Apprentissage par Arbres de Décisions

BRETONNIERE Corentin / SERREAU Antoine / GUIGON Benjamin

January 2021

1 Introduction

Les arbres de décisions modélisent une hierarchie de test pour prendre une décision ou prédire un résultat en fonction des expériences précédentes.

Il existe 2 types d'arbres de décisions pour la prédiction :

- **Les arbres de régressions (Regression Tree)** permettent de prédire une réponse quantitative (par exemple le prix d'une maison).
- **Les arbres de classifications (Classification Tree)** permettent de prédire une réponse qualitative (quel maladie a quelqu'un en fonction de ses symptomes).

Leur but est de prédire la variable cible de sortie depuis plusieurs variables d'entrée connue. Un arbre est constitué de noeuds avec des arrètes qui en découlent jusqu'au noeud suivant ainsi de suite, l'arbre se termine lorsque l'ajout de noeuds n'améliore pas la prédiction.

Chaque noeud est un test sur une variable qui affine la prédiction, pourvu que le test soi pertinent. Chaque branche présente donc un resultat du test. Pour avoir des test pertinents et éviter les biais de séparation il faut "élager" l'arbre en effectuant des tests qui maximisent un critère donné, ensuite ce critère est de nouveau soumis à un test, ainsi de suite.

L'ensemble d'apprentissage est noté comme suit :

$$(\mathrm{x},\, Y) = (x_1, x_2, x_3, ..., x_k, Y)$$

La variable Y désigne la variable cible à prédire, les x sont les variables d'entrées qui constituent le vecteur x et qui serviront à prédire Y.

2 Arbre de régression

A la différence des arbres de classification, les arbres de régressions sont constitués de variables *quantitatives*, à chaque noeud on associe une *coupure*, à chaque coupure on associe une *variable de coupure* X^{j_t} selon laquelle on va découper le noeud.
Pour qu'un noeud soi pertinent, le seuil de coupure doit avoir une valeur qui maximise un caractère, on note ce *seuil de coupure* $\emptyset_t$:

- Si $X_i^{j_t} \leq \emptyset_t$ alors on continue dans la branche de **gauche**
- Si $X_i^{j_t} \leq > \emptyset_t$ alors on continue dans la branche de **droite**

2.1 Pureté

On considère un noeud pur si tous les individus associés à une valeurs appartiennent effectivement à cette classe. Les illustrations ci-jointes sont issues de la vidéo youtube *"Tout savoir sur les arbres de décision - LES MODELES D'ARBRES 1"* de la chaine *"AIforyou - Morgan Gautherot"*, elles illustrent très bien le concept de *pureté*.

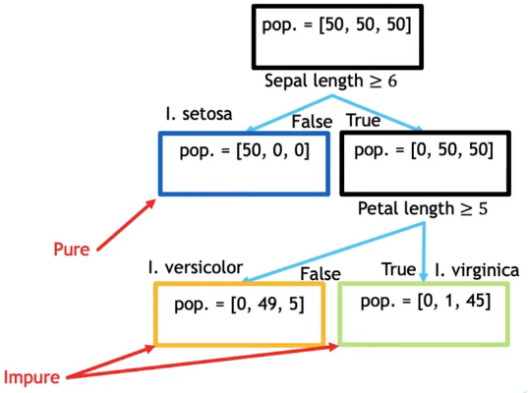

Figure 1: Illustration pureté d'un noeud

Sur cette illustration reprenant la base de donnée Iris où les seuils de coupures choisis sont quantitatif, le premier noeud est pur car 100% des setosa vont

bien dans la branche de gauche. Le 2ème noeud est impur car 5 virginicas se retrouvent dans la catégorie des versicolors et une versicolor se retrouve avec les virginicas.

La **pureté** d'un noeud se mesure avec l'indice de Gini, plus la valeur de l'indice est proche de 0, plus le noeud est pur.

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

La variable $p_{i,k}$ est la probabilité d'avoir un individu de la classe k parmi la population du $i^{ème}$ noeud.

La pureté du 1^{er} noeud noir est :
$G_{1noir} = 1 - \left(\frac{50}{100}\right)^2 - \left(\frac{50}{100}\right)^2 - \left(\frac{50}{100}\right)^2 = 0{,}667$

La pureté du noeud bleu est :
$G_{bleu} = 1 - \left(\frac{50}{50}\right)^2 - \left(\frac{0}{50}\right)^2 - \left(\frac{0}{50}\right)^2 = 0$ Ce noeud est pur.

La pureté du $2^{ème}$ noeud noir est :
$G_{2noir} = 1 - \left(\frac{0}{50}\right)^2 - \left(\frac{50}{100}\right)^2 - \left(\frac{50}{100}\right)^2 = 0{,}5$

La pureté du noeud jaune est :
$G_{jaune} = 1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 = 0.168$

La pureté du noeud vert est :
$G_{vert} = 1 - \left(\frac{0}{46}\right)^2 - \left(\frac{1}{46}\right)^2 - \left(\frac{45}{46}\right)^2 = 0.043$

2.2 Coût du noeud

Avec cette notion de *coût du noeud* on va mesurer à quel point le choix de la variable de décision est bonne.

$$J(k) = \left(\frac{m_{gauche}}{m}\right) G_{gauche} + \left(\frac{m_{droite}}{m}\right) \mathrm{G}_{droite}$$

Où $\mathrm{G}_{gauche/droite}$ mesure l'impureté des noeuds descendant de droite et de gauche,
Et $m_{gauche/droite}$ est la proportion de la population dans chacun des noeuds descendant.

Ci dessous plusieurs choix de noeuds pour la première coupure de la base de donnée Iris, on privilégiera celui qui à un coût le plus faible :

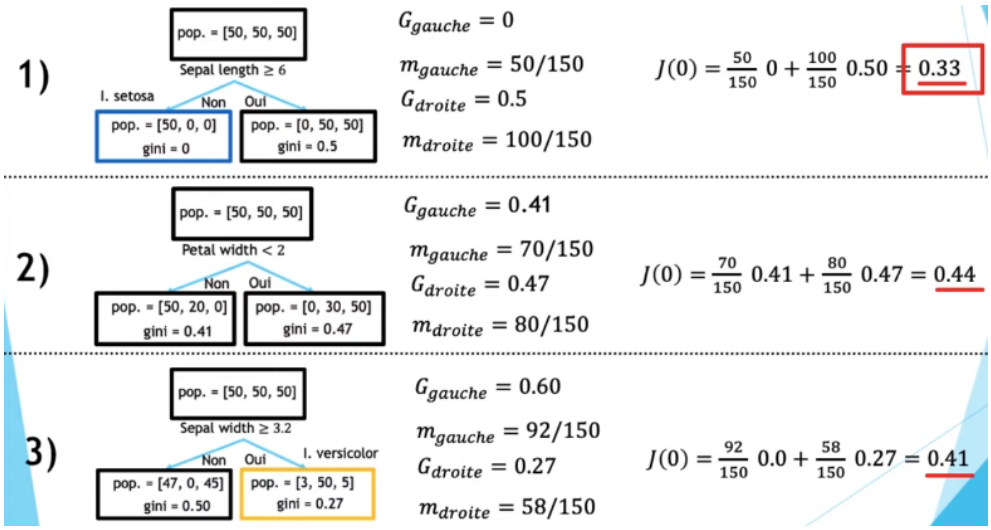

Figure 2: Coût des noeuds

2.3 Exemple : Base de donnée Iris avec R

Cet exemple est inspiré du travail de Christophe Chesnau : "Introduction aux arbres de décisions".

On vas chercher à donner une valeur plausible de y_* de `Petal.Width` à un individu dont on connaît déjà la valeur des autres carctères, il y à également un caractère qualificatif qui est `species`. On utilise alors un arbre de regression modélisé sur R :

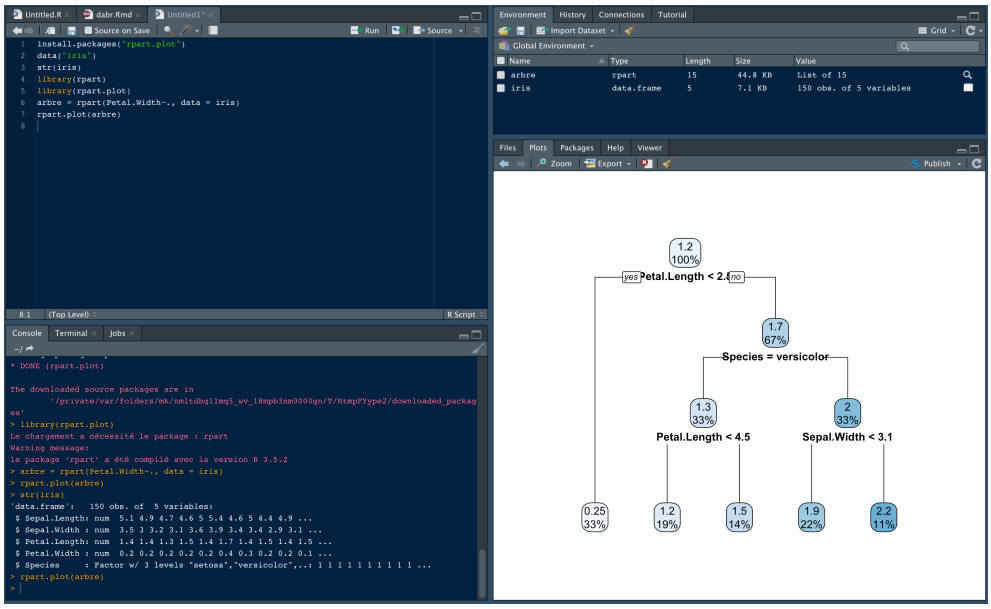

Figure 3: Code R de l'arbre de régression de la base de donnée Iris

On remarque qu'à la 2ème étape c'est le caractère qualicatif qui sert de coupure est la modalité seuil est `versicolor`.

Pour avoir une valeur cohérente de `Petal.Width` pour un individu vérifiant `Sepal.Length` $= 5.7$, `Sepal.Width` $= 3.5$, `Petal.Length` $= 1.6$ et `Species` $=$ `Setosa`, on fait :

```r
install.packages("rpart.plot")
data("iris")
str(iris)
library(rpart)
library(rpart.plot)
arbre = rpart(Petal.Width~., data = iris)
rpart.plot(arbre)
predict(arbre, newdata = data.frame(Sepal.Length = 5.7, Sepal.Width = 3.5,Petal.Length = 1.6, Species = "setosa"))
```

Figure 4: Code R de la prédiction par régression

```
> predict(arbre, newdata = data.frame(Sepal.Length = 5.7, Sepal.Width = 3.5,Petal.Length = 1.6, Species = "setosa"))
    1
0.246
>
```

Figure 5: Resultat de la prédiction

Cela indique que la valeur prédite pour `Petal.Width` est $y_* = 0{,}246$.

On peux aussi sur R rapidement trouver la valeur prédite avec une *regression logistique* (beaucoup plus précise qu'un arbre de régression aussi "simple") :

```
10  reg = lm(Petal.Width~., data = iris)
11  predict(reg, newdata = data.frame(Sepal.Length = 5.7, Sepal.Width = 3.5,Petal.Length = 1.6, Species = "setosa"))
12
```

Figure 6: Code R de la prédiction par régression logistique

```
> reg = lm(Petal.Width~., data = iris)
> predict(reg, newdata = data.frame(Sepal.Length = 5.7, Sepal.Width = 3.5,Petal.Length = 1.6, Species = "setosa"))
        1
0.2323665
>
```

Figure 7: Resultat de la prédiction

Cela indique que la valeur prédite par la régression logistique pour `Petal.Width` est $y_* = 0{,}232$, donc une valeur très proche de la régression simple précédente. Cependant, l'avantage de l'arbre de régression simple est la visualisation très claire des caractères discriminants avec le seuil associé.

3 Arbre de Classification

Les noeuds des arbres de classification traitent de caractères *qualificatif*, la variable qualitative est notée ${X_i}^{j_t} \in$ R.

Après le test du noeud, l'observation rejoint un groupe parmi $\{A_t, {}^cA_t\}$:

- Si ${X_i}^{j_t} \in A_t$ alors on continue dans la branche de **gauche**
- Si ${X_i}^{j_t} \notin A_t$ alors on continue dans la branche de **droite**

3.1 Indice de Gini et coût du noeud

De la même manière que pour les arbres de régression, il est pertinent de calculer les indices de Gini et le coûts de chacun des noeuds, cela ce permet de ce rendre compte si les optimums locaux sont atteints.

Cependant il faut faire attention car la recherche d'optimum locaux (au niveau des noeuds) ne conduit pas systématiquement à un arbre optimal dans sa globalité, c'est une des limites des **CART** (Classification And Regression Trees).

4 Limites

Les arbres de décisions ont cependant quelques limites, parmis elles on peux citer :

- **Problème du NP complet.**
- **Instabilité :** sensible à des fluctuations d'échantillon, si il y a une petite variation l'arbre peux changer du tout au tout.
- **Problème de sur-apprentissage :** plus l'arbre est grand plus il vas faire du cas par cas et on entre dans le biais de sur-apprentissage.

Le **Random Forest** est une solution aux problèmes d'instabilités et de sur-apprentissage. C'est est un algorithme qui, contrairement aux arbres de décisions qui sont **uniques** et optimaux à tout les noeuds, l'algorithme créer une foret d'arbre aléatoires différents issus d'une même base de donnée, le resultat est ensuite la moyenne de la valeur prédite par chacun des arbres.

Thomas MASSE, élève de la promotion MSc Data Management 2020/2022 a produit un excellent travail sur le principe du random forest et son utilisation sur R. Je vous invite, si cet approfondissement des arbres de décision vous interresse, d'aller consulter son travail sur son Github :

"THOMAS-MAS"