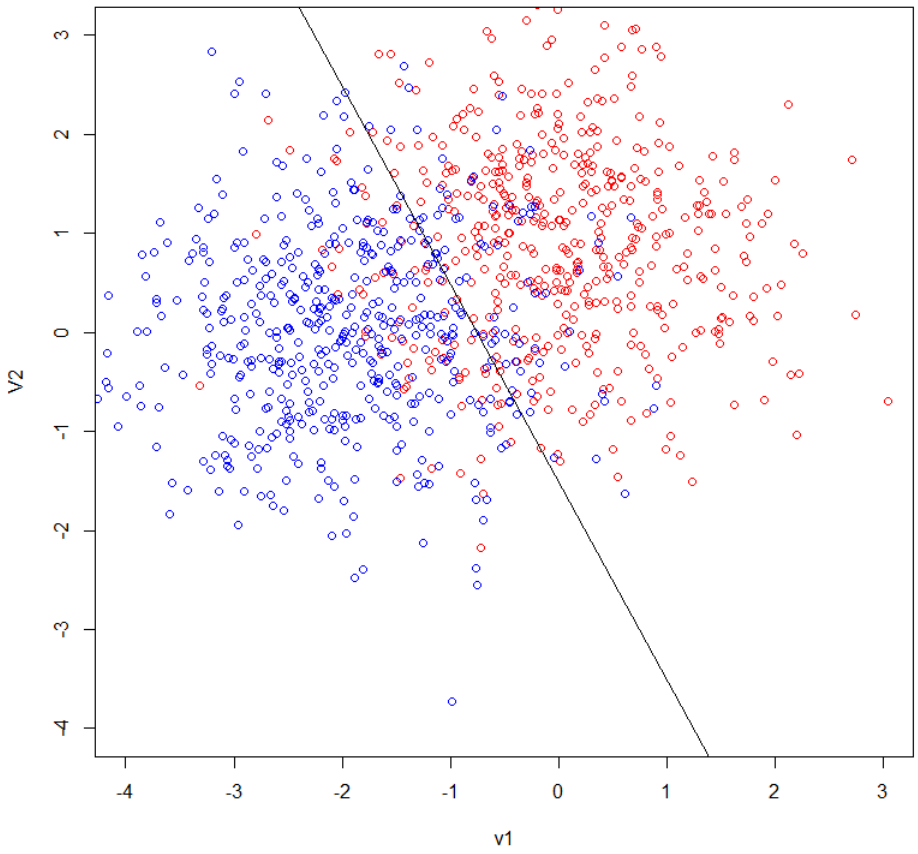

FIGURE 9 – Représentation des données *Synth1_1000* dans le plan avec la frontière définie par l'équation $x_2 = -\frac{3}{2} - 2x_1$

On en déduit que si $2x_1 + x_2 > k$, avec $k = -\frac{3}{2}$, alors x sera assigné à la classe ω_1, sinon à la classe ω_2.

4.

En partant de l'équation 16 on obtient :

$$\frac{\frac{1}{2\pi det\Sigma_1^{1/2}} \times e^{-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}}{\frac{1}{2\pi det\Sigma_2^{1/2}} \times e^{-\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-2}(x-\mu_2)}} > \frac{\pi_2}{\pi_1}$$

Après application de nos valeurs numériques, on obtient :

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

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

$$-4x_1^2 + 34x_1 > 5ln(\frac{1}{5}) + 41 \tag{18}$$

5.

Expression de l'erreur de Bayes

Dans le cas de *Synth1-n*, les variables X vérifient la condition d'homoscédasticité et g=2, par conséquent on a :

$$\delta^*(x) = \begin{cases} a_1 & si \;\; h(x) > ln(\frac{\pi_1}{\pi_2}) \\ a_2 & sinon \end{cases} \tag{19}$$

avec

$$h(x) = (x - \frac{\mu_1 + \mu_2}{2})^T \Sigma^{-1}(\mu_1 + \mu_2) \tag{20}$$

qui est la fonction discriminante. On sait aussi que

$$h(x) \sim \mathcal{N}(-\frac{\Delta^2}{2},\ \Delta)$$

selon ω_1 et

$$h(x) \sim \mathcal{N}(-\frac{\Delta^2}{2},\ \Delta)$$

selon ω_2 avec $\Delta^2 = (\mu_2 - \mu_1)^T \Sigma^{-1}(\mu_2 - \mu_1)$ ce qui nous mène à la condition $\pi_2 = \pi_1$ et donc à l'expression de l'erreur de Bayes :

$$\varepsilon^* = \Phi(-\frac{\Delta}{2}) \tag{21}$$

où Φ est la fonction de répartition de la loi normale univariée qui est centrée réduite.

Avec les paramètres du jeu de données *Synth1* on obtient $\varepsilon^* = 0.2136$. Dans le cas des données *Synth2*, il nous est impossible de calculer l'erreur avec le même moyen car les matrices de variance-covariance ne sont pas les mêmes selon les deux classes. Sans expression formelle, nous allons tout de même en donner une approximation avec la méthode du carré de la distance de Bhattacharyya entre les deux classes qui donne une borne supérieure de ε :

$$\varepsilon^* < \sqrt{\pi_2 \pi_1} \times e^{-\Delta_B^2} \tag{22}$$

avec

$$\Delta_B^2 = \frac{1}{16}(\mu_2 - \mu_1)^T (\frac{(\Sigma_1 + \Sigma_2)}{2})^{-1}(\mu 2 - \mu_1) + ln \frac{det \frac{(\Sigma_1 + \Sigma_2)}{2}}{\sqrt{det\Sigma_1\, det\Sigma_2}} \tag{23}$$

Une application numérique donne $\varepsilon^* = 0.1217$

. Nous pouvons comparer ces valeurs avec celles obtenues avec les deux classifieurs précédemment.

5 Conclusion

Tout d'abord, les résultats obtenus sur les intervalles de confiances qui contiennent notre taux d'erreur de chaque classifieur sont à considérer précautionneusement car l'estimation de la variance n'est pas bonne (indépendances des échantillons, voir 1).

Afin d'améliorer la qualité de nos estimations, deux méthodes peuvent être utilisées :
— Préférer les grands jeux de données sur lesquels on compte les erreurs : estimation ponctuelle du taux d'erreur du classifieur plus précise. Cependant on a vu que c'était parfois néfaste pour les performances des classifieurs, les taux seront donc plus hauts.
— Augmenter le nombre d'estimations faites afin de converger en espérance vers le taux d'erreur réel du classifieur (approximation de la loi Normale, voir graphique 4)

Ce travail pratique nous a permis de programmer en R de nouveaux outils d'analyse de données, et a constitué une première introduction à l'apprentissage automatique supervisé. Nous avons bien vu les différences avec la classification non supervisée qui ne s'attache qu'à classifier un ensemble d'individus déjà observés et non de prédire de nouvelles observations.

Table des matières

Table des figures