Les arbres de décision et les forêts aléatoires

Jean-Michel Marin

Université de Montpellier
Institut Montpelliérain Alexander Grothendieck (IMAG)

HMMA303

Arbres de décision uniques

Méthodes basées sur des arbres

- Nous décrivons ici des méthodes *basées sur des arbres* pour la classification et la régression
- Cela implique de *stratifier* ou *segmenter* l'espace des prédicteurs en un certain nombre de régions simples
- Comme l'ensemble des règles des partitionnement peuvent être résumées par un arbre, ce type d'approches sont connues comme des méthodes à *arbres de décision*

Arbres de décision uniques

Pours et contres

- Les méthodes basées sur des arbres sont simples et utiles pour l'interprétation
- Cependant, elles ne sont pas capables de rivaliser avec les meilleures approches d'apprentissage supervisé en terme de qualité de prédiction
- Nous discuterons donc aussi de *bagging*, *forêts aléatoires (random forests)*, et *boosting* ; ces méthodes développent de nombreux arbres de décision qui sont ensuite *combinés* pour produire une réponse consensus

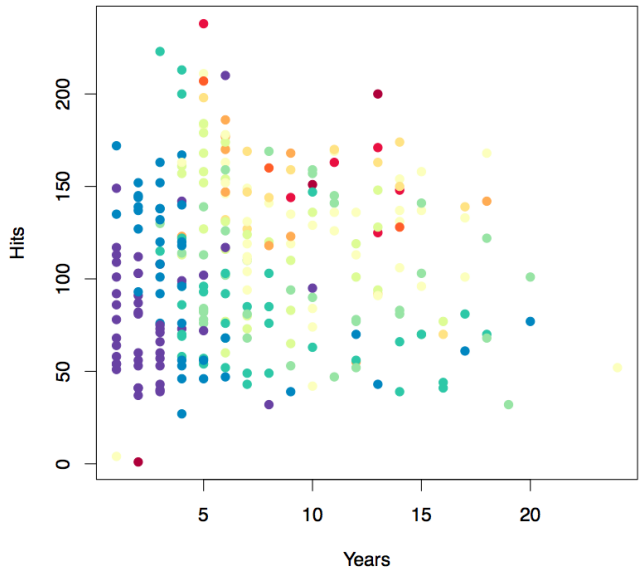
Arbres de décision uniques

Les bases des arbres de décision

- Les arbres de décision sont utiles aussi bien pour des problèmes de régression que de classification
- Nous commençons par présenter des problèmes de régression et nous viendrons ensuite à la classification

Arbres de décision uniques

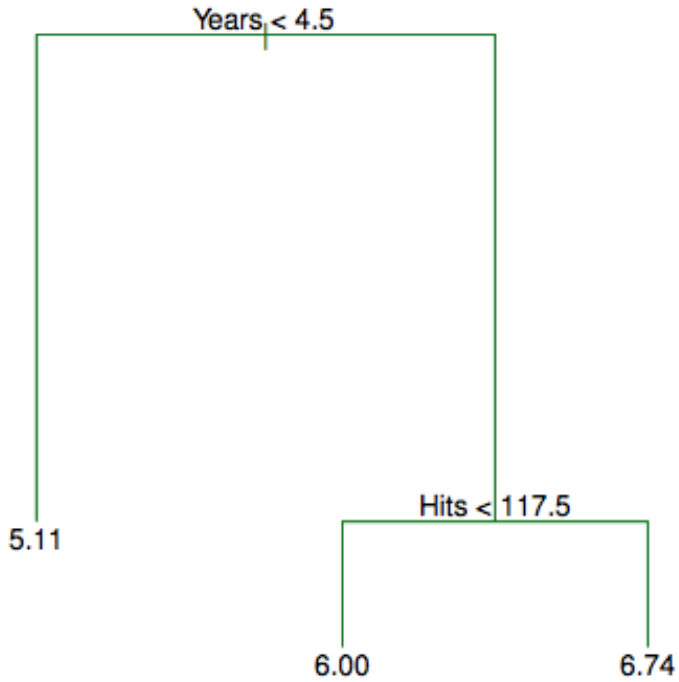
Données de salaire au baseball : comment les stratifier ?



Le salaire est codé par des couleurs : les faibles valeurs sont en bleu, puis vert, les plus fortes valeurs en orange puis rouge

Arbres de décision uniques

Un arbre de décision sur ces données



Arbres de décision uniques

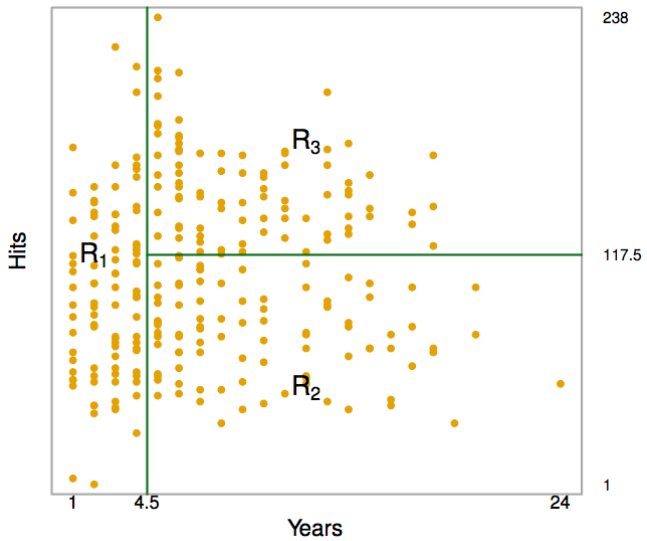
Détails sur la précédente figure

- C'est un arbre de régression pour prédire le **log** des salaires des joueurs, basé sur
  - l'expérience (`Years`)
  - le nombre de succèes (`Hits`)
- Pour chaque nœud interne, l'étiquette (de la forme $X_j < t_k$) indique la branche de gauche émanant du nœud, et la branche droite correspond à $X_j \geqslant t_k$
- Cet arbre a deux nœuds internes et trois nœuds terminaux ou feuilles ; le nœud le plus haut dans la hiérarchie est la racine
- L'étiquette des feuilles est la réponse moyenne des observations qui satisfont aux critères pour la rejoindre

Arbres de décision uniques

Résultats

- En tout, l'arbre distingue trois classes de joueurs en partitionnant l'espace des prédicteurs en trois régions :
  $R_1 = \{X : \texttt{Years} < 4.5\}$,
  $R_2 = \{X : \texttt{Years} \geqslant 4.5, \texttt{Hits} < 117.5\}$ et
  $R_3 = \{X : \texttt{Years} \geqslant 4.5, \texttt{Hits} \geqslant 117.5\}$



Arbres de décision uniques

Interprétation des résultats

- `Years` est le facteur le plus important pour expliquer `Salary` : les joueurs de moindre expérience gagnent moins que les joueurs expérimentés
- Sachant qu'un joueur a peu d'expérience, le nombre de `Hits` l'année passée n'influence pas son salaire
- Mais, parmis les joueurs expérimentés, le nombre de `Hits` de l'année passée affecte son salaire (positivement)
- C'est sûrement une simplification de la réalité, mais comparé à un modèle de régression (linéaire par exemple), la fonction de régression est simple à décrire, interpréter et expliquer

Arbres de décision uniques

Détails sur la construction de l'arbre

Algorithme CART (Classification and Regression Trees)

1. Division de l'espace des prédicteurs en J régions distinctes, non recouvrantes : $R_1, R_2, \ldots R_J$
2. Pour toute nouvelle observation des prédicteurs $X = x_0$, on regarde dans quelle région on tombe, disons R_ℓ ; la prédiction est la moyenne des valeurs observées dans la partie de l'ensemble d'entrainement qui tombent dans R_ℓ

Arbres de décision uniques

- Pour limiter l'espace des partionnements possibles, les arbres de décision divisent l'espace en rectangles ou boîtes parallèles aux axes
- Le but est de trouver les boîtes $R_1, \ldots, R_J$ qui minimisent un critère des moindres carrés, ici

$$SSE = \sum_{j=1}^{J} \sum_{i:x_i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2,$$

où $\hat{y}_{R_j}$ est la réponse moyenne sur les observations d'entrainement qui tombent dans R_j

Arbres de décision uniques

- Malheureusement, il est impossible de traverser l'ensemble des partitionnements de l'espace des prédicteurs en J boîtes
- Pour cette raison, on met en place un algorithme *glouton*, *top-down* qui construit l'arbre binaire de façon récursive.
- L'algorithme démarre à la racine de l'arbre et sépare ensuite l'espace des prédicteurs en ajoutant progressivement des nœuds
- On parle d'algorithme *glouton* car à chaque étape de la construction de l'arbre, on construit la meilleur division possible du nœ ud en deux sous-nœuds

Arbres de décision uniques

L'algorithme de construction de l'arbre $\mathscr{T}_0$ (phase 1)

Initialisation on place tout l'échantillon d'apprentissage à la racine de l'arbre

Récurrence sur chaque noeud on partitionne chaque nœud en deux classes

$$\mathscr{R}_1(j,s) = \{X : X_j \leqslant s\}, \quad \mathscr{R}_2(j,s) = \{X : X_j > s\}$$

en cherchant j et s qui minimisent

$$\sum_{x_i \in \mathscr{R}_1(j,s)} \left(y_i - \hat{c}_1\right)^2 + \sum_{x_i \in \mathscr{R}_2(j,s)} \left(y_i - \hat{c}_2\right)^2$$

où $\hat{c}_m = \text{ave}\left(y_i \middle| x_i \in \mathscr{R}_m(j,s)\right)$

Arbres de décision uniques

Phase 1 Construction de $\mathscr{T}_0$

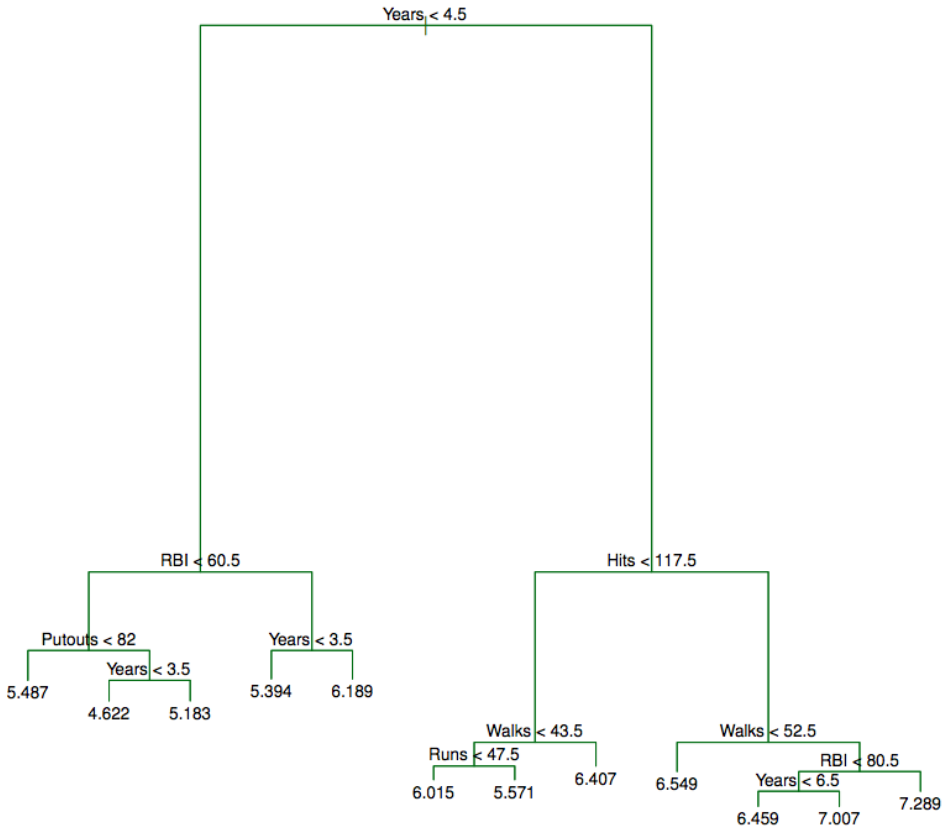
Terminaison On arrête de diviser un noeud de $\mathscr{T}_0$ lorsqu'il y a peu d'observations (disons 5)

Problème de sur-apprentissage
- un arbre trop profond sur-apprend
- un arbre trop peu profond n'est pas assez précis

$\Longrightarrow$ Élagage de $\mathscr{T}_0$

Arbres de décision uniques

Baseball : l'arbre $\mathscr{T}_0$



Arbres de décision uniques

Sur-apprentissage
L'arbre $\mathscr{T}_0$ obtenu est trop profond. Faire un compromis entre

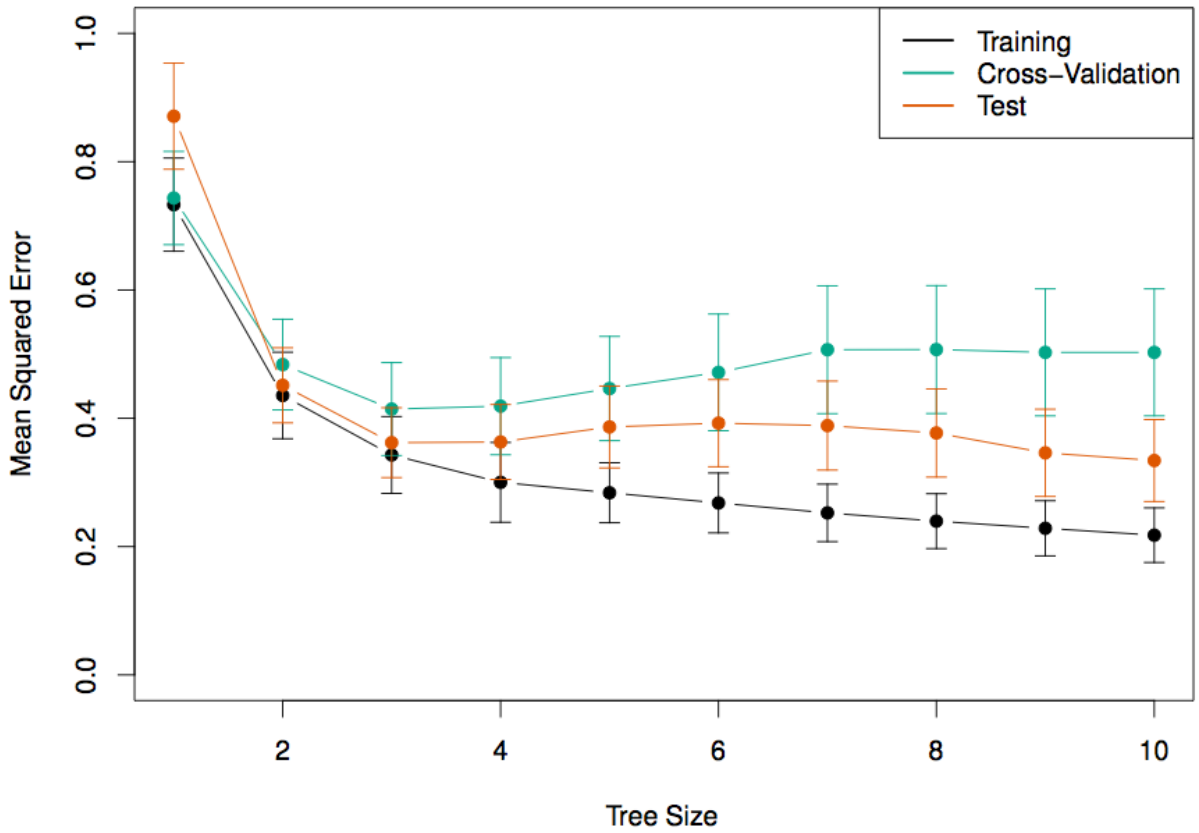
- sur-apprentissage (trop profond)
- erreur de prédiction trop grande (trop court)

Introduire un paramètre α qui règle le compromis, et minimiser le critère pénalisé défini pour $T \subset \mathscr{T}_0$ par

$$\mathscr{C}_\alpha(T) := \sum_{m=1}^{|T|} N_m(T) Q_m(T) + \alpha|T|,$$

où $|T|$ = nombre de feuilles de T ; $N_m(T) = \text{Card}\{x_i \in \mathscr{R}_m(T)\}$;
$Q_m(T) = \frac{1}{N_m(T)} \sum_{x_i \in \mathscr{R}_m(T)} (y_i - \hat{c}_m(T))^2$; $\hat{c}_m(T) = \text{ave}\left(y_i \middle| x_i \in \mathscr{R}_m(T)\right)$

Arbres de décision uniques



Arbres de décision uniques

Phase 2 : Élagage

Calcul des minima $\mathscr{T}_\alpha$ du critère pénalisé :
On part de $\mathscr{T}_0$ et on construit une suite d'arbres itérativement.
À chaque étape, on supprime le noeud interne de tel sorte à produire la plus petite augmentation de

$$\sum_m N_m(T) Q_m(T)$$

et l'on s'arrête lorsque l'arbre est réduit à un seul noeud (racine)

Tous les minima $T = \mathscr{T}_\alpha$ des fonctions $T \mapsto \mathscr{C}_\alpha(T)$ sont dans cette suite
Calibration de α par validation croisée à 5 ou 10 blocs

Arbres de décision uniques

Arbres de classification

- Similaires aux arbres de régression, sauf qu'ils sont utilisés pour prédire une réponse discrètes
- Pour un arbre de classification, on prédit à l'aide la classe la plus fréquente dans cette feuille parmi les données d'entrainement

Arbres de décision uniques

Il faut changer le critère des moindres carrés en un critère plus adapté, lié à la mesure des erreurs de classification

Si la feuille m représente la région $\mathscr{R}_m$ avec N_m observations, on définit

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in \mathscr{R}_m} \mathbf{1}\{y_i = k\},$$

la proportion d'observation du noeud m appartenant à la k^e classe

Alors la valeurs de $\hat{f}$ sur $\mathscr{R}_m$ est $\hat{c}_m = \text{argmax}_k \, \hat{p}_{mk}$ (vote à la majorité simple)

Arbres de décision uniques

Les différentes mesures d'impureté $Q_m(T)$ d'une feuille m sont

- **erreur de classification** : $\frac{1}{N_m} \sum_{x_i \in \mathcal{R}_m} \mathbf{1}\{y_i \neq \hat{c}_m\} = 1 - \hat{p}_{m\hat{c}_m}$
- **Index de Gini** : $\sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_k \hat{p}_{mk}(1 - \hat{p}_{mk})$
- **Entropie** : $-\sum_k \hat{p}_{mk} \log \hat{p}_{mk}$.

Le critère que l'on minimise pour construire $\mathcal{T}_0$ est

$$\sum_{m=1}^{|T|} N_m(T) Q_m(T)$$

Agrégation par bagging

- L'agrégation bootstrap ou bagging est méthode de réduction de la variance en apprentissage statistique ; elle est particulièrement utile sur les arbres de décision
- Rappelons que, sur un ensemble de n observations indépendantes $Z_1, \ldots, Z_n$, chacune de variance σ^2, la variance de la moyenne $\bar{Z}$ est σ^2/n
- En pratique, il n'est pas possible de moyenner des arbres de décision construits sur de multiples ensembles d'entrainement (pas assez de données observées)

Agrégation par bagging

- Au lieu de cela, on peut bootstrapper en ré-échantillonnant plusieurs fois les données d'entrainement
- Alors, à partir de B échantillons bootstrap, on entraine une méthode d'apprentissage pour ajuster B fonctions de régressions, notées $\hat{f}^{*b}(x)$, $b = 1, \ldots, B$
- La fonction de régression bagguée est alors

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Agrégation par bagging

- Sur un problème de classification, $\hat{f}^{*b}(x)$ renvoie une classe possible pour chaque échantillon bootstrap b
- La décision finale $\hat{f}_{bag}(x)$ se prend par un vote à la majorité simple parmi les B prédictions des classifieurs bootstrap
- Sur des gros jeux de données d'entrainement, faire parfois du sous-échantillonnage bootstrap

Agrégation par bagging

Erreur out-of-bag

- Il y a une façon simple d'estimer l'erreur de test quand on fait du bagging
- La clé du bagging est l'entrainement de nombreux $\hat{f}(x)$ sur des échantillons bootstraps. On peut donc utiliser les observations hors du b^e bootstrap pour évaluer chaque $\hat{f}^{*b}(x)$

Agrégation par bagging

1. Pour chaque observation (x_i, y_i), calculer $\hat{y}_i^{OOB}$ la prédiction en n'utilisant que les estimateurs $\hat{f}^{*b}(x)$ qui n'ont pas vu cette observation dans leur entrainement
2. Évaluer l'erreur entre $\hat{y}_i^{OOB}$ et les y_i (erreur quadratique moyenne ou taux de mauvaise classification)

Forêts aléatoires

- Les *forêts aléatoires* améliorent le bagging des arbres CART en dé-corrélant ces arbres ; ce qui permet de mieux réduire la variance
- Comme précédemment, on construit une suite d'arbres de décisions sur des échantillons bootstraps
- Au lieu d'utiliser l'algorithme CART, on utilise un algorithme randomisé
- Au moment de diviser un nœud, cet algorithme ne regarde que m prédicteurs parmi les $\mathrm{p} > \mathrm{m}$ prédicteurs, tirés au hasard
- Une nouvelle sélection de prédicteurs est faite à chaque division de nœud
- Il est inutile d'élaguer les arbres de décision, le bagging se charge d'éviter les erreurs de sur-apprentissage

Forêts aléatoires

Paramètres de réglage

- Il faut choisir
  - le nombre de prédicteurs m (par défaut $\sqrt{\mathrm{p}}$ en régression et $\mathrm{p}/3$ en classification) tirés à chaque division de noeud
  - le nombre total d'arbres
  - la taille des sous-échantillons bootstraps si gros jeu de données
- On peut s'appuyer sur l'erreur *out-of-bag*

Résumé

- Les arbres de décision sont des modèles simples et interprétables
- Cependant, ils produisent souvent de mauvais résultats comparés à d'autres méthodes
- La méthodes des forêts aléatoires (Random Forest) est une bonne méthode pour améliorer la qualité de la prédiction des arbres de décision
- Les forêts aléatoires font parmi de l'état de l'art actuel des méthodes d'apprentissage supervisé ; cependant, le classifieur ou la fonction de régression produite est difficile à interpréter