TP 1: L'algorithme des k - plus proches voisins

emilie.kaufmann@univ-lille1.fr

Présentation

L'objectif de ce TP est de vous familiariser avec la manipulation de données sous forme de dataframe dans R, et de tester un premier classifieur, celui des k plus-proche voisins.

Le début du TP vous est donné dans le fichier TP1.Rmd (au format R Markdown) que l'on pourra compléter.

1 Exploration du jeu de données

La base de données `iris.dat` contient les caractéristiques de plusieurs iris, ainsi que leur espèce. La commande `read.csv()` permet d'importer les données, en sélectionnant le type de séparateur, la présence d'une en-tête ou d'autres paramètres (un coup d'oeil à la base de données ou la commande Import Dataset permet d'aider à les choisir). En plus d'importer les données sous forme de dataframe, on nomme ici chaque variable (en accord avec la description donnée dans `iris.names`) :

```
iris.data <- read.csv('iris.data',sep=',',header=F)
names(iris.data) <- c('Sepal.Length','Sepal.Width','Petal.
    Length','Petal.Width','Species')
```

Observer l'effet des fonctions `head()`, `str()` et `summary()` appliquées au dataframe `iris.data`. Ces commandes sont cruciales pour un premier contact avec une base de données, afin d'avoir une idée de ce qu'elle contient.

L'objectif du TP est d'arriver à prédire l'espèce d'une iris à partir des quatre autres variables (quantitatives) donnant la longueur des pétales et des sépales. Dans un premier travail exploratoire, on cherche à étudier l'influence des différentes variables sur la variable à prédire, `Species`.

Quel est l'effet de la commande suivante ? Visualiser la distribution des quatre variables explicatives à travers les espèces. Lesquelles vous paraissent les plus discriminantes ?

```
boxplot(iris.data$Petal.Length ~ iris.data$Species)
```

On peut également regarder comment un couple de variables est distribué à travers les espèces en affichant un nuage de point étiqueté par les espèces avec la fonction plot :

```
palette <- c("red","green3","blue")
plot(iris.data$Sepal.Length,iris.data$Sepal.Width,col=palette
    [iris.data$Species])
```

La fonction `pairs()` permet d'afficher simultanément les nuages de points de toutes les paires de variables.

```
pairs(iris.data[1:4],pch = 21,bg = c("red", "green3", "blue")
    [iris.data$Species])
```

En se basant sur toutes ces visualisations préliminaires, quelle paire de variables semble la plus utile pour prédire l'espèce ?

2 Classifieur des k-plus proches voisins

Dans un problème d'apprentissage, le but est de construire un classifieur à partir d'une base d'apprentissage. Pour évaluer ce classifieur, on regarde ses performances de prédiction sur une nouvelle base de données, la base de test. Ici on ne nous donne qu'une base de données. On va donc artificiellement la séparer en deux, créant ainsi une base d'apprentissage, que nous appelerons `train` et un base de test, `test`.

Pour faire cela en pratique, on peut se servir de la commande `sample(n, nTest)` qui retourne une liste de `nTest` entiers entre 1 et `n` choisis aléatoirement. Construire `train` et `test`.

```
ind_test <- sample(n, nTest);
test <- iris.data[ind_test,]
```

On pourrait écrire nous même une fonction qui, pour un point $x \in \mathbb{R}^4$, calcule ses k plus proches voisins dans la base de données et renvoie l'espèce majoritaire parmi ceux-ci. Une implémentation efficace existe toutefois déjà dans R avec la fonction `knn()` du package `class`.

La syntaxe suivante permet de calculer les étiquettes prédites pour les entrées de la base de test, ici pour $k = 3$:

```
library(class)
prediction <- knn(train[1:4], test[1:4], train$Species, k=3)
```

Que contient `prediction` ? Calculer l'erreur de test, c'est-à-dire la fraction d'étiquettes mal prédites dans la base de test.

On pourra aussi comparer les prédictions avec les vraies espèces en affichant une table de contingence :

```
confusion <- table(prediction, test$Species)
```

Vous pouvez tester plusieurs valeurs de k. Le classifieur des k-ppv vous parait-il avoir de bonnes performances sur ce jeu de données ?

3 Choix du nombre de voisins

Pour toutes les valeurs de k entre 1 et la taille de la base d'apprentissage, calculer l'erreur d'apprentissage et l'erreur de test du classifieur des k-plus proches voisins et présenter le résultat sur un graphique. Pour faire cela en pratique, on pourra faire une boucle et stocker dans deux tableaux les valeurs des erreurs obtenues.

```
kValues <- c(1:nTrain); trainErrors <- array(0,nTrain);
testErrors <- array(0,nTrain);
for (i in 1:nTrain){
  nb <- kValues[i];
  prediction<- knn(train[1:4], test[1:4], train$Species, k=nb);
  [...]
}
```

Quelle valeur de k donne l'erreur de test la plus petite ?

Vous paraît-il pertinent de choisir la valeur de k en se basant sur l'ensemble de test ? Par la suite on prendra l'habitude de séparer le jeu de données en trois parties :

- **ensemble d'apprentissage** : pour construire le classifieur (ex. k-nn pour différentes valeurs de k)
- **ensemble de validation** : pour sélectionner le meilleur classifieur (ex. choix de k pour k-nn)
- **ensemble de test** : pour estimer le risque du classifieur proposé

4 Visualisation de la règle de classification

On a remarqué précédemment que deux attributs semblaient suffisants pour prédire l'espèce de chaque plante. A k fixé, écrivez une fonction `knnclassifier()` qui étant donné un dataframe à deux colonnes X contenant les caractéristiques choisies, renvoie le vecteur des prédiction k-nn se basant sur ces deux caractéristiques seulement.

```
k <- 2
knnclassifier <- function(X){}
```

Le code ci-dessous permet de visualiser le classifieur k-nn basé sur les caractéristiques de pétales. Observez comment ce dernier varie avec k.

```
N <- 100; X <- array(0, dim = c(N^2, 2));
# points extremaux
a <- 0; b <- max(iris.data$Petal.Length)+1;
c <- 0; d <- max(iris.data$Petal.Width)+1;
# generer des points regulierement espaces
for(i in 0:N) for(j in 1:N)
{X[(N-1)*i+j,] <- c(a+(b-a)*i/N, c + (d-c)*j/N)}
# affichages des points generes et des labels predits
points(X[,1], X[,2], col=palette[knnclassifier(X)], pch=15,
    cex=1.7);
# affichage de l'ensemble d'apprentissage
symbols=c(16,8,17)
points(train$Petal.Length, train$Petal.Width, col="black",pch=
    symbols[train$Species])
```