

TP3

Discrimination, théorie bayésienne de la décision

Sy09P019

Shuhan LIN

Grégory Mayemba

Exercice 01

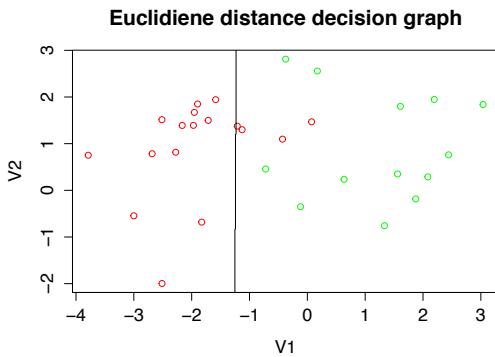
Classifieur euclidien, K plus proches voisins

EX 1.1

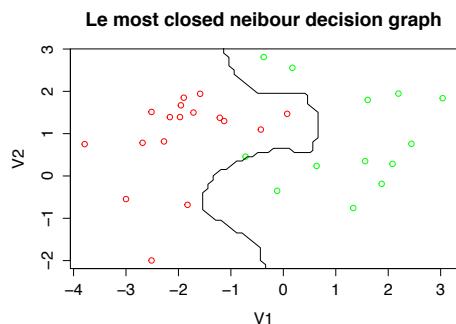
Dans cet exercice, nous écrivons les fonctions pour réaliser le classifieur euclidien et le classifieur K plus proches voisins (Kppv).

En prenant les données de test, nous illustrons le classifieur avec les fonctions de frontière.

D'abord, Avec le classifieur euclidien, nous obtenons le graphe ci-dessous :



Ensuite, avec le classifieur Kppv, nous obtenons la frontière ci-dessous :

**EX 1.2**

Jeux de données Synth1

Question 01 Estimer les paramètres des distributions conditionnelles

D'abord, nous avons un jeu de données $X_i, i = 1 \dots n$ (X est une matrice n par p , n est le nombre d'individus et p est le nombre de variable, $p= 2$ dans ce cas). Et puis, nous avons un vecteur $z_i, i = 1 \dots n$, qui indique la classe de chaque individu. Dans nos données, il y a deux groupes. Nous supposons que le nombre d'individus de chaque groupe est $n_k, k = 1,2$

Nous savons à priori que les jeux des données sont générés par un modèle gaussien mélangé (2 groupes et 2 dimensions). Nous supposons que les proportions sont $\pi_k, k = 1,2$, et puis pour chaque groupe, son espérance est μ_k et sa matrice de variance est Σ_k .

D'après l'estimation du maximum vraisemblance, nous avons les estimateurs suivants :

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} * \sum_{i=1}^n t_{ik} * X_i, t_{ik} = \begin{cases} 1, & \text{si } z_i = w_k \\ 0, & \text{sinon} \end{cases}$$

$$\hat{\Sigma}_k = \frac{1}{n_k} * \sum_{i=1}^n t_{ik} * (X_i - \mu_k)(X_i - \mu_k)^t$$

Avec cela, nous calculons des estimateurs pour chaque jeu de données.

Synth1-40

$$\hat{\pi}_1 = 0.55, \hat{\pi}_2 = 0.45$$

$$\hat{\mu}_1 = (-1.904681, 0.698841)$$

$$\hat{\mu}_2 = (0.880859, 0.858783)$$

$$\hat{\Sigma}_1 = [0.770133 \quad 0.256621 \\ 0.256621 \quad 1.079868]$$

$$\hat{\Sigma}_2 = [1.415465 \quad -0.010114 \\ -0.010114 \quad 1.221917]$$

Question A.**Synth1-100**

$$\pi_1^{\wedge} = 0.53, \pi_2^{\wedge} = 0.47$$

$$\mu_1^{\wedge} = (-1.808223, 1.057774)$$

$$\mu_2^{\wedge} = (0.983141, 1.161077)$$

$$\Sigma_1^{\wedge} = \begin{bmatrix} 1.508802 & 0.070977 \\ 0.070977 & 0.958333 \end{bmatrix}$$

$$\Sigma_2^{\wedge} = \begin{bmatrix} 0.846772 & 0.045459 \\ 0.045459 & 0.716319 \end{bmatrix}$$

Synth1-500

$$\pi_1^{\wedge} = 0.48, \pi_2^{\wedge} = 0.52$$

$$\mu_1^{\wedge} = (-1.910170, 0.935935)$$

$$\mu_2^{\wedge} = (0.997900, 0.984353)$$

$$\Sigma_1^{\wedge} = \begin{bmatrix} 1.160506 & 0.031492 \\ 0.031492 & 0.887273 \end{bmatrix}$$

$$\Sigma_2^{\wedge} = \begin{bmatrix} 0.919787 & -0.034844 \\ -0.034844 & 0.951468 \end{bmatrix}$$

Synth1-1000

$$\pi_1^{\wedge} = 0.488, \pi_2^{\wedge} = 0.512$$

$$\mu_1^{\wedge} = (-1.998992, 1.008186)$$

$$\mu_2^{\wedge} = (1.090890, 0.983732)$$

$$\Sigma_1^{\wedge} = \begin{bmatrix} 1.039694 & -0.007116 \\ 0.07116 & 0.971077 \end{bmatrix}$$

$$\Sigma_2^{\wedge} = \begin{bmatrix} 1.002469 & -0.024887 \\ -0.024887 & 1.022287 \end{bmatrix}$$

D'après les résultat des estimations, nous trouvons que

$$\pi_1 \approx 0.5, \pi_2 \approx 0.5$$

$$\mu_1 \approx (-2, 1), \mu_2 \approx (1, 1)$$

$$\Sigma_1 \approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \Sigma_2 \approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Comme $\Sigma_1 \approx \Sigma_2$, nous pouvons imaginer qu'un modèle d'analyse discriminante linéaire est adapté à ces jeux de données.

Selon l'énoncé, $E = \frac{1}{m} * \sum_{i=1}^m 1_{z_i \neq z_i^{\wedge}}$

Nous supposons que $S_i = \begin{cases} 1, & \text{si } z_i \neq z_i^{\wedge} \\ 0, & \text{sinon} \end{cases}$, qui suit évidemment la loi de Bernoulli. Si nous supposons que la probabilité d'erreur d'un point ici est p , nous savons que $S_i \sim \beta(\varepsilon)$, où ε est le taux d'erreur.

Ensuite, nous supposons que $X = m * E$, comme les variable S_i suivent une loi de Bernoulli, $X = \sum_{i=1}^m S_i$, donc X suit une loi binomiale, nous avons $X \sim \beta(m, \varepsilon)$

Comme $E = \frac{1}{m} * X$, E suit également une loi binomiale. Nous supposons que son espérance est μ , sa variance est σ^2 . D'après la loi binomiale, $\mu = m * \frac{\varepsilon}{m} = \varepsilon$, $\sigma^2 = \frac{m\varepsilon(1-\varepsilon)}{m^2} = \varepsilon(1-\varepsilon)/m$

Quand m est grand, selon le théorème de la limite centrale, E suit approximativement la loi normale. Nous avons $E \sim N(\varepsilon, \varepsilon(1-\varepsilon)/m)$

Par conséquent, E est l'estimateur du taux d'erreur, nous le nommons comme $\hat{\varepsilon}$. Pour calculer l'intervalle de confiance, nous pouvons remplacer la variance par $\frac{1}{m} * \hat{\varepsilon}(1-\hat{\varepsilon})$, et nous obtenons une intervalle de confiance $IC = [\hat{\varepsilon} - \mu_{1-\frac{\alpha}{2}} * \sqrt{\frac{1}{m} * \hat{\varepsilon}(1-\hat{\varepsilon})}, \hat{\varepsilon} + \mu_{1-\frac{\alpha}{2}} * \sqrt{\frac{1}{m} * \hat{\varepsilon}(1-\hat{\varepsilon})}]$

Maintenant, nous avons un ensemble d'échantillon de E , c'est $E_j, \forall j = 1 \dots N$. Nous avons une variable aléatoire $E^{moyen} = \sum_{j=1}^N E_j / N$

Nous pouvons obtenir une intervalle de confiance bilatérale de $IC = [\hat{\varepsilon} - \mu_{1-\frac{\alpha}{2}} * \sqrt{\frac{1}{mN} * \hat{\varepsilon}(1-\hat{\varepsilon})}, \hat{\varepsilon} + \mu_{1-\frac{\alpha}{2}} * \sqrt{\frac{1}{mN} * \hat{\varepsilon}(1-\hat{\varepsilon})}]$

Question B

Pour chaque jeu de données, nous avons calculé l'estimation du taux d'erreur et l'intervalle de confiance sur l'ensemble d'apprentissage et l'ensemble de test.

Pour le niveau de confiance, nous prenons

$$1 - \alpha = 0.95$$

Synth1-40

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.1148148$$

$$IC_{\hat{\varepsilon}} = [0.08792627, 0.14170336]$$

L'ensemble de test

$$\hat{\varepsilon} = 0.1384615$$

$$IC_{\hat{\varepsilon}} = [0.09647952, 0.18044356]$$

Synth1-100

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.1560606$$

$$IC_{\hat{\varepsilon}} = [0.1364828, 0.1756384]$$

L'ensemble de test

$$\hat{\varepsilon} = 0.1647059$$

$$IC_{\hat{\varepsilon}} = [0.1368275, 0.1925843]$$

Synth1-500

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.1238739$$

$$IC_{\hat{\varepsilon}} = [0.1159619, 0.1317858]$$

L'ensemble de test

$$\hat{\varepsilon} = 0.1122754$$

$$IC_{\hat{\varepsilon}} = [0.1015687, 0.1229822]$$

Synth1-1000

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.1070571$$

$$IC_{\hat{\varepsilon}} = [0.1018064, 0.1123077]$$

L'ensemble de test

$$\hat{\varepsilon} = 0.1041916$$

$$IC_{\hat{\varepsilon}} = [0.09686533, 0.11151790]$$

Plus le jeu de données est grand, plus l'estimation du taux d'erreur est faible.

Quand le nombre de données est 1000, nous obtenons un taux d'erreur de 10% environ pour l'ensemble d'apprentissage et 11% pour ceux de test.

Généralement, l'estimation du taux d'erreur d'apprentissage est la même que celle du taux d'erreur de test pour le classifieur de distances euclidienne.

Question 03 Déterminer le nombre optimal du classifieur le plus proche voisin.

Nous prenons le jeu de données de 40 comme exemple, et nous prenons le nombre des voisins k de 1 jusqu'à 39.

Nous faisons le test 10 fois, et les 10 fois, le meilleur k vaut 1.

Le résultat est évident, car nous prenons l'ensemble d'apprentissage comme ensemble de validation. Si on choisit 1 voisin, le plus proche est forcément le points lui-même. Du coup, la classification est toujours correct et le taux d'erreur vaut 0. Par conséquent, le k le plus optimal vaut 1.

Question 04 Estimer le taux d'erreur avec le classifieur Kppv

Comme l'exercice précédent, pour chaque jeu de données, nous avons calculé l'estimation du taux d'erreur et l'intervalle de confiance sur l'ensemble d'apprentissage et l'ensemble de test.

Pour le niveau de confiance, nous prenons $1 - \alpha = 0.95$

Synth1-40

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.0725$$

$$IC_{\hat{\varepsilon}} = [0.0470877, 0.0979123]$$

L'ensemble de test

$$\hat{\varepsilon} = 0.13$$

$$IC_{\hat{\varepsilon}} = [0.08339158, 0.17660842]$$

Synth1-100

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.08$$

$$IC_{\hat{\varepsilon}} = [0.06318538, 0.09681462]$$

L'ensemble de test

$$\hat{\varepsilon} = 0.13$$

$$IC_{\hat{\varepsilon}} = [0.1005222, 0.1594778]$$

Synth1-500

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.0682$$

$$IC_{\hat{\varepsilon}} = [0.06121258, 0.07518742]$$

L'ensemble de test

$$\hat{\varepsilon} = 0.0908$$

$$IC_{\hat{\varepsilon}} = [0.07953709, 0.10206291]$$

Synth1-1000

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.0571$$

$$IC_{\hat{\varepsilon}} = [0.05255223, 0.06164777]$$

L'ensemble de test

$$\hat{\varepsilon} = 0.072$$

$$IC_{\hat{\varepsilon}} = [0.06483521, 0.07916479]$$

Nous observons également que plus le jeu de données est grand, plus l'estimation du taux d'erreur pour l'ensemble d'apprentissage et celle de l'ensemble de test est faible.

En comparant avec celles obtenues par le classifieur de distances euclidiennes, nous constatons que :

$$\hat{\varepsilon}_{app_ppv} < \hat{\varepsilon}_{app_ceuc}$$

$$\hat{\varepsilon}_{tst_ppv} < \hat{\varepsilon}_{tst_ceuc}$$

Comme le classifieur Kppv est beaucoup plus complexe que le classifieur des distances euclidiennes, il obtient un résultat plus fiable.

Ensuite, nous observons également que

$$|\hat{\varepsilon}_{app_ppv} - \hat{\varepsilon}_{tst_ppv}| > |\hat{\varepsilon}_{app_ceuc} - \hat{\varepsilon}_{tst_ceuc}|$$

Pour le classifieur de distances euclidiennes, les résultat des deux ensemble sont presque les même. Cependant, pour le classifieur Kppv, le résultat de l'ensemble d'apprentissage est plus fiable que celui de l'ensemble de test.

La raison est que, pour le classifieur de distances euclidiennes, nous calculons les distances des points avec les centres de gravité. Du coup, cela ne diffère pas trop pour l'ensemble d'apprentissage et l'ensemble de teste.

Par contre, dans le classifieur le plus proche voisin, nous recherchons les distances pour déterminer les k plus proches voisin. Du coup, pour l'ensemble d'apprentissage, il compare avec les points de lui-même, cela fait moins d'erreur. C'est pourquoi il est plus fiable que l'ensemble de teste.

Jeux de données Synth2

Ensuite, nous étudions le jeu de données Synth2.

Question 01 Estimer les paramètres des distributions conditionnelles

Comme l'exercice précédent, nous utilisons l'estimation du maximum vraisemblance à estimer des paramètres. Nous avons des résultats comme ci-dessous :

$$\pi_1^* = 0.488, \pi_2^* = 0.512$$

$$\mu_1^* = (-4.055942, 1.011496)$$

$$\mu_2^* = (4.028957, 1.067821)$$

$$\Sigma_1^* = \begin{bmatrix} 1.014560 & 0.018451 \\ 0.018451 & 0.939754 \end{bmatrix},$$

$$\Sigma_2^* = \begin{bmatrix} 4.953398 & 0.107909 \\ 0.107909 & 5.022125 \end{bmatrix}$$

Nous pouvons estimer que :

$$\pi_1 \approx 0.5, \pi_2 \approx 0.5$$

$$\mu_1 \approx (-4, 1), \mu_2 \approx (4, 1)$$

$$\Sigma_1 \approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \Sigma_2 \approx \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

Question 02 Estimer les résultats avec les deux classifieurs*Classifieur de distance euclidienne*

Nous avons obtenu le résultat comme ci-dessous :

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.007057057$$

$$IC_{\hat{\varepsilon}} = [0.005635482, 0.008478633]$$

L'ensemble de teste

$$\hat{\varepsilon} = 0.005838323$$

$$IC_{\hat{\varepsilon}} = [0.004011349, 0.007665298]$$

Classifieur de plus proche voisin

L'ensemble d'apprentissage

$$\hat{\varepsilon} = 0.004$$

$$IC_{\hat{\varepsilon}} = [0.002762892, 0.005237108]$$

L'ensemble de teste

$$\hat{\varepsilon} = 0.0062$$

$$IC_{\hat{\varepsilon}} = [0.004024253, 0.008375747]$$

Avec ce jeu de données, nous obtenons les estimations des taux d'erreur beaucoup plus fiable.

EX2 Règle de Bayes

Toutes les covariances sont nulles donc les variables X_1 et X_2 sont indépendantes:

$$f_{X^1}(x) = f_{X^1}(x/\omega_1)\pi_1 + f_{X^1}(x/\omega_2)\pi_2$$

Question 01

Jeu de données synth1, classe 1 :

$$\mu_1 = \begin{pmatrix} -2 \\ 1 \end{pmatrix} \quad \Sigma 1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$f_{X^1}(x/\omega_1) = \frac{1}{\sqrt{2\pi}} e^{[-(x_1+2)^2/2]}$$

$$f_{X^2}(x/\omega_1) = \frac{1}{\sqrt{2\pi}} e^{[-(x_2-1)^2/2]}$$

Jeu de données synth1, classe 2:

$$\mu_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \Sigma 2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$f_{X^1}(x/\omega_2) = \frac{1}{\sqrt{2\pi}} e^{[-(x_1-1)^2/2]}$$

$$f_{X^2}(x/\omega_2) = \frac{1}{\sqrt{2\pi}} e^{[-(x_2-1)^2/2]}$$

Jeu de données synth2, classe 1:

$$\mu_1 = \begin{pmatrix} -4 \\ 1 \end{pmatrix} \quad \Sigma 1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$f_{X^1}(x/\omega_1) = \frac{1}{\sqrt{2\pi}} e^{[-(x_1+4)^2/2]}$$

$$f_{X^2}(x/\omega_1) = \frac{1}{\sqrt{2\pi}} e^{[-(x_2-1)^2/2]}$$

Jeu de données synth2, classe 2 :

$$\mu_2 = \begin{pmatrix} 4 \\ 1 \end{pmatrix} \quad \Sigma 2 = \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}$$

$$f_{X^1}(x/\omega_1) = \frac{1}{\sqrt{10\pi}} e^{[-(x_1-4)^2/10]}$$

$$f_{X^2}(x/\omega_1) = \frac{1}{\sqrt{10\pi}} e^{[-(x_2-1)^2/10]}$$

Question 02

Pour le jeu de données 1

D'abord, nous calculons la fonction de densité jointe des classe 1 et classe2 :

$$f_X(x/\omega_1) = \frac{1}{2\pi} e^{[-(x_1+2)^2+(x_2-1)^2/2]}$$

$$f_X(x/\omega_2) = \frac{1}{2\pi} e^{[-(x_1-1)^2+(x_2-1)^2/2]}$$

La courbe d'iso-densité est la courbe où des points ont la même densité : $f(x) = K$

Du coup, pour les classes 1 et 2, on a
 $(x_1 + 2)^2 + (x_2 - 1)^2 = Cste$ et $(x_1 - 1)^2 + (x_2 - 1)^2 = Cste$

Par résultat, les courbes sont bien des cercles.

Pour la classe 1 :

Cercle centre : (-2, 1)

Rayon: $\sqrt{2\ln(2\pi K)^{-1}}$ $f(x) = K$

Pour la classe 2 :

Cercle centre : (1, 1)

Rayon: $\sqrt{2\ln(2\pi K)^{-1}}$ $f(x) = K$

De même pour le jeu de données Synth2

$$f_X(x/\omega_1) = \frac{1}{2\pi} e^{[-(x_1+4)^2+(x_2-1)^2/2]}$$

$$f_X(x/\omega_2) = \frac{1}{10\pi} e^{[-(x_1-4)^2+(x_2-1)^2/10]}$$

Pour classe 1 :

Cercle centre : (-4, 1)

Rayon: $\sqrt{2\ln(2\pi K)^{-1}}$ $f(x) = K$

Pour classe 2:

Cercle centre : (4, 1)

Rayon: $\sqrt{10\ln(10\pi K)^{-1}}$ $f(x) = K$

Question 03

D'abord, nous obtenons la règle de Bayes :

$$\delta(x) = \begin{cases} \omega_1 \text{ si } f_X(x/\omega_1)\pi_1 > f_X(x/\omega_2)\pi_2 \\ \omega_2 \text{ sinon} \end{cases}$$

Pour les premiers jeux de données :

$$\delta(x) = \begin{cases} \omega_1 \text{ si } x_1 < -1/2 \\ \omega_2 \text{ sinon} \end{cases}$$

C'est un classifieur linéaire

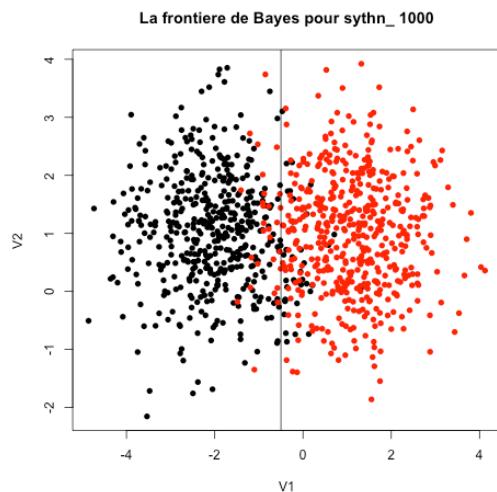
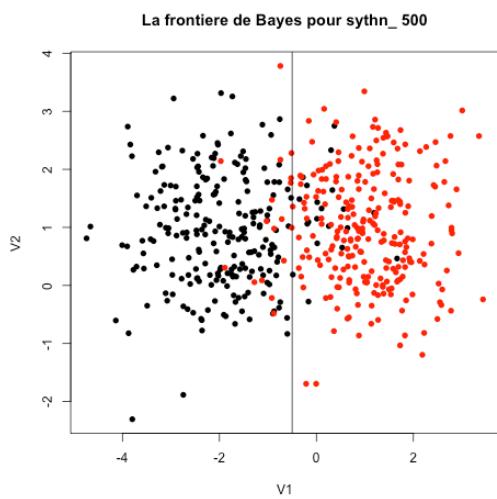
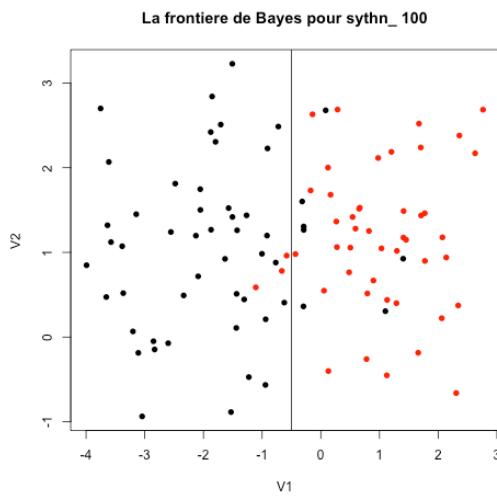
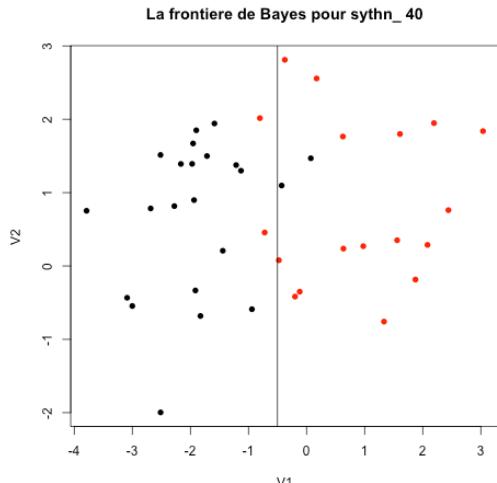
Pour le deuxième jeu de données (synth2):

$$\delta(x) = \begin{cases} \omega_1 \text{ si } (x_1 + 4)^2 + (x_2 - 1)^2 < 5/2 \ln 5 \\ \omega_2 \text{ sinon} \end{cases}$$

La frontière de décision du classifieur est un cercle.

Question 04

Ci-dessous, les frontières de décision dans le plan formé par les variables X1 et X2 pour les 4 premiers jeux de données :



Question 05

Taux d'erreur	Classifieur de Bayes	Méthode des plus proches voisins
synth-40	0.1010101	0.13
synth-100	0.0401445	0.13
synth-500	0.008012821	0.0908
synth-1000	0.004002305	0.072

On constate que les erreurs de Bayes pour le classifieur de Bayes pour les 4 premiers jeux de données sont inférieures aux erreurs de Bayes obtenues avec la méthode des K plus proches voisins

On remarque que plus le nombre de données est grand, plus le classifieur de Bayes est efficace par rapport au classifieur kppv

Conclusion

Dans ce TP3, nous avons approfondit l'étude de l'apprentissage supervisé.

D'abord, nous avons appris comment réaliser deux types simples de classificateurs : le classificateur euclidien et le classificateur K plus proches voisins. Le premier est plus simple tandis que le second est plus fiable.

Ensuite, nous avons appris à déterminer la règle de Bayes sur un jeu de données du modèle gaussien mélangé. Grâce à cela, nous savons comment créer un classificateur de Bayes et calculer son taux d'erreur.

L'Annexe

Code du classifieur euclidien

```

ceuc.app <- function(Xapp,zapp)
{
  n <- dim(Xapp)[1]
  p <- dim(Xapp)[2]
  g <- unique(zapp)
  nb_g <- length(unique(zapp)) # number of
groups

#Firstly, calculate the gravity center for each
class
  dist_center <- matrix(0,nrow = nb_g,ncol = p)
  dist <- matrix(0,nrow = 0, ncol=p)
  for(k in 1:nb_g)
  {
    for(i in 1:n)
    {
      if(zapp[i] == g[k])
      {
        dist <- rbind(dist,Xapp[i,])
        #print(dist)
      }
    }
    dist_center[k,] <- colMeans(dist)
  }
  return (dist_center)
}

ceuc.val <- function(mu,Xtst)
{
  n <- dim(Xtst)[1]
  p <- dim(Xtst)[2]
  nb_g <- dim(mu)[1]

#Step 1 Calculate the distance between chaque
points and every groups' gravity center
#Step 2 Choose the minimal distance to
determine which class the point belongs to
  dist <- matrix(0,nrow = n,ncol = nb_g)
  class <- rep(0, length=n)
  for(i in 1:n)
  {
    for(k in 1:nb_g)
    {
      dist[i,k] <- dist(rbind(Xtst[i,],mu[k,]))
    }
    class[i] <- which(dist[i,] == min(dist[i,]))
  }

#print(class)
  return (class)
}

```

Code du classifieur K plus proches voisins

```

kppv.app<-function(Xapp,zapp,Xval,zval,nppv)
{
  napp <- dim(Xapp)[1]
  p <- dim(Xapp)[2]
  nval <- dim(Xval)[1]
  g <- unique(zapp)
  nb_g <- length(unique(zapp))

  dist <- rep(0,napp)
  Error_proba <- rep(1,max(nppv))

  for(k in nppv)
  {
    class <- rep(0,nval)
    nerror <- 0

    for(j in 1:nval)
    {
      for(i in 1:napp)
      {
        dist[i] <- dist(rbind(Xapp[i,],Xval[j,]))
      }
    }
    Xapp_dist <- cbind(Xapp,dist)

    index <- indexOf_K_Minimum(Xapp_dist,k)

    class[j] <- class_decision(zapp,index,k)

  }

  Error_proba[k] <- Error_proba(class,zval,nval)
}

return (Find_min_k(Error_proba,nppv))
}

kppv.val <- function(Xapp,zapp,K,Xtst)
{
  n <- dim(Xapp)[1]
  app_len <- dim(Xapp)[1]
  tst_len <- dim(Xtst)[1]

  class <- rep(0,tst_len)
  for(i in 1:tst_len)
  {
    dist <- rep(0,app_len)
    for(j in 1:app_len)
    {
      dist[j] <- dist(rbind(Xapp[j,],Xtst[i,]))
    }
  }

  Xapp_dist <- cbind(Xapp,dist)

  index <- indexOf_K_Minimum(Xapp_dist,K)
  class[i] <- class_decision(zapp,index,K)
}

return (class)
}

```