Arbres de décision

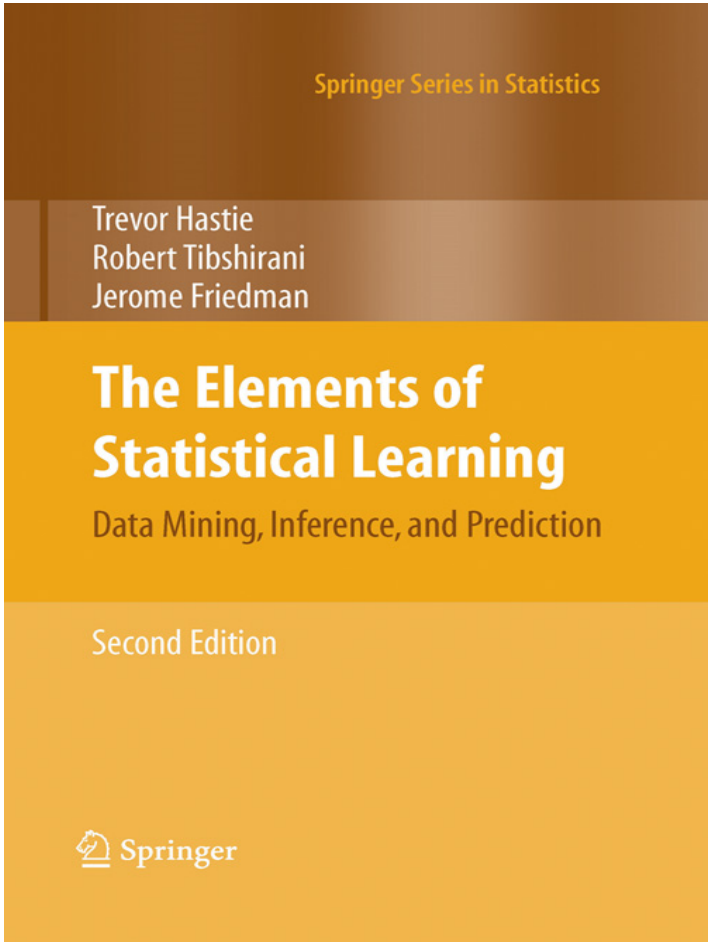
masedki.github.io

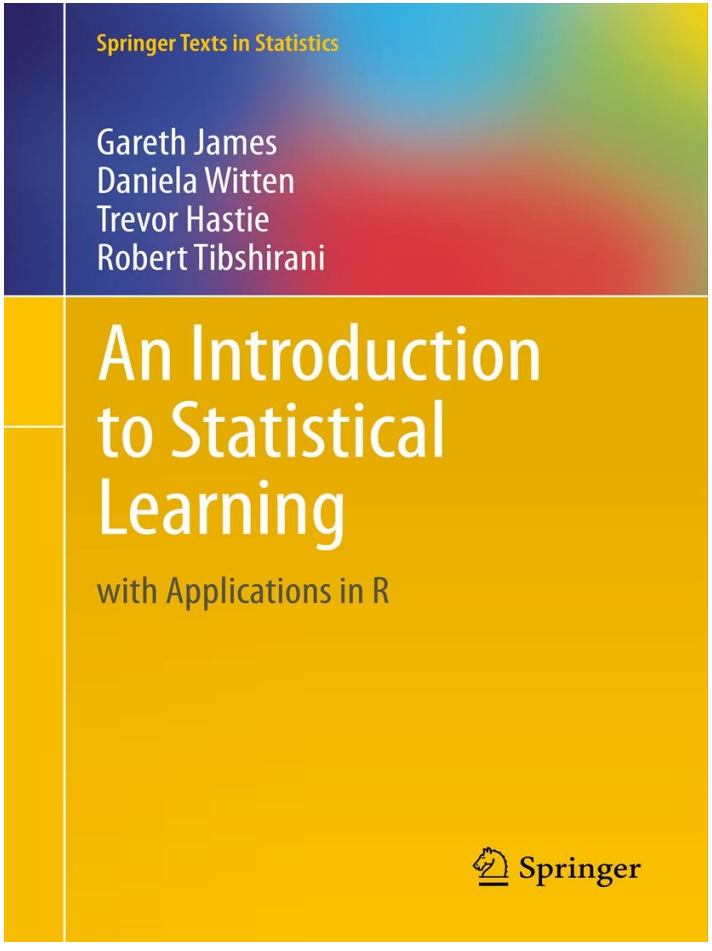
mar. 21 mai 2019

Outline

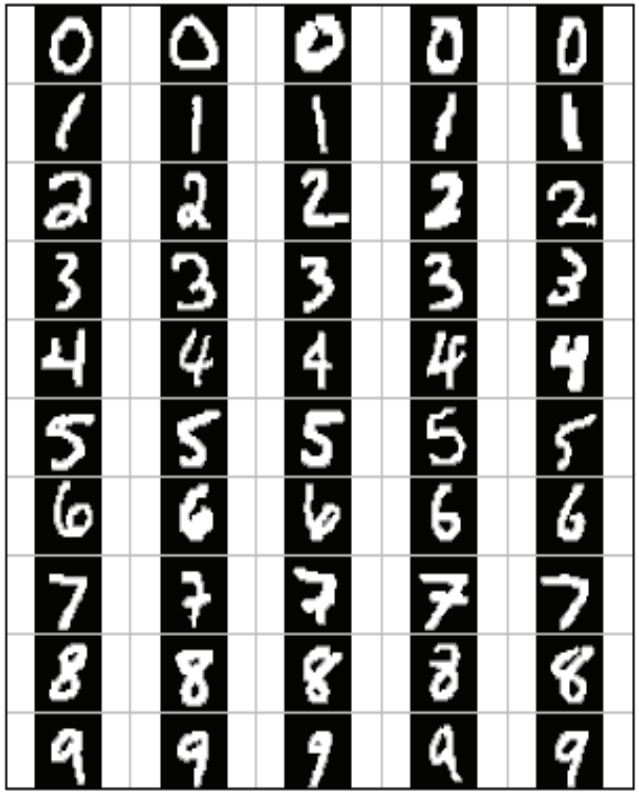
1. **Introduction**
2. **Régression vs classification supervisée**
3. **Arbres de décision uniques**

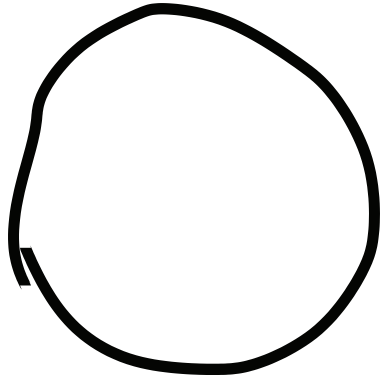
Références


Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer


Springer Texts in Statistics
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani
An Introduction to Statistical Learning
with Applications in R
Springer

Problème d'apprentissage





- Reconnaissance de chiffres manuscrits ?

0, 1, 2 ... ?

Problèmes d’apprentissage statistique

1. Identifier les facteurs de risque du cancer de la prostate
2. Prédire si une personne est sujette aux crises cardiaques, à partir de mesures cliniques, son régime et des données démographiques
3. Classification d’échantillons de tissus dans différents types de cancer, en fonction de données d’expression de gènes
4. Classer des images de tumeurs

Question

- Sur 4 601 mails, on a pu identifier 1813 spams.
- On a également mesuré sur chacun de ces mails la présence ou absence de 57 mots.

Peut-on construire à partir de ces données une méthode de détection automatique de spam ?

Outline

Réprésentation du problème

- La plupart de ces problèmes peuvent être appréhendés dans un contexte de régression : on cherche à expliquer une variable Y par d'autres variables dites explicatives $X_1, \ldots, X_p$:

Y	X
Chiffre	image
Mot	courbe
Spam ou pas	présence/absence d'un ensemble mots
Type de leucémie	expressions de gênes

- Lorsque la variable à expliquer est quantitative, on parle de régression.
- Lorsqu'elle est qualitative, on parle de discrimination ou classification supervisée.

Régression

- Un échantillon i.i.d $(X_1, Y_1), \ldots, (X_n, Y_n)$ à valeurs dans $\mathbb{R}^p \times \mathbb{R}$.
- Objectif : Prédire ou expliquer la variable Y à partir d'une nouvelle observation X.
- Méthode : construire un régresseur

$$m : \mathbb{R}^p \mapsto \mathbb{R}.$$

- Critère de performance pour m : l'erreur de prévision

$$\mathbb{E}\left[\left(Y - m(X)\right)^2\right].$$

La fonction de régression

- Un champion

$$m^*(x) = \mathbb{E}\big[Y|X = x\big]$$

  appelé fonction de régression.

- Pour toute autre fonction m, on a

$$\mathbb{E}\left[\big(Y - m^*(X)\big)^2\right] \leq \mathbb{E}\left[\big(Y - m(X)\big)^2\right]$$

.

- Problème: m^* est inconnu en pratique. Il faut construire un régresseur $\widehat{m}_n$ à partir des données $(X_1, Y_1), \ldots, (X_n, Y_n)$, tel que

$$\widehat{m}_n(x) \approx m^*(x).$$

La classification binaire

- Un échantillon i.i.d $(X_1, Y_1), \ldots, (X_n, Y_n)$ à valeurs dans $\mathbb{R}^p \times \{0, 1\}$.
- Objectif : Prédire ou expliquer la variable Y à partir d'une nouvelle observation X.
- Méthode : construire une règle classification

$$g : \mathbb{R}^p \mapsto \{0, 1\}.$$

- Critère de performance pour g : probabilité d'erreur ou de mauvais classement

$$L(g) = \mathbb{P}\big(g(X) \neq Y\big).$$

La règle de Bayes

- Un autre champion :

$$g^*(x) = \begin{cases} 0 & \text{si} \quad \mathbb{P}(Y = 0 | X = x) \geq \mathbb{P}(Y = 1 | X = x) \\ 1 & \text{sinon,} \end{cases}$$

appelé règle de Bayes.

- Quelque soit la règle de décision g, nous avons

$$L^* = L(g^*) = \mathbb{P}(g^*(X) \neq Y) \leq \mathbb{P}(g(X) \neq Y) = L(g).$$

- Problème: g^* est inconnue en pratique. Il faut construire une règle $\widehat{g}_n$ à partir des données $(X_1, Y_1), \ldots, (X_n, Y_n)$, tel que

$$\widehat{g}_n(x) \approx g^*(x).$$

Notations

- On s'intéresse au cas où on cherche à expliquer une variable qualitative Y par p variables explicatives $X_1, \ldots, X_p$.
- Y est à valeurs dans un ensemble discret fini de modalités qui peuvent être numérotées par des les indices $\{1, 2, \ldots, K\}$ et les variables $X_1, \ldots, X_p$ peuvent être qualitatives et/ou quantitatives.
- Néanmoins, pour présenter les méthodes, on se restreint au cas où Y est à 2 modalités (0 et 1).

Outline

1. Introduction

2. Régression vs classification supervisée

3. **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 les règles des partitionnement peuvent être résumées par un arbre, ce type d'approches sont connues comme des méthodes à *arbres de décision*.

Pours et contres

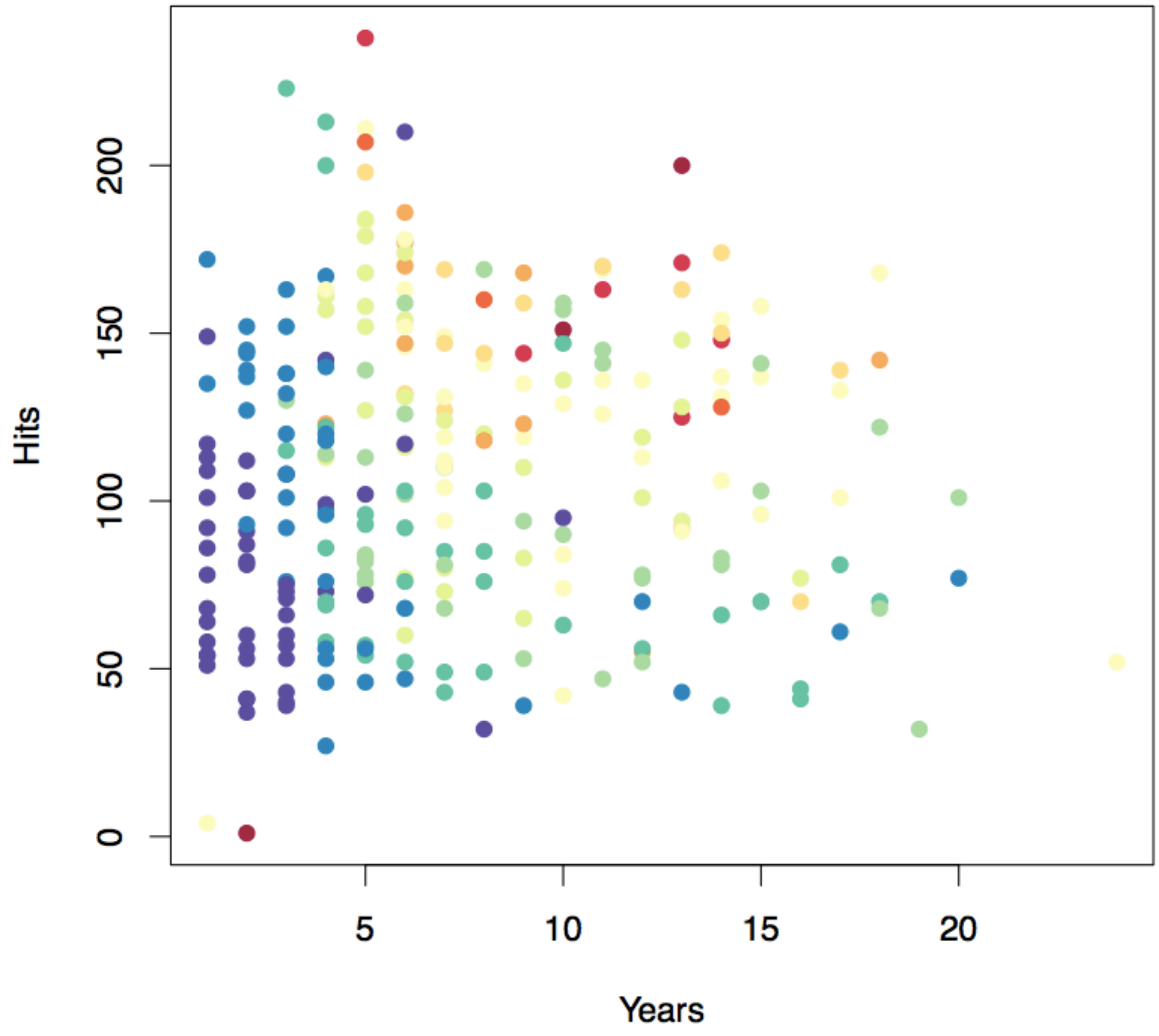
- Les méthodes basées sur des arbres sont simples et utiles pour l'interprétation.
- Cependant, elles ne sont pas capables de rivaliser avec les meilleures approches d'apprentissage supervisé en terme de qualité de prédiction.
- Nous discuterons aussi (dans l'avenir en M2) de *bagging*, *forêts aléatoires (random forests)*, et *boosting*. Ces méthodes développent de nombreux arbres de décision qui sont ensuite *combinés* pour produire une réponse consensus.

Les bases des arbres de décision

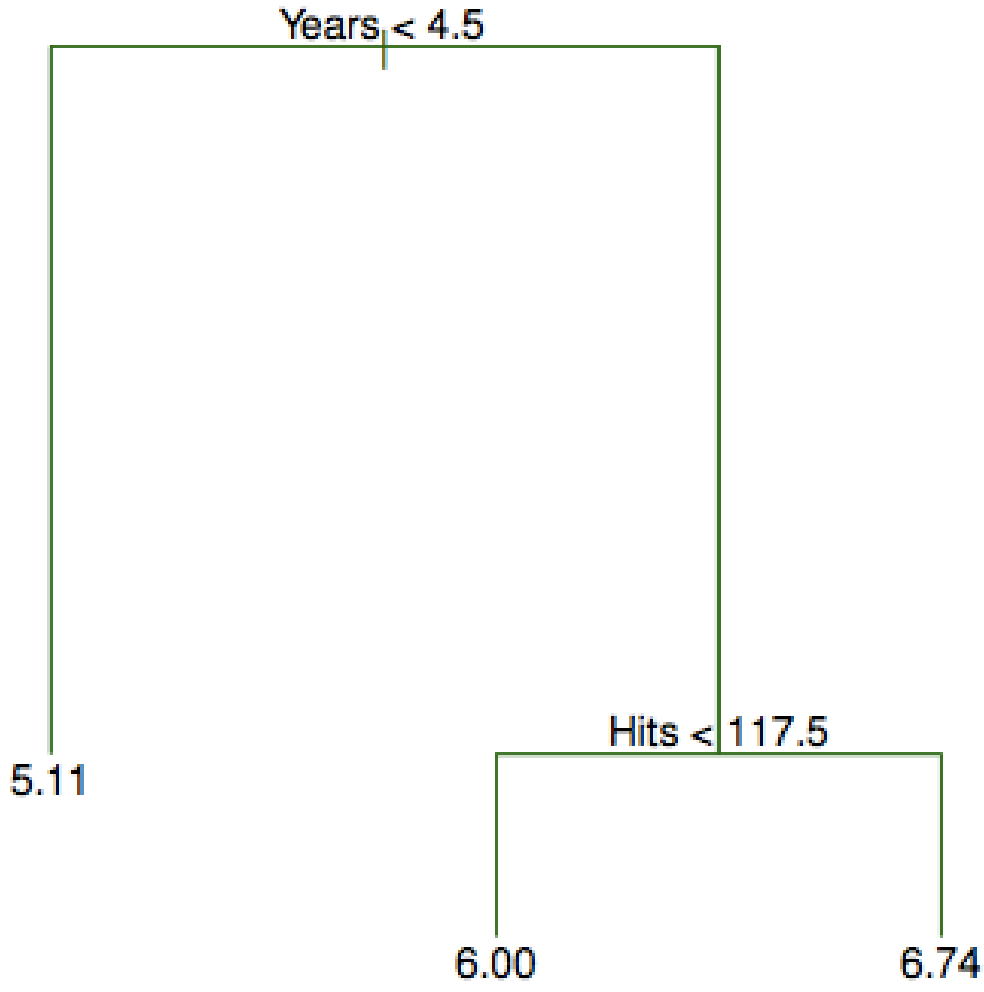
- Les arbres de décision sont utiles aussi bien pour des problèmes de régression que de classification.
- Nous commençons par présenter des problèmes de régression et nous viendrons ensuite à la classification.

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.



L’arbre de décision sur ces données

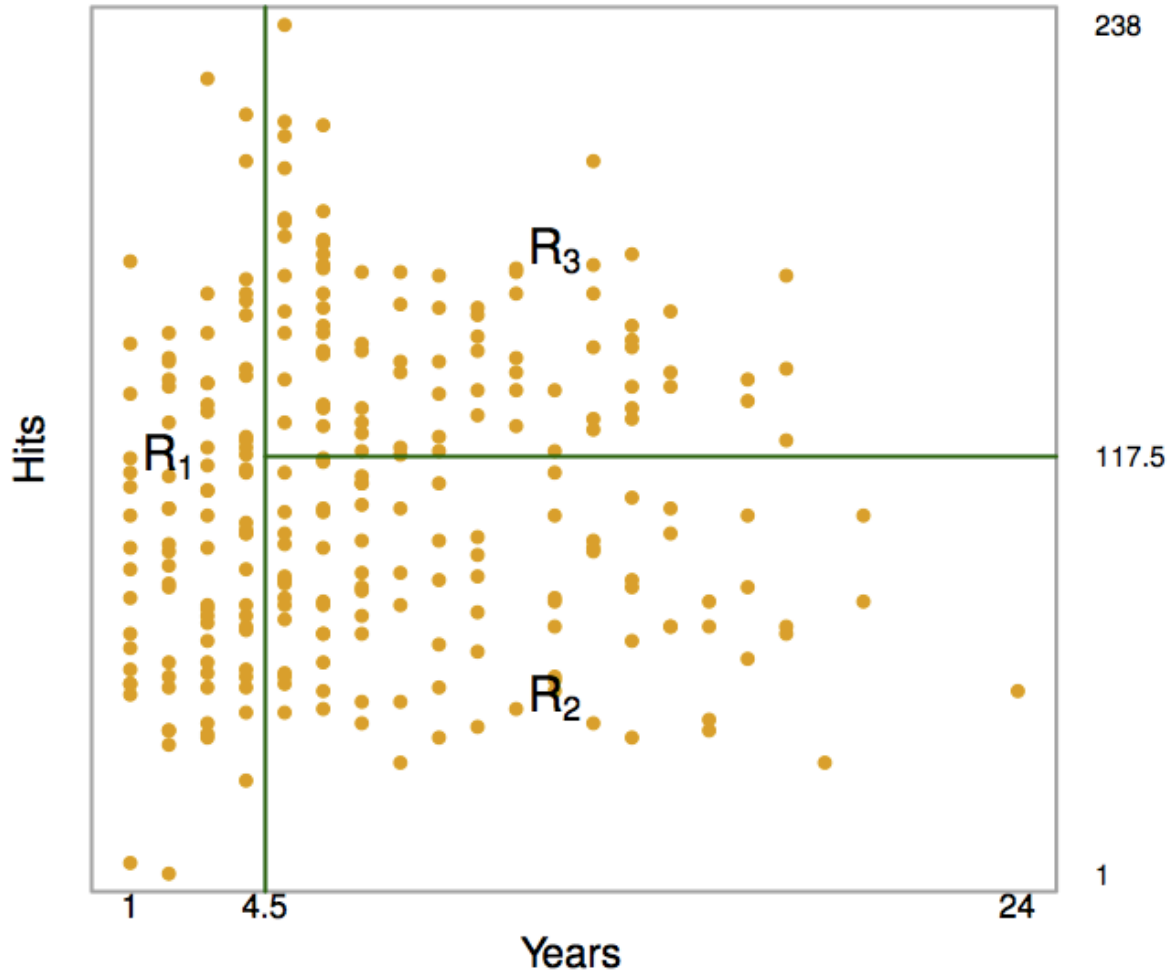


Détails de la précédente figure

- C'est un arbre de régression pour prédire le **log** des salaires des joueurs, basé sur
  - l'expérience (`Years`)
  - le nombre de succès (`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 $X_j \geq t_k$.
- 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.

Résultats

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



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.

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.

Détails sur la construction de l'arbre (suite)

- Pour limiter l'espace des partitions 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

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

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

Détails sur la construction de l'arbre (suite)

- 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 nœud en deux sous-nœuds.

L'algorithme de construction de l'arbre T_0 (phase 1)

Initialisation

Nœud racine : on place l'ensemble de l'échantillon d'estimation à la racine de l'arbre

Récurrence sur chaque nœud

On partionne chaque nœud en deux classes:

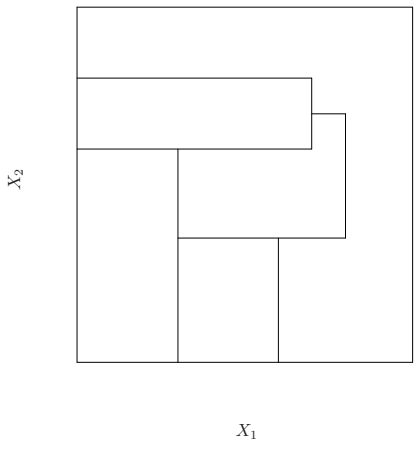
$$\mathcal{R}_1(j,s) = \{X : X_j \leq s\}, \quad \mathcal{R}_2(j,s) = \{X : X_j > s\}$$

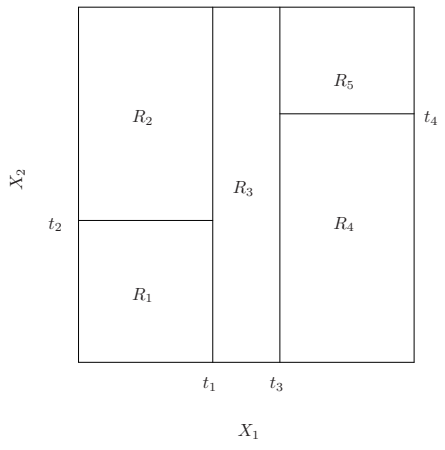
en cherchant j et s qui minimisent

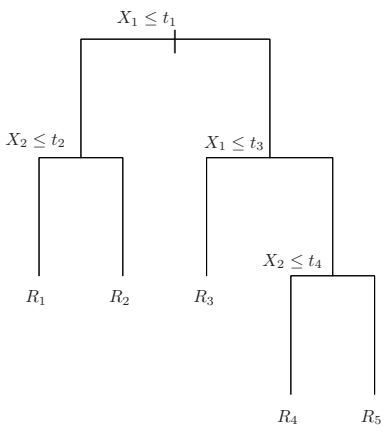
$$\text{RSS}_{\text{new}} = \sum_{i:x_i \in \mathcal{R}_1(j,s)} \left(y_i - \hat{y}_1\right)^2 + \sum_{i:x_i \in \mathcal{R}_2(j,s)} \left(y_i - \hat{y}_2\right)^2 \tag{1}$$

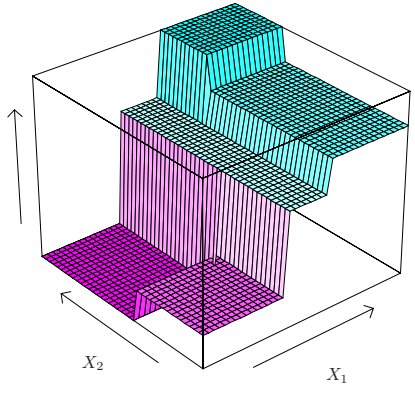
où $\hat{y}_m = \text{ave}\left(y_i \middle| x_i \in \mathcal{R}_m(j,s)\right)$ est la réponse moyenne des données d'apprentissage qui tombent dans la région $R_m(j,s)$ pour $m = 1$ ou $m = 2$.
Trouver le couple (j,s) optimal est un problème relativement facile lorsque le nombre de variables p n'est pas trop grand.

Exemples de récurrence binaire









Exemples de récurrence binaire

- En haut à gauche : exemple de partition qui ne peut être le résultat d'une partition binaire
- En haut à droite : résultat d'une partition binaire récursive
- En bas à gauche : l'arbre binaire correspondant à la partition en haut à droite
- En bas à droite : surface de prédiction associé à cet arbre

Algorithme (suite…)

Phase 1 : Construction de T_0

Initialisation

[…]

Récurrence sur chaque nœud

[…]

Terminaison

On arrête de diviser un nœud de T_0 lorsqu'il y a peu d'observations (disons 5).

Critère d'arrêt

- La récurrence jusqu'à 5 observations par noeud terminal est arbitraire
- Trop d'étapes de partitionnement : beaucoup de feuilles (noeuds terminaux), modèle trop complexe, petit biais mais grande variance, sur-apprentissage
- Peu d'étapes de partitionnement: peu de feuilles, modèle trop simple, grand biais mais petite variance, sous-apprentissage

Une première idée

- Division d'une région R en deux régions R_1 et R_2 on considère la somme des carrés des résidus avant la division

$$\text{RSS}_{\text{old}} = \sum_{i \in R} \left(y_i - \hat{y}\right)^2$$

où $\hat{y}$ est la moyenne de la variable réponse des données qui se situent dans la région R. Avec la division optimale, la réduction de RSS

$$\text{RSS}_{\text{old}} - \text{RSS}_{\text{new}}$$

- On peut choisir un seuil h et décider de la significativité d'une partition
- Si la réduction du RSS est supérieure à h on applique le *split* sinon on arrête

Une première idée (suite)

- L'idée est raisonnable mais trop *locale*
- Une partition peut être jugée non-significative et peut cacher d'autres partitions plus significatives

Sur-apprentissage

L'arbre T_0 obtenu est trop profond. Faire un compromis entre

- sur-apprentissage : trop profond
- arbre trop peu précis (grande erreur de prédiction): trop peu profond

Solution : élagage de T_0 appelé *Cost complexity pruning*

Élagage

Une stratégie consiste à construire un très grand arbre, puis à l'élaguer afin d'obtenir un sous-arbre.

- Comment détermine-t-on le meilleur moyen d'élaguer l'arbre ?
- Sélectionner un sous-arbre menant à l'erreur de test la plus faible.
- Nous pouvons estimer l'erreur de test en utilisant la validation croisée (chaque sous-arbre : explosion combinatoire !!).
- Sélectionner un petit ensemble de sous-arbres à prendre en compte.
- L'élagage du maillon le plus faible permet de considérer une séquence d'arbres indexés par un paramètre de réglage non négatif α.

Élagage : détails

Introduire un paramètre α qui règle le compromis, et minimiser le critère pénalisé *perte + pénalité* défini pour $T \subset T_0$ par

$$\mathcal{C}_\alpha(T) := \sum_{m=1}^{|T|} N_m(T) Q_m(T) + \alpha|T| = \sum_{m=1}^{|T|} \sum_{x_i \in R_m(T)} \left(y_i - \hat{y}_m\right)^2 + \alpha|T|,$$

où

- $\left|T\right|$ est nombre de feuilles de T
- $N_m(T) = \text{Card}\left\{x_i \in \mathcal{R}_m(T)\right\}$ et $Q_m(T) = \frac{1}{N_m(T)} \sum_{x_i \in \mathcal{R}_m(T)} \left(y_i - \hat{y}_m\right)^2$
- $\hat{y}_m = \text{ave}\left(y_i \middle| x_i \in \mathcal{R}_m(T)\right)$
- On notera T_α le sous-arbre qui minimise $C_\alpha(T)$ à α fixé
- Rôle de α ? Cas particuliers $\alpha = 0$ et $\alpha \to +\infty$!!

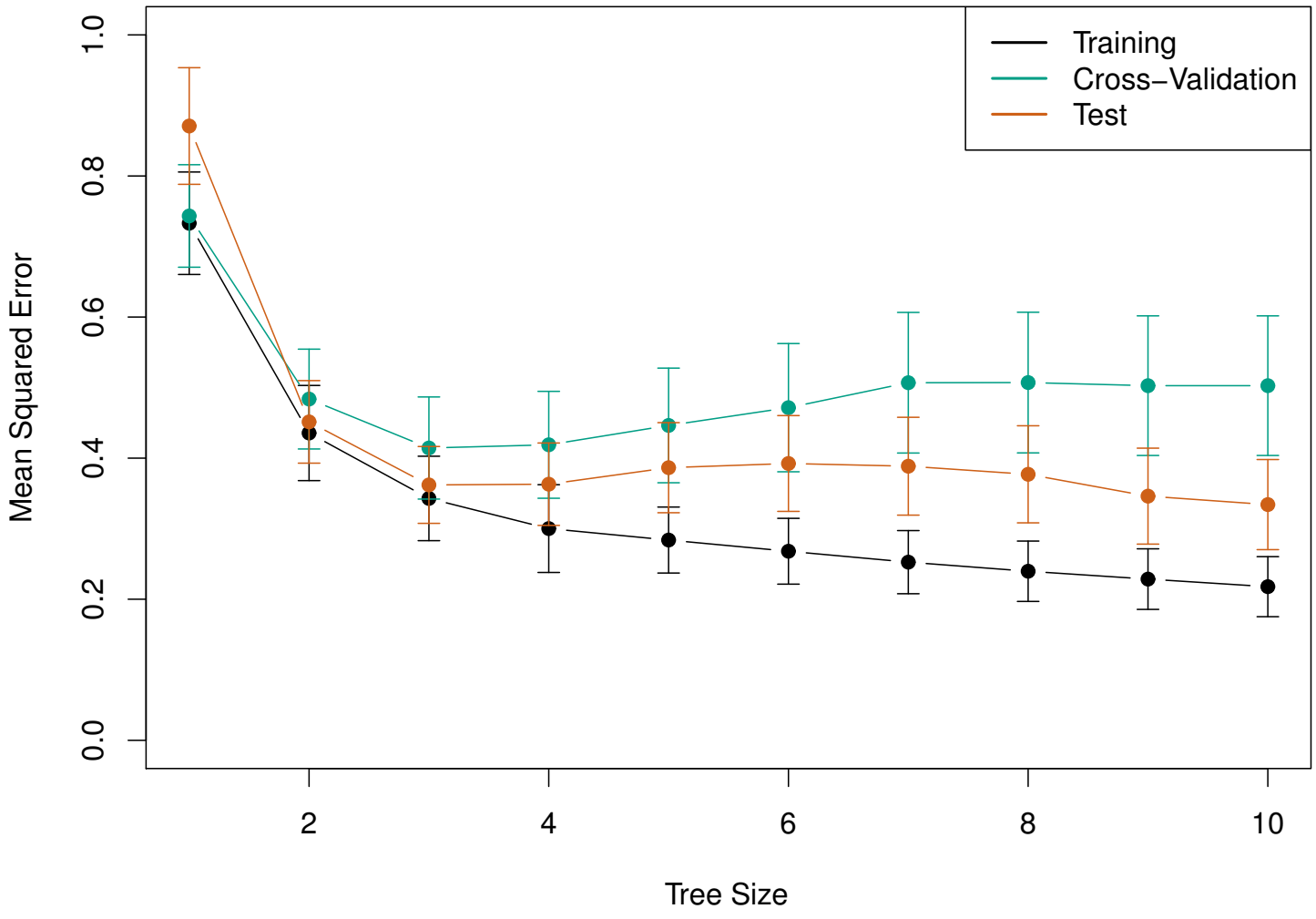
Élagage : Calcul des minima T_α du critère pénalisé

1. On construit une suite d'arbres itérativement
   - On part de T_0
   - À chaque étape, on supprime le nœud interne de tel sorte à produire la plus petite augmentation de
   $$\sum_m N_m(T) Q_m(T)$$
   - On s'arrête lorsque l'arbre est réduit à un seul nœud (racine)
2. Tous les minima $T = T_\alpha$ des fonctions $T \mapsto \mathcal{C}_\alpha(T)$ sont dans cette suite

Élagage : Choix de α par validation croisée K-folds

- Diviser le jeu de données d'apprentissage en K-folds
- Pour $k = 1, \ldots, K$:
  - Calculer les minima T_α du critère pénalisé sur l'ensemble du jeu de données privé du $k^{\text{ième}}$ fold
  - Pour chaque T_α, calculer l'erreur de prédiction moyenne des données du $k^{\text{ième}}$ fold comme une fonction $\text{err}_{-k}(\alpha)$ de α
- Choisir la valeur de α^* qui minimise la fonction moyenne $\frac{1}{K}\sum_{k=1}^{K} \text{err}_{-k}(\alpha)$
- Renvoyer T_{α^*} calculé par élagage sur l'ensemble du jeu de données d'apprentissage

Illustration : *Hitters* dataset



Exemple : coût de soins

- Compétition kaggle https://www.kaggle.com/mirichoi0218/insurance
- À l'aide de la fonction rpart, ajuster un arbre de décision sans élagage pour prédire la variable medv en fonction des autres variables présentes dans le jeu de données.
    - Utiliser la fonction rpart.control pour construire un arbre en continuant les découpages dans les feuilles qui contiennent au moins 5 observations (paramètre minsplit=5) et sans contrainte sur la qualité du découpage (paramètre cp=0)
    - Visualiser l'arbre obtenu à l'aide de la fonction rpart.plot
    - Évaluer l'erreur de prédiction du modèle sur le jeu de données test
- Découvrir l'élagage effectué automatiquement à l'aide de la fonction plotcp
- À l'aide de la fonction prune, extraire l'arbre obtenu par élagage correspondant à l'erreur minimale par validation croisée
- Tracer le nouvel arbre obtenu par élagage et évaluer son erreur de prédiction sur le jeu de données test

Arbres de classification

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

Classification : différence avec la régression

- Rappelons qu'en régression, on vise à réduire les moindres carrés (ou somme des carrés des résidus) notés RSS qui sert à mesurer l'erreur du modèle
- En classification, on a besoin d'une d'une mesure d'erreur appropriée
- Réponse catégorielle $Y \in \{1, 2, \ldots, K\}$ donc la prédiction $\hat{f}(x) \in \{1, 2, \ldots, K\}$

Taux d'erreur pour la classification

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

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

  la proportion d'observations du nœud m appartenant à la $k^{\text{ième}}$ classe.

- On assigne une nouvelle observation dans la région $\mathcal{R}_m$ à la classe $\hat{c}_m = \operatorname{argmax}_k \hat{p}_{mk}$ (vote à la majorité simple)

Mesures *d'impureté*

En classification, les différentes mesures d'impureté $Q_m(T)$ d'une feuille m sont

- **Taux de mauvais classement** :

$$\frac{1}{N_m} \sum_{x_i \in \mathcal{R}_m} \mathbf{1}\{y_i \neq \hat{c}_m\} = 1 - \hat{p}_{m\hat{c}_m}$$

- **Indice de Gini** :

$$\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_k \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- **Entropie** :

$$-\sum_k \hat{p}_{mk} \ln \hat{p}_{mk}$$

Mesures *d'impureté*

- Si $\mathcal{R}_m$ est presque *pure*, la plupart des observations proviennent d'une seule classe, alors l'indice de Gini et l'entropie prendraient des valeurs plus petites que le taux de mauvais classement
- L'indice de Gini et l'enropie sont plus sensibles à la pureté des nœuds
- Pour évaluer la qualité d'une partition, l'indice de Gini et l'entropie sont souvent utilisés comme mesure d'erreur (plus que le taux de mauvais classement)
- Chacune de ces trois mesures peut être utilisée lors de l'élagage d'un arbre
- Le taux de mauvais classement est préférable si on vise une meilleure précision de prédiction de l'arbre élagué final

Jeu de données Pima

```
rm(list=ls())
require(rpart)
require(rpart.plot)
require(MASS)
data(”Pima.tr”)
data(”Pima.te”)
```

- Reprendre les étapes de l’exemple de régression pour ajus ter un arbre de décision visant à prédire `type` (diabète : Yes or No) en fonction des autres variables présentes dans le jeu de données..

Avantages et inconvénients des arbres

- ▲ Les arbres sont faciles à expliquer à n'importe qui. Ils sont plus faciles à expliquer que les modèles linéaires
- ▲ Les arbres peuvent être représentés graphiquement, et sont interprétables même par des non-experts
- ▲ Ils peuvent gérer des variables explicatives catégorielles sans introduire des variables binaires
- ▼ Malheureusement, ils n'ont pas la même qualité prédictives que les autres approches d'apprentissage.

Cependant, en agrégeant plusieurs arbres de décision, les performances prédictives s'améliorent substantiellement.

Table of Contents