Classification supervisée
k-plus proches voisins et arbres de décision CART

Charlotte Baey (charlotte.baey@univ-lille.fr)

1 Classification de vins

L'objectif de cet exercice est d'apprendre à classer différents types de vins en fonction de certaines de leurs caractéristiques. On dispose pour cela d'un jeu de données portant sur 178 vins, pour lesquels on a relevé :

- la teneur en alcool `alcool`
- la teneur en acide malique `malid`
- la teneur en cendres `cendres`
- l'alcalinité des cendres `alcal`
- la teneur en magnésium `mg`
- la teneur totale en phénols `phenols`
- la teneur en flavanoïdes `flava`
- la teneur en phénols non flavanoïdes `nonflava`
- la teneur en tanins `tanins`
- l'intensité de la couleur `couleur`
- la teinte du vin (mesure de spectométrie) `teinte`
- la mesure OD280/OD315 du vin dilué `od280`
- la teneur en proline `proline`

1. Importer la base de données `wines.csv`, et faites une analyse descriptive de la base de données. Afficher les boxplots de chaque variable explicative en fonction de la variable de classe.
2. Transformer la variable `Classe` en facteur (à l'aide de la commande `as.factor`).
3. Diviser aléatoirement le jeu de données en deux parties, une partie entraînement (75% du jeu de données) et une partie test (les 25% restants).
4. Écrire une fonction `knn` qui prend en argument :
   - un entier `k`,
   - un jeu de données d'entraînement `d_train` qui ne contient que les variables explicatives
   - un jeu de données test `d_test` qui ne contient que les variables explicatives
   - une variable `labels_train` contenant les identifiants de classe des individus de la table `d_train`

   et qui renvoie un classement pour chaque point de `d_test`, en utilisant la distance euclidienne.
5. Utiliser la fonction `knn` pour choisir le nombre de voisins k par validation croisée (leave-one-out cross validation).

6. A l'aide de la valeur optimale de k trouvée à la question précédente, proposer un classement pour les observations de la base de test. Quel est le taux de bon et de mauvais classement ?

Pour aller plus loin ... : les k plus proches voisins pondérés

1. Écrire une fonction `knn_pond` qui prend en argument un entier `k`, un noyau `kernel`, une distance `dist`, un jeu de données d'entraînement `d_train` et un jeu de données test `d_test` contenant uniquement les variables explicatives, une variable `labels_train` contenant les identifiants de classe des individus de la table `d_train`, et qui renvoie un classement pour chaque point de `d_test`.
2. Utiliser la fonction `knn_pond` pour choisir le nombre de voisins par validation croisée sur le jeu de données d'entraînement, en testant plusieurs valeurs de k (mais sans faire une recherche exhaustive sur toutes les valeurs possibles de k). Comparer les résultats avec différents noyaux et différentes distances. Choisissez le meilleur modèle au sens du taux de bon classement.
3. Quelles sont les performances du modèle sur le jeu test ?

Il existe sous `R` deux fonctions permettant de faire des k-plus proches voisins, classiques ou pondérés :

— la fonction `knn` du package `class`

— la fonction `kknn` du package `kknn` (cette fonction peut paraître déroutante car elle gère également la prédiction de variables quantitatives ou ordinales, il faut donc bien s'assurer que les variables qualitatives sont considérées comme des facteurs)

2 Classification de courriers électroniques

Dans cet exercice, on souhaiterait trouver une règle de décision pour classer des courriers électroniques comme 'spam' ou 'non spam'. On dispose pour cela d'une base de données contenant 4601 courriers électroniques, classés dans deux catégories, 'spam' ou 'non spam'. Pour chacun de ces courriers, on dispose de 57 variables explicatives, la plupart (54 variables) correspondant à la fréquence de certains mots ou caractères dans le texte du courrier (ex. fréquence d'apparition du caractère `$`, ou du mot `money`), les trois dernières étant liées aux nombre de lettres majuscules dans le mot (nombre moyen de lettres dans les mots écrits en majuscules, nombre de lettres dans le plus long mot écrit en majuscules, et nombre total de lettres majuscules dans le texte).

1. Importer la base de données `spam`, accessible dans le package `kernlab`.
2. Diviser la base en une partie apprentissage et une partie test, de tailles respectives 75% et 25% de la base totale.
3. En utilisant le package `rpart`, construire un arbre de décision binaire de type CART, en utilisant la base d'apprentissage et en comparant les résultats obtenus pour différents seuils du coefficient de coût complexité (option `cp=`).
4. À partir de la plus grande séquence d'arbres emboîtées possible (i.e. avec `cp=0`), identifier la valeur du coefficient de coût complexité à partir de laquelle le taux d'erreur se stabilise et en déduire l'arbre optimal associé.

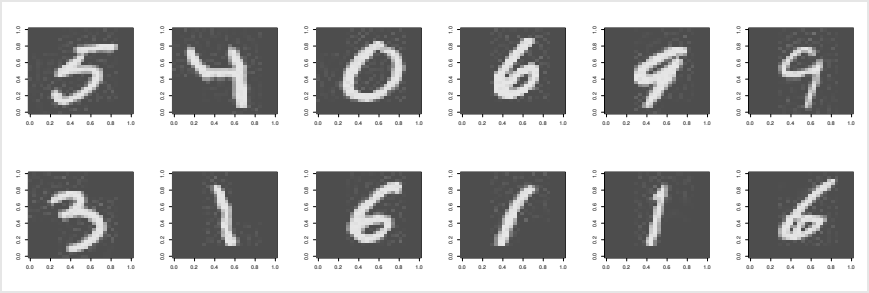
5. Comparer avec le taux de mauvais classement obtenu, avec cet arbre optimal, sur la base de test.

Package `caret`

Il existe un package dédié aux méthodes de classification et de régression : `caret` (pour Classification And REgression Training). Ce package permet de faire, entre autres, la séparation base d'apprentissage et base de test, de la validation croisée, de la comparaison de méthodes. Il permet également d'unifier le code en terme de syntaxe, car l'utilisateur fait appel à une unique fonction `train` pour entraîner le modèle (cette fonction fait, en interne, appel aux différents packages R nécessaires et se charge de convertir la syntaxe initiale en une syntaxe adaptée aux package en question).
Certains apprécieront sa flexibilité et sa facilité d'utilisation, d'autres pourront lui reprocher un côté "boîte noire" et préféreront coder par eux-mêmes. Libre à vous d'essayer et de choisir ce qui vous convient le mieux !

Pour aller plus loin ... : base de données MNIST

On travaille dans cet exercice avec un extrait de la base de données MNIST. Cette base de données contient des images représentant plusieurs versions manuscrites des chiffres de 0 à 9. Chaque image est de taille 28×28 pixels (voir figure ci-dessous).



Les données sont fournies dans le dossier `mnist.zip`, qui contient 10 dossiers numérotés de 0 à 9. Chaque dossier contient plusieurs images au format jpeg du chiffre correspondant.

1. Importer les données sous R, en créant une base de données qui pourra être exploitée par la fonction `rpart`. Ne pas oublier de créer la variable contenant le label de chaque image. ***Indication** : on pourra utiliser la fonction* `list.files` *qui prend en argument le chemin d'un dossier et le type de fichier, et renvoie la liste de tous les fichiers de ce type dans le dossier indiqué. Taper* `?list.files` *pour obtenir plus de détails sur cette fonction.*
2. Créer un échantillon d'apprentissage et un échantillon test à partir de la base de données créée en 1.
3. Construire un arbre de décision binaire de type CART sur la base d'apprentissage.