Analyse de données
M1 Statistique et économétrie
V. Monbet
Analyse Discriminante

La collection des iris de Fisher est probablement l'une des plus célèbres dans le domaine de la reconnaissance de formes. Bien qu'ancienne, elle continue de faire référence dans le domaine. Le corpus consigne des mesures biométriques relevées sur des échantillons végétaux de type iris

1. longueur du sépale (en centimètres)
2. largeur du sépale (en centimètres)
3. longueur du pétale (en centimètres)
4. largeur du pétale (en centimètres)

Les 150 individus de la base sont répartis en trois classes équilibrées (50 individus dans chaque catégorie) correspondant à trois espèces florales : *iris setosa*,*iris versicolor* et *iris virginica*. La classe *setosa* est linéairement séparable des deux autres, alors que *versicolor* et *virginica* ne le sont pas entre elles.

Nous proposons dans ce TP de comparer plusieurs méthodes d'analyse discriminante ; plus précisément, nous allons considérer différents estimateurs des lois des variables explicatives pour construire des règles de classement.
Remarque - Vous pouvez commencer par traiter toutes les questions portant sur l'analyse univariée.

1 Analyse descriptive

1. Sous `R`, on peut charger les données par l'instruction suivante `data(iris)`. Faites une analyse descriptive rapide des données. Discuter notamment le graphique suivant :

   ```
   plot(iris[1:4], pch=20,
         col=c("red", "green", "blue")[as.numeric(iris$Species)])
   ```

2. L'analyse de la variance (ANOVA) permet de tester l'effet de la variable discrète (Species) sur les variables continues. La variable continue qui correspond à la plus grande statistique de test de Ficher observée est la variable qui conduira à la meilleure règle de classement si on choisit de travailler en dimension 1. Aidez-vous des instructions suivantes pour déterminer les deux variables les plus discriminantes parmi les 4 variables continues.

```
var.names = c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width")
nb_var = length(var.names)
for (k in 1:nb_var){
print(var.names[k])
print(anova(aov(iris[,k]~Species, data = iris)))
}
```

3. Densités marginales univariées
   (a) Par la suite, nous allons chercher à estimer les lois (ou la loi jointe) de ces deux variables afin de construire une règle de classement. Il est donc intéressant de visualiser leurs histogrammes. Pour la première variable, on fait par exemple

```
ISE = which(iris$Species == "setosa")
IVI = which(iris$Species == "virginica")
IVE = which(iris$Species == "versicolor")
br = seq(min(iris[,3]),max(iris[,3]),length.out=30)
hist(iris[ISE,3],breaks=br,col="red",main=NULL,freq=FALSE,
    xlab="Longueur de pétales (cm)",ylab="Fréquences")
hist(iris[IVE,3],breaks=br,col="blue",main="",add=TRUE,freq=FALSE)
hist(iris[ISE,3],breaks=br,col="red",add=TRUE,main=NULL,freq=FALSE)
hist(iris[IVI,3],breaks=br,col="green",add=TRUE,main=NULL,freq=FALSE)
title("Histogrammes de la longueur de pétales")
```

   (b) Adapter ces commandes pour réaliser le même graphique pour la seconde variable la plus discriminante.
   (c) Superposer aux histogrammes les densités des lois de Gauss s'ajustant au mieux aux lois des variables dans chacune des classes. On peut utiliser la fonction `dnorm` pour calculer les densités. Commenter les graphiques.

4. On peut visualiser une estimation des lois jointes bivariées en utilisant la fonction `bkde2D` du package `KernSmooth` et la fonction `contour`.

```
library(KernSmooth) # Installer le package avant
          # si R ne reconnait pas la library
par(mfrow=c(1,3))
kde.vers = bkde2D(iris[IVE,3:4],bandwidth=.2)
image(kde.vers$x1,kde.vers$x2,kde.vers$fhat)
contour(kde.vers$x1,kde.vers$x2,kde.vers$fhat,add=TRUE)
points(iris[IVE,3:4],pch=21,bg=c("blue"))

kde.set = bkde2D(iris[ISE,3:4],bandwidth=.1)
image(kde.set$x1,kde.set$x2,kde.set$fhat)
contour(kde.set$x1,kde.set$x2,kde.set$fhat,add=TRUE)
points(iris[ISE,3:4],pch=21,bg=c("red"))

kde.virg = bkde2D(iris[IVI,3:4],bandwidth=.2)
image(kde.virg$x1,kde.virg$x2,kde.virg$fhat)
contour(kde.virg$x1,kde.virg$x2,kde.virg$fhat,add=TRUE)
points(iris[IVI,3:4],pch=21,bg=c("green"))
```

```
dev.new()
kde = bkde2D(iris[,3:4],bandwidth=.1) # Choisir une largeur de fenêtre
                                      # qui conduise à des contours "assez" lisses
image(kde$x1,kde$x2,kde$fhat)
contour(kde$x1,kde$x2,kde$fhat,add=TRUE)
points(iris[,3:4],pch=21,
       bg=c("red", "green", "blue")[as.numeric(iris$Species)])
```

5. Superposer les contours des lois de Gauss bivariées correspondantes. Commenter le graphique.

```
library(mvtnorm)
par(mfrow=c(1,3))
image(kde.vers$x1,kde.vers$x2,kde.vers$fhat)
points(iris[IVE,3:4],pch=21,bg="blue")
contour(kde.vers$x1,kde.vers$x2,kde.vers$fhat,
      add=TRUE,lty=2,levels=seq(0.1,1,by=0.1))

t1 = seq(min(iris[,3])-1,max(iris[,3])+1,by=0.01)
t2 = seq(min(iris[,4])-1,max(iris[,4])+1,by=0.01)
T1 = matrix(t1,nrow = length(t1),ncol = length(t2))
T2 = t(matrix(t2,nrow = length(t2),ncol = length(t1)))
g.vers = dmvnorm(cbind(c(T1),c(T2)),mean = mean(iris[IVE,3:4]),
      sigma = as.matrix(cov(iris[IVE,3:4])))
contour(t1,t2,matrix(g.vers,nrow = length(t1),ncol = length(t2)),
      add=TRUE,levels=seq(0.1,1,by=0.1))
```

2 Règles de décisions sous une hypothèse de normalité

2.1 Modèle homoscédastique gaussien

2.1.1 Cas univarié

1. Rappeler ce que veut dire le mot *homoscédactique*.
2. On va supposer que les espèces *virginica* et *versicolor* ont la même variance pour la longueur de pétale. Cette hypothèse vous semble t'elle raisonnable ? Pourquoi (on peut par exemple utiliser la fonction `var.test` pour justifier la réponse) ? Écrire (à la main ![1]) la règle de décision associée à la variable longueur de pétale et qui permet de séparer les espèces *virginica* et *versicolor*.
3. Avant d'exécuter les commandes ci-dessous, pariez entre vous sur le résultat qui va s'afficher ! Si vous comprenez ce que font ces lignes de code, vous devez tomber très près du résultat.

```
pv = NULL
for (k in 1:1000)
```

1. "à la main" signifie ici *en utilisant un papier et un crayon*. En effet, il n'existe pas de routine R qui donne ce genre de réponse.

```
  {pv[k] = var.test(sqrt(.26)*rnorm(50),sqrt(.26)*rnorm(50))$p.value}
sum(pv<.05)/1000
```

Interpréter le graphique suivant

```
vsim = NULL
for (k in 1:1000) {vsim[k] = var(sqrt(.26)*rnorm(50))}
hist(vsim)
abline(v=.26,col="red",lwd=1.5)
```

4. Tracer sur un même graphique les probabilités a posteriori pour ces deux espèces. Le graphique obtenu vous semble t'il cohérent ? Pourquoi ?

2.1.2 Cas bivarié

1. On suppose maintenant que les espèces *virginica* et *versicolor* ont la même covariance pour le couple longueur de pétale et largeur de pétale. Estimer la covariance puis écrire (encore à la main) et développer (toujours à la main) la règle de décision associée au couple (longueur de pétale, largeur de pétale) et qui permet de séparer les espèces *virginica* et *versicolor*.
   On peut vérifier les résultats obtenus à l'aide de la fonction `lda` du package `MASS`.

```
library(MASS) # chargement des librairies
fitl2 = lda(Species~Petal.Length + Petal.Width,data=iris)
     # analyse discriminante linéaire
```

2. Tracer sur un même graphique les contours des densités a posteriori pour ces deux espèces. Ajouter les nuages de points en choisissant des couleurs appropriées. Le graphique obtenu vous semble t'il cohérent ? Pourquoi ?
3. On veut classer de nouveaux iris n'appartenant pas à la base de données. Ajoutez les sur le graphique en utilisant la fonction `points` avec l'option `pch=10`. Utiliser la règle ci-dessus pour les classer. Calculer leurs probabilités a posteriori d'être dans chacune des deux classes.

	Long. pétale	Larg. pétale	Espèce
1	5.5	1.6	
2	6	2	
3	4	1.2	

3 Modèle hétéroscédastique gaussien

Un test de comparaison des variances montre que les variances de largeurs de pétales ne sont pas égales dans les classes *virginica* et *versicolor*. Vous pouvez le vérifier facilement à l'aide de la fonction `var.test`. On peut aussi réaliser un test basé sur la statistique de Wilks pour comparer les covariances :

```
species = iris$Species[51:150] ;
Y = cbind(iris[51:150,3],iris[51:150,4] )
ma = manova(Y ~ species)
summary(ma,test="W")
```

Reprendre les questions du cas gaussien homoscédastique bivarié sous l'hypothèse d'hétéroscédasticité. Classer de nouveau les trois iris du tableau ci-dessus.

On peut vérifier les résultats obtenus à l'aide de la fonction `qda` du package `MASS`.

```
fitq2 = qda(Species~Petal.Length + Petal.Width,data=iris)
      # analyse discriminante quadratique
```

4 Modèle non paramétrique - estimateurs à noyaux

1. Considérons le cas univarié et la variable longueur de pétales. Estimer les densités par classe par un estimateur à noyau en utilisant la fonction `density`. Tracer les densités a posteriori correspondantes. En déduire (à la main) une règle de classement.
2. Cas bivarié. Tracer les densités a posteriori.

5 Comparaison de différentes méthodes dans le cadre multivarié

On utilise le package `MASS` pour l'analyse discriminante de Fisher (cas gaussien) et le package `class` pour l'analyse discriminante de Bayes naive (plus proches voisins).

1. Ajuster les modèles.

   ```
   library(MASS) # chargement des librairies
   library(class) # pour kNN

   fitq.disl=lda(Species~.,data=iris) # analyse discriminante linéaire
   fitq.disq=qda(Species~.,data=iris) # analyse discriminante quadratique
   fitq.knn=knn(iris[,1:4],iris[,1:4],iris$Species,k=5) # k plus proches voisins
   ```

2. Estimer le risque de Bayes associé à chacune des règles en utilisant les instructions suivantes et en déduire quelle est la meilleure règle de classement au sens du risque de Bayes.

   ```
   # erreur d'apprentissage
   table(iris[,"Species"],predict(fitq.disl,iris)$class)
   table(iris[,"Species"],predict(fitq.disq,iris)$class)
   table(iris[,"Species"],knn(iris[,1:4],iris[,1:4],iris$Species,k=5))
   ```

3. Pouvez vous expliquer le résultat suivant :

   ```
   table(iris[,"Species"],knn(iris[,1:4],iris[,1:4],iris$Species,k=1))
   ```

4. Dans la question précédente, le risque de Bayes est estimé par *resubstitution*. Cette méthode d'estimation de l'erreur conduit généralement à des résultats trop optimistes et il est préférable de travailler en *validation croisée*. En pratique on retire quelques individus du jeux de données typiquement 1/3), on ajuste les modèles à partir des individus restant puis on utilise les individus retirés pour tester les modèles, autrement dit pour estimer le risque de Bayes.

```
n = length(iris[,5])
n.test = 50
i.test = sample(1:n,n.test,replace=FALSE) ;
i.train = setdiff(1:n,i.test)

fitq.disl=lda(Species~.,data=iris,subset=i.train)
fitq.disq=qda(Species~.,data=iris,subset=i.train)
fitq.knn=knn(iris[i.train,1:4],iris[i.test,1:4],iris$Species[i.train],k=5)

table(iris[i.test,"Species"],predict(fitq.disl,iris)$class[i.test])
table(iris[i.test,"Species"],predict(fitq.disq,iris)$class[i.test])
table(iris[i.test,"Species"],fitq.knn)
```

On peut faire varier le nombre de plus proches voisins pour essayer d'améliorer les résultats. Quels sont les individus mal classés ?

5. Comparer les résultats à ceux qu'on obtient avec un arbre de décision

```
library(rpart)
ad = rpart(Species~., data=iris)
plot(ad)
text(ad, use.n = TRUE)
```

6. Comparer les résultats précédentes à ceux qu'on obtient avec la régression logistique.

6 Avec Python

Les données sont disponibles dans un package. Pour les importer.

```
from sklearn.datasets import load_iris
from sklearn import tree
from sklearn import metrics
iris = load_iris()
```

6.1 Arbre de décision

On peut alors construire un arbre de décision

```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
y_pred = clf_gini.predict(iris.data)
accuracy = metrics.accuracy_score(y_pred,iris.target)
```

Il est bien sûr préférable de valider le modèle sur un ensemble de validation ou de test.

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test =  \
    train_test_split( iris.data, iris.target, test_size = 0.3, random_state = 100)
```

```
clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100,
                                  max_depth=3, min_samples_leaf=5)
clf_gini.fit(X_train, y_train)
y_pred = clf_gini.predict(X_test)
y_pred
accuracy = metrics.accuracy_score(y_pred,y_test)
```

6.2 Algorithme des plus proches voisins

```
from sklearn import neighbors, datasets
X = iris.data
y = iris.target
n_neighbors = 15
clf = neighbors.KNeighborsClassifier(n_neighbors, weights='uniform')
clf.fit(X, y)
accuracy.knn = metrics.accuracy_score(y_pred,iris.target)
```

Adapter le programme pour valider cette règle de décision sur le même ensemble de test que l'arbre construit à la question précédente.