Master MAS de Bordeaux
M2 : apprentissage supervisé

TP1 : Introduction via les k plus proches voisins

Exercice 1. Les objectifs :

- Appliquer les k plus proches voisins sur un exemple.
- Découvrir les notions d'erreur d'apprentissage et d'erreur test.
- Comprendre l'importance du choix du paramètre k.

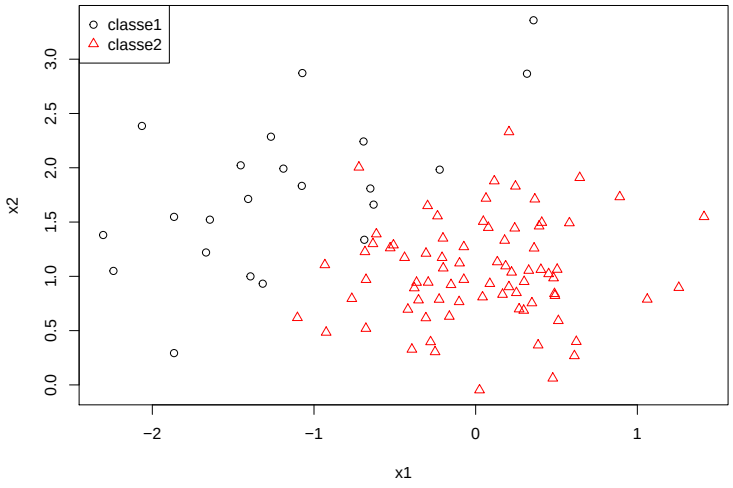
Récupérer les jeux de données synth_train.txt et synth_test.txt. On a $Y \in \{1, 2\}$ et $X = (X^1, X^2) \in \mathbb{R}^2$. On dispose de 100 données d'apprentissage et 200 données test.

1. Charger le jeu de données d'apprentissage dans R avec la commande `read.table`.

```
train <- read.table(file="../data/synth_train.txt", header=TRUE)
Xtrain <- train[,-1] # matrice des entrées
Ytrain <- train$y # vecteur des sorties (les classes des entrées)
```

2. Représenter graphiquement les données d'apprentissage à l'aide de la fonction `plot`.

```
plot(Xtrain, pch=Ytrain, col=Ytrain)
legend("topleft", legend=c("classe1", "classe2"), pch=1:2, col=1:2)
```



3. On veut prédire les classes de deux nouveaux points de coordonnées (0,0) et (-2,1) et estimer leurs probabilités à posteriori d'appartenir à chaque classe. Pour cela on utilise la méthode des k plus proches voisins (avec $k = 30$ voisins) et la fonction `knn` (pour k nearest neighbors) du package `class` :

```
library(class)
# les deux points à prédire sont mis dans une matrice "Xtest"
Xtest <- matrix(c(0,0,-2,1), nrow=2, byrow=TRUE)
```

```r
# tous les calculs de la fonction knn sont stockés dans "pred"
pred <- knn(Xtrain, Xtest, Ytrain, k=30, prob=TRUE)
# Contenu de l'objet "pred"
pred
```

```
## [1] 2 1
## attr(,"prob")
## [1] 1.000 0.567
## Levels: 1 2
```

```r
attr(pred, "prob") # permet de récupérer les probabilités à posteriori
```

```
## [1] 1.000 0.567
```

Les résultats stockés dans l'objet `pred` nous donnent les informations suivantes :

— La première ligne indique que le premier point $(0, 0)$ (première ligne de Xtest) est affecté à la classe 2 et le second point $(-2, 1)$ (seconde ligne de Xtest) est affecté à la classe 1.
— La ligne `attr(,"prob")` indique que la probabilité (à postersiori) que le premier point soit dans la classe 2 (classe d'affectation) est égale à 1. La probabilité que le second point appartienne à la classe 1 est égale à 0.56.
— La ligne `Levels` indique simplement les libellés des classes.

Pourquoi la prédiction du second point est moins sûre que celle du premier ? Regarder la représentation graphique des données d'apprentissage ci-dessus pour vérifier.

4. On veut maintenant prédire (toujours avec $k = 30$ voisins) les classes des 100 points de l'échantillon d'apprentissage et en déduire un taux d'erreur d'apprentissage. Pour cela on utilise le code ci-dessous :

```r
pred_train <- knn(Xtrain, Xtrain, Ytrain, 30)
head(pred_train) # les 5 premiers points sont affectés à la classe 2
```

```
## [1] 2 2 2 2 2 2
## Levels: 1 2
```

```r
table(pred_train,Ytrain) # 10+0 données d'apprentissage sont mal prédites
```

```
##           Ytrain
## pred_train  1  2
##          1 12  0
##          2 10 78
```

```r
# Il y a 10/100 données d'apprentissage mal prédites
sum(pred_train!=Ytrain)/length(Ytrain) # taux d'erreur d'apprentissage
```

```
## [1] 0.1
```

On obtient ainsi au taux d'erreur d'apprentissage pour la fonction `knn` (et 30 voisins) de 10%.

5. Charger le jeu de données test dans R.

```r
test <- read.table(file="../data/synth_test.txt", header=TRUE)
Xtest <- test[,-1] # matrice des entrées
Ytest <- test$y    # vecteur des sorties (les classes des entrées)
```

6. On veut maintenant prédire (toujours avec $k = 30$ voisins) les classes des 200 points de l'échantillon test et en déduire un taux d'erreur test. Pour cela on utilise le code ci-dessous :

```r
pred_test <- knn(Xtrain, Xtest, Ytrain, 30)
head(pred_test) # les 5 premiers points sont affectés à la classe 2
```

```
## [1] 2 2 2 2 2 2
## Levels: 1 2
```

```r
table(pred_test,Ytest) # 36+0 données test sont mal prédites
```

```
##          Ytest
## pred_test   1   2
##         1  27   0
##         2  35 138
```

```r
# Il y a ?/200 données test mal prédites
sum(pred_test!=Ytest)/length(Ytest) # taux d'erreur test
```

```
## [1] 0.175
```

On obtient un taux d'erreur test d'environ 17%.

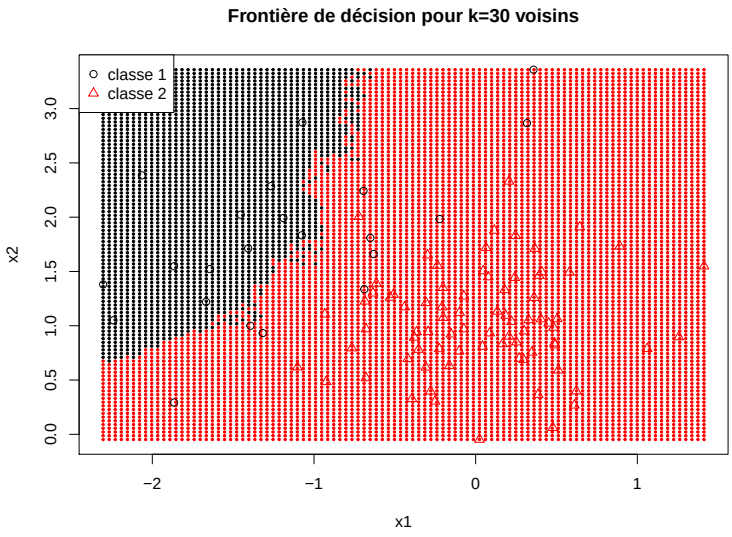
En executant plusieurs fois les lignes de code ci-dessus, vous pouvez obtenir des prédictions légèrement différentes et donc des taux d'erreurs test légèrements différents. En effet, en cas d'égalité dans les proportions de chaque classe parmi les voisins, un choix aléatoire est réalisé par la fonction `knn` ce qui peut conduire à des prédictions différentes. Comment éviter ce problème ?

On constate que le taux d'erreur test (17%) est plus grand que celui d'apprentissage (10%). C'est normal. En effet le taux d'erreur d'apprentissage a été obtenu en utilisant les mêmes données pour apprendre et prédire ce qui correspond à une situation de sur-apprentissage. L'erreur d'apprentissage est donc toujours une estimation trop optimiste du vrai taux d'erreur de la méthode.

Remarque : pour évaluer une méthode, il faut toujours estimer ses performances sur des données test qui n'ont jamais été utilisées pour apprendre.

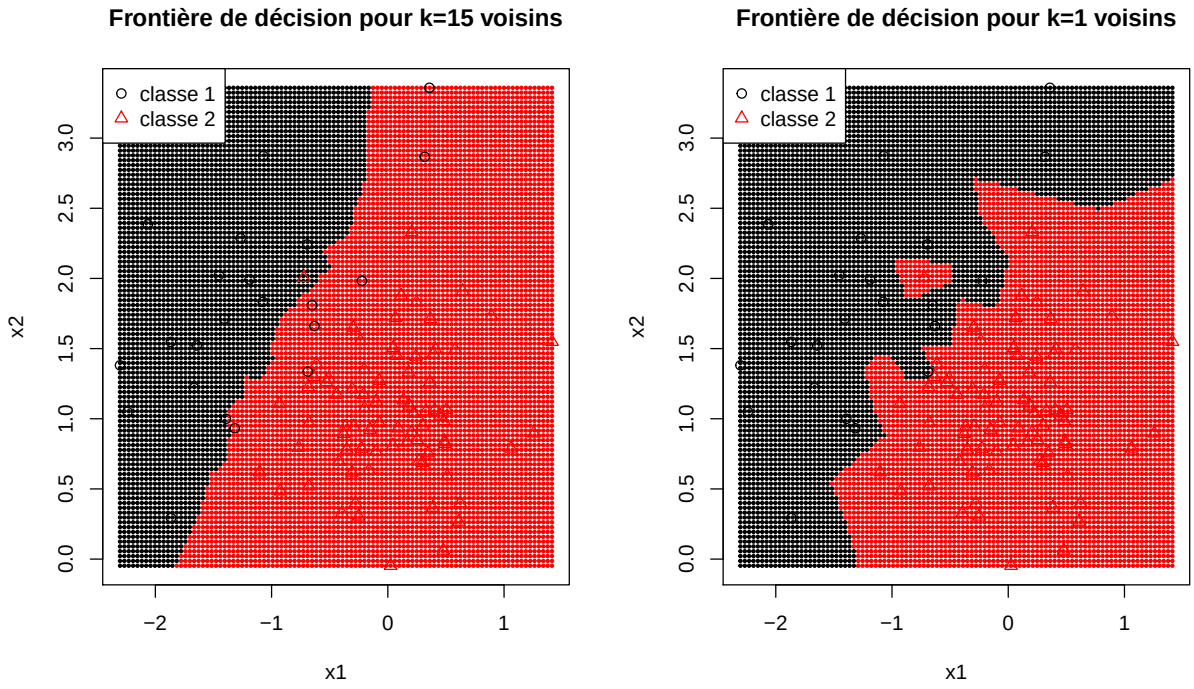
7. Afin de comprendre l'influence du choix du nombre k de voisins, on représente graphiquement la frontière de décision pour $k = 30$ voisins. Pour cela, le code ci-dessous prédit les classes d'une grille de points et représente les points de la grille, les points de l'échantillon d'apprentissage et leurs prédictions sur un graphique.

```r
load("../data/grille.rda")
# avec k=30 voisins
pred_grille_30 <- knn(Xtrain, grille, Ytrain, 30)
plot(grille, pch = 20, col = pred_grille_30, cex = 0.5,
     main="Frontière de décision pour k=30 voisins")
points(Xtrain, pch=Ytrain, col=Ytrain)
legend("topleft", legend=c("classe 1", "classe 2"), pch=1:2, col=1:2, bg="white")
```



On note que cette frontière de décision semble décalée vers la gauche. Cette frontière risque de conduire à de meilleures prédictions des points de la classe 2 que des points de la classe 1.

8. Recommencer avec $k = 15$ voisins puis $k = 1$ voisins pour retrouver les graphiques ci-dessous.



On visualise sur ces graphiques l'influence de choix du paramètre k (nombre de voisins) sur la qualité de la prédiction. En effet avec trop peu de voisins (1 voisin), ou trop de voisins (30 voisins) on ne retrouve pas bien la frontière de décision. Dans le premier cas on colle trop aux données et dans le second pas assez. Il faudra donc calibrer ce paramètre pour prédire au mieux. Ce sera l'objet du TP3.

Exercice 2. Appliquer les k plus proches voisins à la reconnaissance automatique de caractères manuscrits.

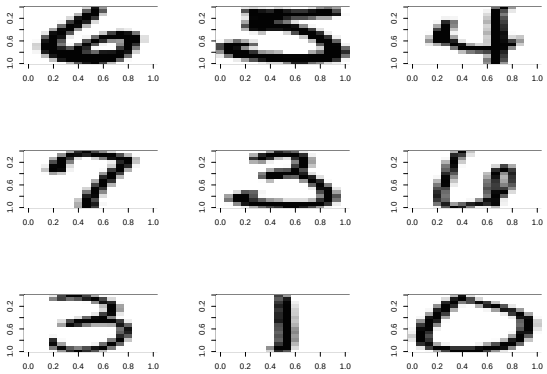
On s'intéresse aux données `numbers_train.txt` et `numbers_test.txt`. Chaque fichier contient 500 images de dimension 16×16 et chaque image représente un caractère manuscript (un chiffre entre 0 et 9). On a donc $Y \in \{0, \ldots, 9\}$ et $X = (X^1, \ldots, X^{256}) \in \mathbb{R}^{256}$. Il s'agit d'images en niveaux de gris où chaque pixel prend une valeur entre 0 (noir) et 1 (blanc).

1. Importer les 500 images du fichier `numbers_train.txt`.

```
data <- read.table("../data/numbers_train.txt", header=TRUE)
Xtrain <- as.matrix(data[,-1])
Ytrain <- as.factor(data[,1])
```

2. Visualiser les neuf premières images.

```
par(mfrow=c(3,3))
for (i in 1:9){
  image(matrix(Xtrain[i,],16,16), col=gray(1:100/100), ylim=c(1,0))
}
```



3. Prédire avec la méthode `knn` les classes des 500 images de l'ensemble d'apprentissage avec la valeur de k par defaut (k=1 voisin) et calculer le taux d'erreur d'apprentissage.
4. Importer les 500 images du fichier `numbers_test.txt`.
5. Prédire maintenant les classes des 500 images de l'ensemble test (toujours avec $k = 1$ voisin) et calculer le taux d'erreur test.
6. Essayer de faire varier la valeur de k pour améliorer le taux d'erreur test et choisir une valeur de k.