SY09 Printemps 2019
TP 11 — Arbres de décision

1 Arbres de décision avec R

1.1 Fonctions

On commencera par se familiariser avec l'implémentation `R` de l'algorithme `CART`. On chargera tout d'abord les packages `rpart` et `rpart.plot`. On analysera les fonctions suivantes.

1. La fonction `rpart` permet de faire l'apprentissage d'un modèle d'arbre (pour construire l'arbre complet, on prendra soin de fixer `control=rpart.control(minsplit=1, cp=0)`).
2. La fonction `prune` permet d'élaguer l'arbre ainsi obtenu, en choisissant une valeur du paramètre λ de coût-complexité (argument `cp`).
3. La fonction `predict` permet de classer un ensemble de données au moyen d'un arbre de décision préalablement appris.

On pourra utiliser la fonction `prp` (package `rpart.plot`) pour afficher un objet de type `rpart`, ainsi que la fonction `front.tree` fournie pour afficher les frontières de décision (dans le cas de données en dimension $p = 2$).

1.2 Questions

1. On considérera tout d'abord les données `iris`. Construire l'arbre complet permettant de prédire l'espèce à partir des variables explicatives `Sepal.Length` et `Petal.Length`, en utilisant toutes les données pour apprendre le modèle. Afficher l'arbre complet et les frontières de décision correspondantes.
2. Comment accéder à la séquence de sous-arbres emboîtés entre l'arbre complet et la racine ? Afficher les valeurs du paramètre λ de coût-complexité et les informations correspondantes (taille de l'arbre, erreur, etc).
3. Déterminer le sous-arbre optimal, c'est-à-dire correspondant à l'erreur de validation la plus faible, et l'afficher.

2 Application

2.1 Test sur données simulées

On souhaite comparer les performances de la régression logistique (linéaire et quadratique) sur les jeux de données simulées `Synth1-1000`, `Synth2-1000` et `Synth3-1000`. Pour ce faire, on utilisera le même protocole expérimental que précédemment, en répétant $N = 20$ fois les étapes suivantes pour chaque jeu de données :

1. séparer le jeu de données en un ensemble d'apprentissage et un ensemble de test ;
2. apprendre le modèle d'arbre sur l'ensemble d'apprentissage,
3. effectuer le classement des données de test et calculer le taux d'erreur associé.

Pour chaque jeu de données, calculer le taux d'erreur (de test) moyen sur les $N = 20$ séparations effectuées.

On pourra s'appuyer sur les frontières de décision obtenues pour analyser les résultats. Sachant que les données suivent dans chaque classe une loi normale multivariée, comment peut-on interpréter ces résultats ? Les comparer aux résultats obtenus au moyen des classifieurs précédemment étudiés.

2.2 Test sur données réelles

2.2.1 Données `Pima`

On souhaite appliquer les arbres de décision à deux jeux de données déjà considérés précédemment :

— la prédiction du diabète chez les individus d'une population d'amérindiens (données `Pima`) :

```
> Pima <- read.csv("donnees/Pima.csv", header=T)
> X <- Pima[,-8]
> z <- as.factor(Pima[,8])
```

— la détection de spams (données `spambase` puis spambase2) :

```
> Spam <- read.csv("donnees/spambase.csv", header=T, row.names=1)
> X <- Spam[,-58]
> z <- as.factor(Spam[,58])
```

Pour chacun de ces jeux de données, on répétera la procédure de séparation des données, apprentissage, classement et calcul du taux d'erreur. On comparera les résultats obtenus à ceux obtenus avec les méthodes étudiées précédemment (analyse discriminante, régression logistique).