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.

Prenant les données du teste, nous illustrons le classifieur avec les fonctions de frontière.

D’abord, Avec le classifieur euclidien, nous obtenons le graphe comme cela :



Ensuite, avec le classifieur k plus proche voisin, nous obtenons la frontière au 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 est une matrice n par p, n est nombre d’individus et p est le nombre de variable, p= 2 dans ce cas). Et puis, nous avons un vecteur , 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

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 , et puis pour chaque groupe, son espérance est et sa matrice de variance est .

D’après l’estimation du maximum vraisemblance, nous avons des estimateurs suivants :

D’ après cela, nous calculons des estimateurs pour chaque jeu de données.

**Synth1-40**

**Synth1-100**

**Synth1-500**

**Synth1-1000**

D’après les résultat des estimations, nous trouvons que

Comme , nous pouvons imaginer que un modèle de l’analyse discriminante linéaire suit bien à ces jeux de données.

**Question 02 Estimer le taux d’erreur avec le classifieur distance euclidienne**

**Question A.**

Selon l’énoncé,

Nous supposons que , qui suit évidemment la loi de Bernoulli. Si nous supposons que la probabilité d’erreur d’un point ici est p, nous savons que , où est le taux d’erreur.

Ensuite, nous supposons que , comme les variable suivons la loi de Bernoulli, , donc X suit la loi binomiale, nous avons

Comme E suit également la loi binomiale. Nous supposons que son espérance est , sa variance est . D’après la loi binomiale, ,

Quand m est grand, selon le théorème de la limite centrale, E suit approximativement la loi normale. Nous avons

Par conséquent, E est l’estimateur du taux d ‘erreur, nous le nommons comme . Pour calculer l’intervalle de confiance, nous remplacer la variance par , et nous obtenons une intervalle de confiance

Maintenant, nous avons un ensemble d’échantillon de E, c’est . Nous avons une variable aléatoire

Nous pouvons obtenir une intervalle de confiance bilatérale de

**Question B**

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

Pour le niveau de confiance, nous prenons que

**Synth1-40**

L’ensemble d’apprentissage

L’ensemble de teste

**Synth1-100**

L’ensemble d’apprentissage

L’ensemble de teste

0.1647059

**Synth1-500**

L’ensemble d’apprentissage

L’ensemble de teste

0.1122754

**Synth1-1000**

L’ensemble d’apprentissage

L’ensemble de teste

D’abord, quand le jeu de données est plus grand, l’estimation du taux d’erreur fait plus faible.

Quand le nombre de données est 1000, nous obtenons une taux d’erreur de 10% environ pour l’ensemble d’apprentissage et 11% pour

ceux de teste.

Généralement, l’estimation du taux d’erreur d’apprentissage sont la même comme la taux d’erreur de teste pour le classifieur distance euclidienne.

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

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

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

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

**Question 04** **Estimer le taux d’erreur avec le classifieur k le plus proche voisin**

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

Pour le niveau de confiance, nous prenons que

**Synth1-40**

L’ensemble d’apprentissage

0.0725

L’ensemble de teste

0.13

**Synth1-100**

L’ensemble d’apprentissage

0.08

L’ensemble de teste

**Synth1-500**

L’ensemble d’apprentissage

0.0682

L’ensemble de teste

0.0908

**Synth1-1000**

L’ensemble d’apprentissage

0.0571

L’ensemble de teste

D’abord, nous observons également que quand le jeu de données est plus grand, l’estimation du taux d’erreur pour l’ensemble d’apprentissage et celle de l’ensemble de teste sont devenue plus faible.

En comparant avec celles obtenue par le classifieur de distance euclidienne, nous constatons que :

Comme le classifieur le plus proche voisin sont beaucoup compliqué que le classifieur distance euclidienne, il obtenu un résultat plus fiable.

Ensuite, nous observons également que

Pour le classifieur de distance euclidien, les résultat des deux ensemble sont presque la même. Cependant, pour le classifieur le plus proche voisin, le résultat de l’ensemble d’apprentissage sont plus fiable que celui de l’ensemble de teste.

La raison est que, pour le classifieur distance euclidien, nous calculons les distances des points avec les centres de gravité. Du coup, cela ne différencie 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 :

Nous pouvons estimer que :

**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

L’ensemble de teste

*Classifieur de plus proche voisin*

L’ensemble d’apprentissage

L’ensemble de teste

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 et sont indépendantes:

**Question 01**

Jeu de données synth1, classe 1 :

Jeu de données synth1, classe 2:

Jeu de données synth2, classe 1:

Jeu de données synth2, classe 2 :

**Question 02**

Pour jeu de données 1

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

La courbe d’iso-densité est la courbe où des points ont la même densité :

Du coup, pour classe 1 et classe2, cela fait

et

Par résultat, les courbes sont bien des cercles.

Pour classe 1 :

Cercle centre : (-2 ,1)

Rayon:

Pour classe 2 :

Cercle centre : (1 ,1)

Rayon:

Même pour le jeu de données Synth2

Pour classe 1 :

Cercle centre : (-4 ,1)

Rayon:

Pour classe 2:

Cercle centre : (4 ,1)

Rayon:

**Question 03**

D’abord, nous obtenons la règle de bayés :

Pour le première données :

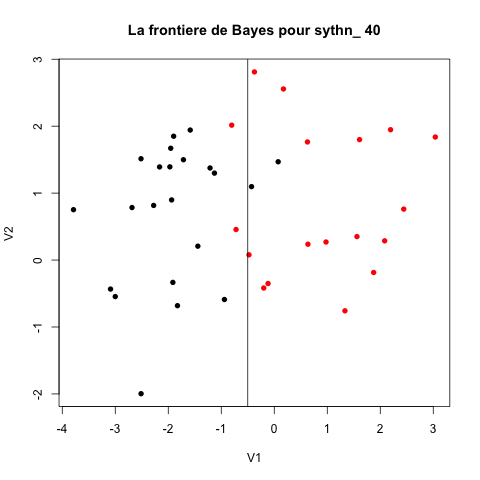
C’est un classifieur linéaire

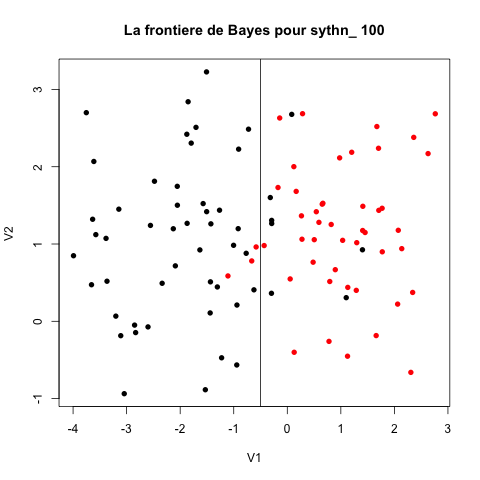
Pour le deuxième données :

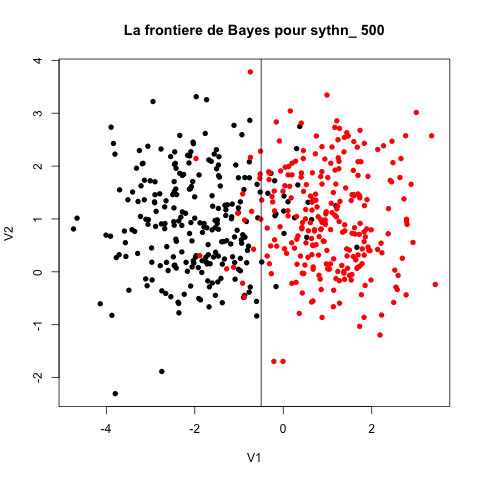
C’est une cercle comme la forme du classifieur.

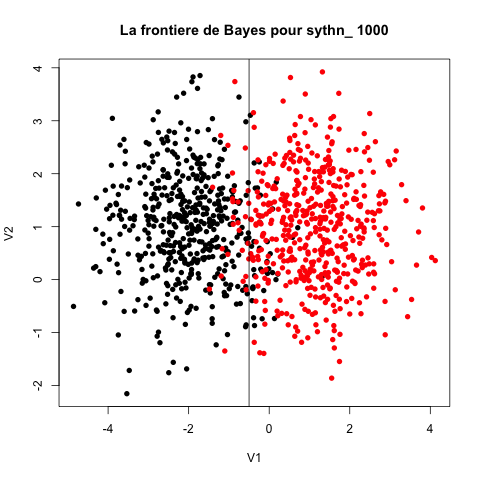
**Question 04**

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**

|  |  |  |
| --- | --- | --- |
| Jeu de données/erreur de bayes avec l'une des méthodes | 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 erreur 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 étudié approfondissement dans l’apprentissage supervisé.

D’abord, nous avons appris comment réaliser deux types simples des classifieurs : le classifieur euclidien et le classifieur K plus proches voisins. Le premier est plus simple tandis que le deuxième est plus fiable.

En suite, nous avons appris également le règle de Bayes sur un jeu de donnée du modèle gaussien mélangé. Grâce à cela, nous savons comment faire le calcul pour un classifieur bayes et son taux d’erreur.

**L’Annexe**

**Code du classfieur 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, caluclate 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 classfieur 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)

nberror <- 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)

}