

# MDI 343 : Arbres pour l'apprentissage

**Joseph Salmon**

<http://josephsalmon.eu>

Télécom Paristech

# Plan

## Introduction

- Rappels de classification

- Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

- Structure efficace : les arbres

- Séparateurs élémentaires

- Algorithme efficace

## Détails et variations

- Fonction de coût

- Fonction d'impureté

- Critères d'arrêt et variantes

- Sélection de modèle

# Sommaire

## Introduction

- Rappels de classification

- Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

- Structure efficace : les arbres

- Séparateurs élémentaires

- Algorithme efficace

## Détails et variations

- Fonction de coût

- Fonction d'impureté

- Critères d'arrêt et variantes

- Sélection de modèle

# Classification supervisée et régression

$X$  : variable explicative, vecteur aléatoire dans  $\mathcal{X} = \mathbb{R}^p$

$Y$  : variable à prédire, aléatoire dans  $\mathcal{Y} = \{C_1, \dots, C_K\}$

(classification avec  $K$  classes) ou  $\mathcal{Y} = \mathbb{R}$  (régression)

$P$  : loi de probabilité jointe de  $(X, Y)$ , fixée mais **inconnue**

$\mathcal{D}_n = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, \dots, n\}$  :  $n$ -échantillon *i.i.d.* tiré selon la loi  $P$

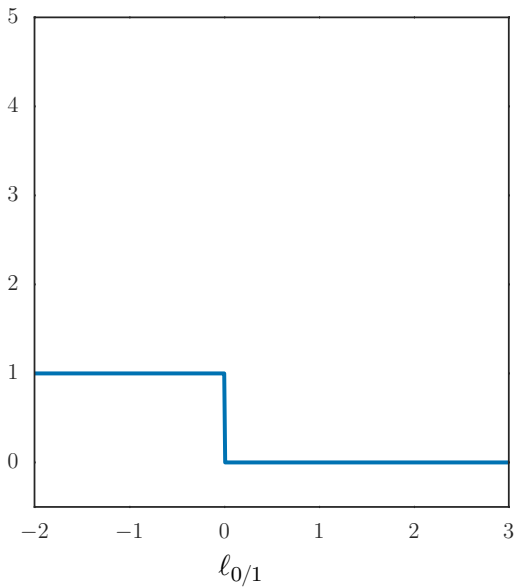
$\mathcal{H}$  : collection de classifieurs/estimateurs,  $h \in \mathcal{H}$

$\ell$  : perte mesurant les erreurs d'un classifieur/estimateur

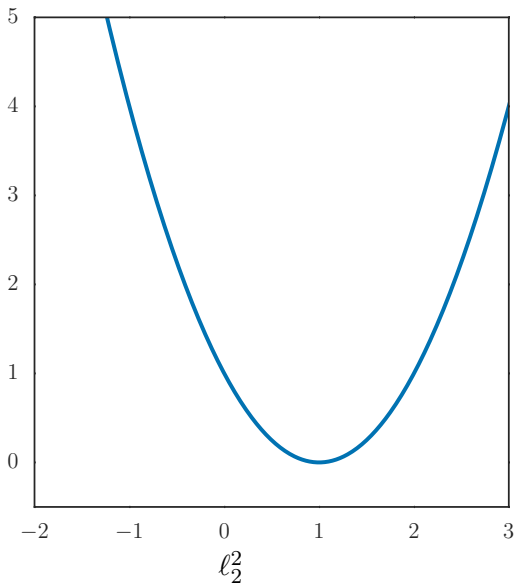
- ▶ Exemple (classification) :  $\ell(\mathbf{x}, y, h(\mathbf{x})) = \begin{cases} 1, & \text{si } h(\mathbf{x}) \neq y, \\ 0, & \text{sinon.} \end{cases}$
- ▶ Exemple (régression) :  $\ell(\mathbf{x}, y, h(\mathbf{x})) = (y - h(\mathbf{x}))^2$

**Objectif** : déterminer à partir de  $\mathcal{D}_n$  la fonction  $h \in \mathcal{H}$  qui minimise le risque  $R(h) = \mathbb{E}_P[\ell(X, Y, h(X))]$

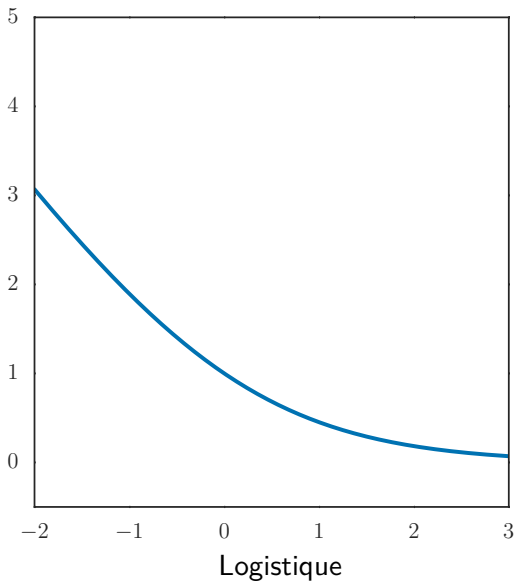
## Divers type d'erreurs



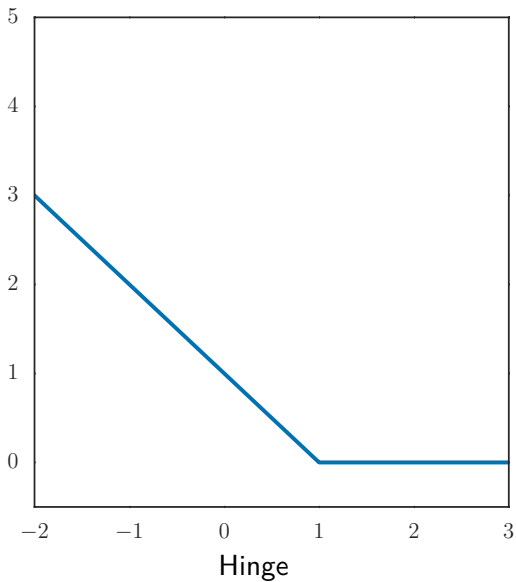
## Divers type d'erreurs



## Divers type d'erreurs

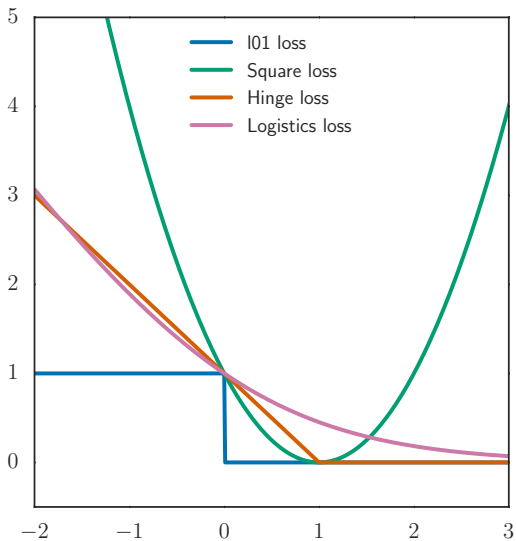


## Divers type d'erreurs





# Divers type d'erreurs



# Apprendre un classifieur

Définir :

- ▶ l'**espace de représentation** des données
- ▶ la **classe des fonctions** de classification binaire considérées

# Apprendre un classifieur

Définir :

- ▶ l'**espace de représentation** des données
- ▶ la **classe des fonctions** de classification binaire considérées
- ▶ la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe

# Apprendre un classifieur

Définir :

- ▶ l'**espace de représentation** des données
- ▶ la **classe des fonctions** de classification binaire considérées
- ▶ la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe
- ▶ l'**algorithme de minimisation** de cette fonction de coût

# Apprendre un classifieur

Définir :

- ▶ l'**espace de représentation** des données
- ▶ la **classe des fonctions** de classification binaire considérées
- ▶ la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe
- ▶ l'**algorithme de minimisation** de cette fonction de coût
- ▶ une **méthode de sélection de modèle** pour définir les hyper-paramètres

# Apprendre un classifieur

Définir :

- ▶ l'**espace de représentation** des données
- ▶ la **classe des fonctions** de classification binaire considérées
- ▶ la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe
- ▶ l'**algorithme de minimisation** de cette fonction de coût
- ▶ une **méthode de sélection de modèle** pour définir les hyper-paramètres
- ▶ une méthode d'évaluation des performances

# Apprendre un classifieur

Définir :

- ▶ l'**espace de représentation** des données
- ▶ la **classe des fonctions** de classification binaire considérées
- ▶ la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe
- ▶ l'**algorithme de minimisation** de cette fonction de coût
- ▶ une **méthode de sélection de modèle** pour définir les hyper-paramètres
- ▶ une méthode d'évaluation des performances

# Sommaire

## Introduction

Rappels de classification

Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

Structure efficace : les arbres

Séparateurs élémentaires

Algorithme efficace

## Détails et variations

Fonction de coût

Fonction d'impureté

Critères d'arrêt et variantes

Sélection de modèle



## Classe des fonctions considérées

La collection  $\mathcal{H}$  des classifieurs/estimateurs est une sous-partie de l'ensemble des **fonctions constantes par morceaux**.

Simplification : séparations parallèles aux axes

Plus précisément, les composantes constantes sont de la forme

$$\mathcal{C} = \{\mathbf{x} \in \mathcal{X} : \mathbf{x}^{j_1} \in [\underline{\mathbf{x}}^{j_1}, \overline{\mathbf{x}}^{j_1}], \dots, \mathbf{x}^{j_r} \in [\underline{\mathbf{x}}^{j_r}, \overline{\mathbf{x}}^{j_r}]\}$$

pour  $r \in \llbracket 1, p \rrbracket$  et  $(j_1, \dots, j_r) \in \llbracket 1, p \rrbracket^r$

Pour une fonction ayant  $M$  composantes constantes, l'estimateur s'écrit :

$$\hat{h} = \sum_{m=1}^M \hat{\alpha}_m \mathbb{1}_{\mathcal{C}_m}$$

Rem: les  $\mathcal{C}_m$  forment une partition de l'espace (recouvrement sans chevauchement) :

$$\mathcal{C}_1 \sqcup \dots \sqcup \mathcal{C}_M = \mathcal{X}$$

# Classifieur/Estimateur associé

Prenons une partition  $\mathcal{C}_1 \sqcup \dots \sqcup \mathcal{C}_M = \mathcal{X}$  et un prédicteur associé :

$$\hat{h} = \sum_{m=1}^M \hat{\alpha}_m \mathbb{1}_{\mathcal{C}_m}$$

Au vu de  $\mathcal{D}_n$  on choisit les coefficients  $\hat{\alpha}_m$  ainsi : pour tout  $\mathbf{x} \in \mathcal{X}$ , il existe un  $m \in \llbracket 1, M \rrbracket$  tel que  $\mathbf{x} \in \mathcal{C}_m$ , et l'on obtient

- Pour la classification, (par “vote majoritaire”) :

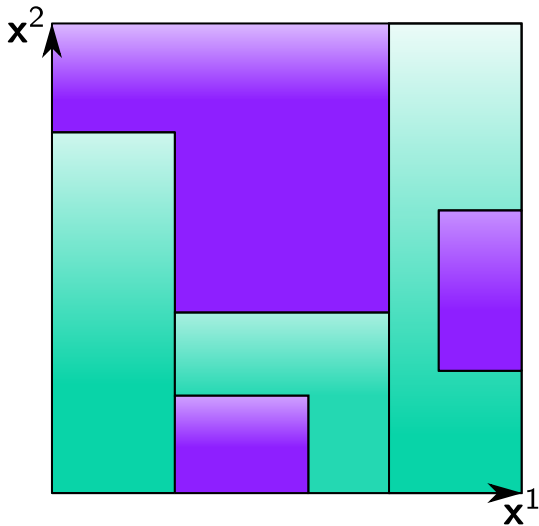
$$\hat{h}(\mathbf{x}) = \arg \max_{k=1, \dots, K} \sum_{\mathbf{x}_i \in \mathcal{C}_m} \mathbb{1}(y_i = k)$$

- Pour la régression :

$$\hat{h}(\mathbf{x}) = \frac{1}{|\mathcal{C}_m|} \sum_{\mathbf{x}_i \in \mathcal{C}_m} y_i$$

Rem: lien avec un estimateur “plug-in” estimant  $\mathbb{P}(Y = 1|X)$

## Exemple de fonction constante par morceaux



# Classifieur/Estimateur associé

- ▶ Motivation : interprétation, seuils “interprétables”
- ▶ Limites :
  - ▶ difficile de décrire efficacement toutes ces fonctions
  - ▶ si la partition est fixée avant de voir les données, la plupart des composantes seront vides.

---

**Exo:** quel problème cela pose-t-il en régression ? en classification ?

---

Solution possible : apprendre (efficacement !) la partition grâce aux données !

# Sommaire

## Introduction

Rappels de classification

Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

Structure efficace : les arbres

Séparateurs élémentaires

Algorithme efficace

## Détails et variations

Fonction de coût

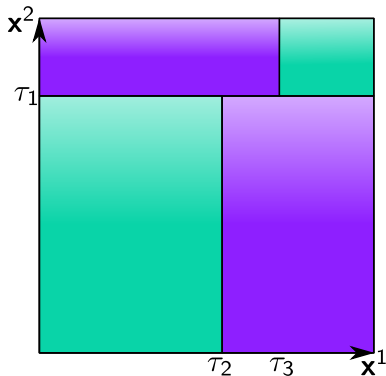
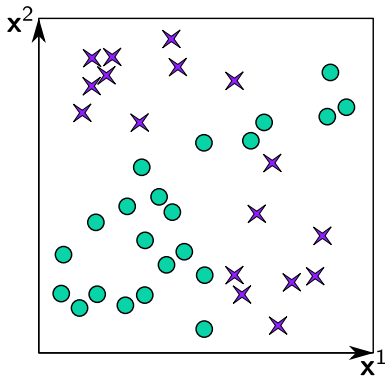
Fonction d'impureté

Critères d'arrêt et variantes

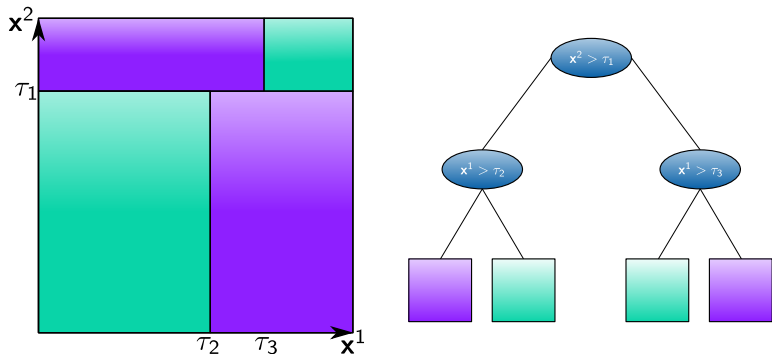
Sélection de modèle

# Arbres de décision

Invention quasi simultanée entre 1979 et 1983 par Breiman *et al.* (1984) (CART, Berkeley, USA) et Quinlan (1986) (ID3, Sydney, Australie) dans deux communautés différentes : en statistique (CART), en *machine learning* (ID3)



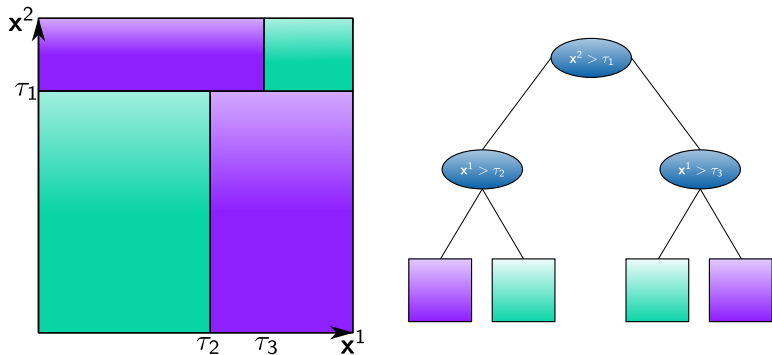
# Arbres de décision



## Première idée :

Utiliser non pas un mais plusieurs séparateurs linéaires pour construire des frontières de décision non linéaires.

# Arbres de décision

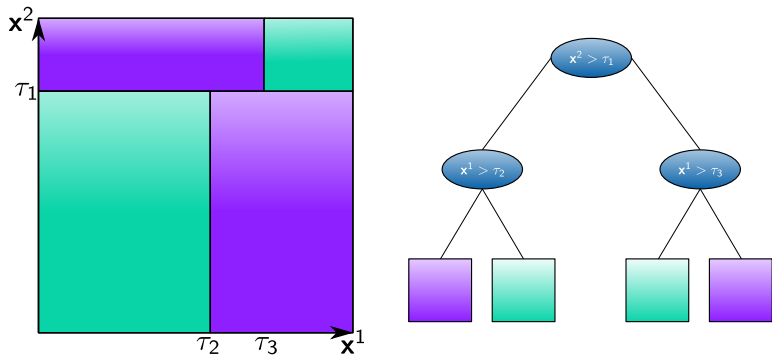


## Deuxième idée :

Utiliser des séparateurs linéaires orthogonaux à chaque direction, *i.e.*, des hyperplans  $\{\mathbf{x} \in \mathcal{X} : \mathbf{x}^j = \tau\}$  pour l'interprétabilité.



# Arbres de décision



## Troisième idée :

Utiliser un prédicteur représenté par un d'arbre : chaque nœud est associé à un hyperplan séparateur  $\{\mathbf{x} \in \mathcal{X} : \mathbf{x}^j = \tau\}$  ; chaque feuille est associée à une fonction constante, *i.e.*, une classe.

# Sommaire

## Introduction

Rappels de classification

Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

Structure efficace : les arbres

**Séparateurs élémentaires**

Algorithme efficace

## Détails et variations

Fonction de coût

Fonction d'impureté

Critères d'arrêt et variantes

Sélection de modèle

# Règles logiques

Après l'apprentissage, on connaît les variables explicatives qui interviennent dans la fonction de décision construite

Rem: localement seule une (faible) partie des variables sont discriminantes.

L'arbre code pour un ensemble de règles logiques du type :

“si  $(\mathbf{x}^{j_1} > \tau_1)$  et  $(\mathbf{x}^{j_2} \leq \tau_2)$  et ... alors  $\mathbf{x}$  est de la classe  $k$ ”

Rem: la prédiction est rapide une fois la règle apprise, le temps de prédiction ne dépend que du nombre de seuils à tester

# Séparateur linéaire orthogonal aux axes

Rappel :  $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^p)$ ,  $p$  variables

- ▶ Variable continue (ou binaire) :  $j^{\text{e}}$  variable  $\mathbf{x}^j$ , seuil  $\tau$  :

$$t_{j,\tau}(\mathbf{x}) = \text{sign}(\mathbf{x}^j - \tau) = \begin{cases} +1, & \text{si } \mathbf{x}^j > \tau \\ -1, & \text{si } \mathbf{x}^j < \tau \end{cases} \quad (1)$$

- ▶ Variable catégorielle à  $M$  modalités  $\{v_1^j, \dots, v_M^j\}$  :

$$t_{j,\mathbf{v},m}(\mathbf{x}) = \mathbb{1}(\mathbf{x}^j = v_m^j) \quad (2)$$

Rem: cette dernière version du traitement des variables catégorielles discrimine simplement : “une modalité” vs. “toutes les autres” ; les variables  $K$ -catégorielles peuvent être transformées en  $K$  variables binaires si besoin

# Sommaire

## Introduction

Rappels de classification

Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

Structure efficace : les arbres

Séparateurs élémentaires

Algorithme efficace

## Détails et variations

Fonction de coût

Fonction d'impureté

Critères d'arrêt et variantes

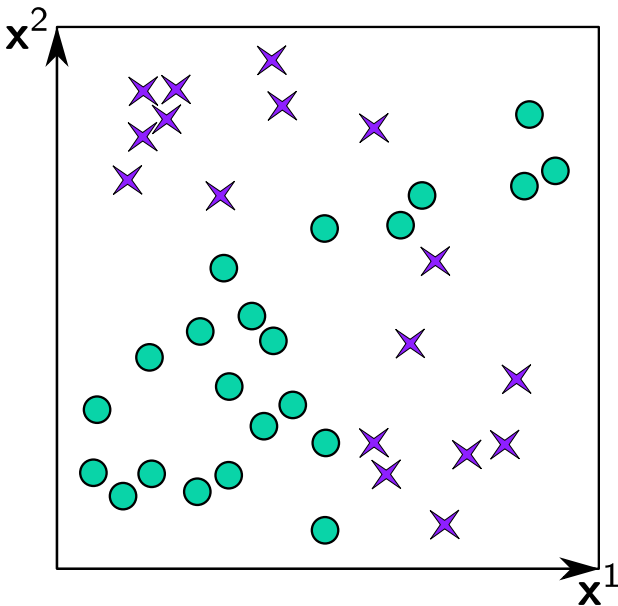
Sélection de modèle

# Algorithme récursif de construction

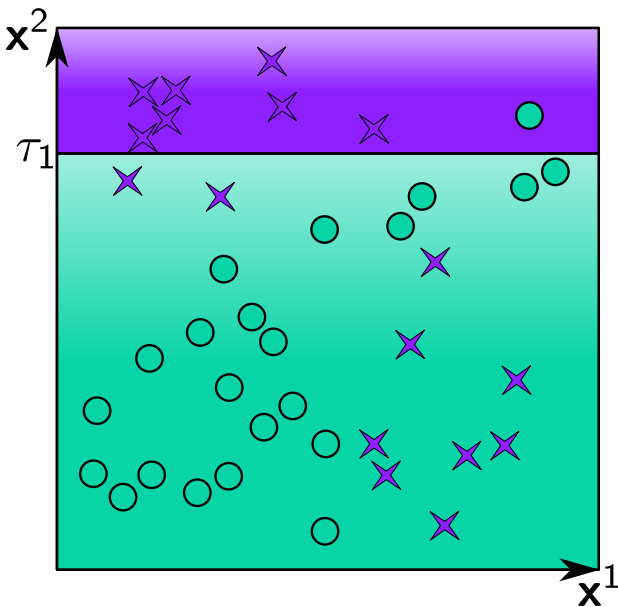
Cas d'un arbre binaire :

1. Soit  $\mathcal{D}_n$  l'ensemble d'apprentissage
2. Construire un nœud racine
3. Chercher la meilleure séparation  $t : \mathcal{X} \rightarrow \{0, 1\}$  à appliquer sur  $\mathcal{D}_n$  telle que le coût local  $L(t, \mathcal{D}_n)$  soit minimal
4. Associer le séparateur choisi au nœud courant et séparer l'ensemble d'apprentissage courant  $\mathcal{D}_n$  en  $\mathcal{D}_n^d$  et  $\mathcal{D}_n^g$  à l'aide de ce séparateur.
5. Construire un nœud fils à droite et un nœud fils à gauche
6. Mesurer le critère d'arrêt à droite, s'il est vérifié, le nœud droit devient une feuille sinon aller en 3 avec  $\mathcal{D}_n^d$  comme ensemble courant
7. Mesurer le critère d'arrêt à gauche, s'il est vérifié, le nœud gauche devient une feuille sinon aller en 3 avec  $\mathcal{D}_n^g$  comme ensemble courant

## Exemple visuel

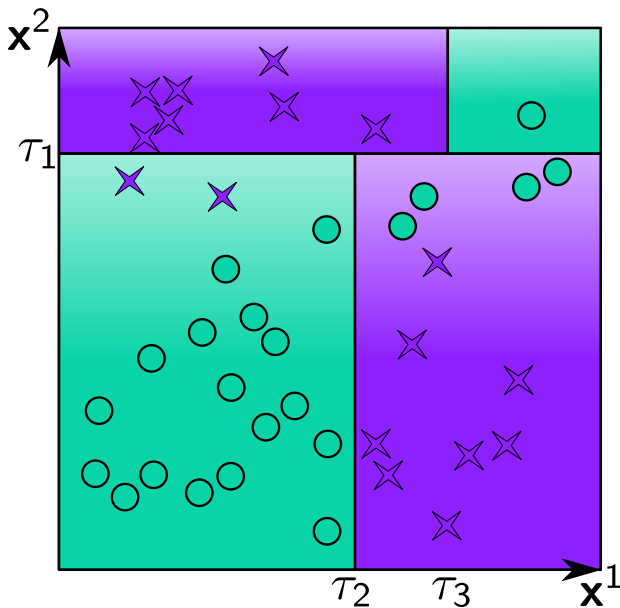


## Exemple visuel





## Exemple visuel



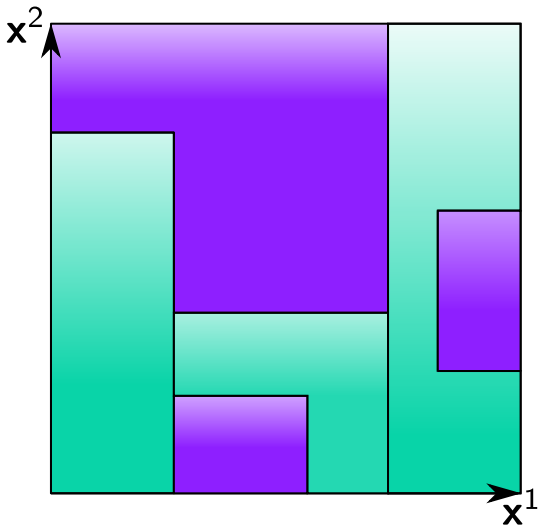
## Point de vue glouton ( : *greedy*)

Tout comme les méthodes *stagewise/stepwise*/OMP en régression linéaire, l'algorithme CART (précédent) est **glouton**.

On n'optimise pas un critère globale : on cherche localement les décisions optimales (au sens de  $L$ ). On espère donc qu'une optimisation locale des décisions permette une décision globalement "optimale".

*cf.* MDI 720 / SD204 pour le cas des méthodes gloutonnes pour en régression linéaire

## Contre-exemple : partition non issue d'un arbre



# Sommaire

## Introduction

Rappels de classification

Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

Structure efficace : les arbres

Séparateurs élémentaires

Algorithme efficace

## Détails et variations


Fonction de coût

Fonction d'impureté

Critères d'arrêt et variantes

Sélection de modèle

# Probabilités / simplexe

L'idée principale est maintenant de définir une notion de pureté/impureté d'une coupure, pour grandir l'arbre par coupures ( : *splitting*) successives

On définit pour un ensemble  $\mathcal{D}_n$  (avec  $n$  exemples étiquetés) la distribution de probabilités pour la classe  $k$  (avec  $K$  classes) par :

$$\rho_k(\mathcal{D}_n) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}(y_i = k)$$

Rem: en notant le simplexe (de dimension  $K$ )

$$\Delta_K := \left\{ \rho \in \mathbb{R}^K : \sum_{k=1}^K \rho_k = 1 \text{ et } \forall k \in \llbracket 1, K \rrbracket, \rho_k \geq 0 \right\} \text{ on a donc}$$

que  $p(\mathcal{D}_n) = (\rho_1(\mathcal{D}_n), \dots, \rho_K(\mathcal{D}_n))^T \in \Delta_K$

Rem: on identifie  $\Delta_K$  aux probabilités discrètes avec  $K$  modalités

## Coupure

Pour un ensemble d'exemples d'apprentissage  $\mathcal{D}_n$  et une fonction de séparation binaire  $t_{j,\tau}$ , notons

$$\mathcal{D}_n^d(j, \tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) > 0\}$$

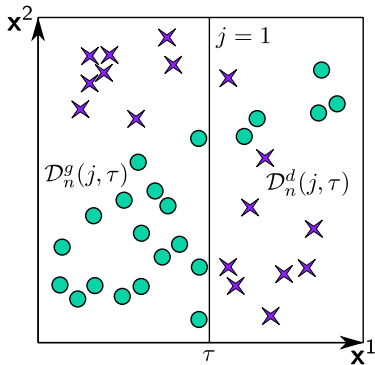
$$\mathcal{D}_n^g(j, \tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) \leq 0\}$$

# Coupure

Pour un ensemble d'exemples d'apprentissage  $\mathcal{D}_n$  et une fonction de séparation binaire  $t_{j,\tau}$ , notons

$$\mathcal{D}_n^d(j, \tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) > 0\}$$

$$\mathcal{D}_n^g(j, \tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) \leq 0\}$$

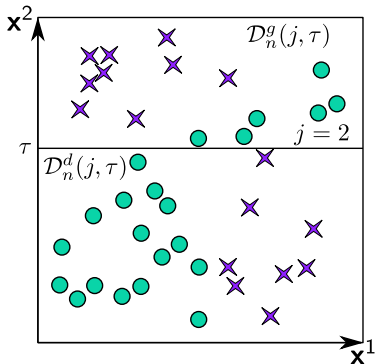
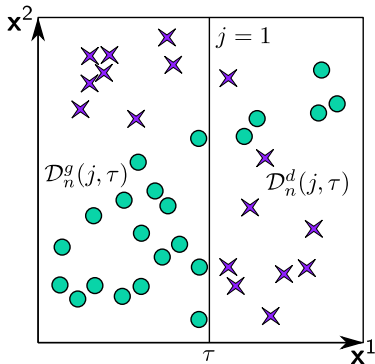


# Coupure

Pour un ensemble d'exemples d'apprentissage  $\mathcal{D}_n$  et une fonction de séparation binaire  $t_{j,\tau}$ , notons

$$\mathcal{D}_n^d(j, \tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) > 0\}$$

$$\mathcal{D}_n^g(j, \tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) \leq 0\}$$





# Fonction de coût locale

Parmi tous les paramètres  $(j, \tau) \in \{1, \dots, p\} \times \{\tau_1, \dots, \tau_m\}$ , on cherche  $\hat{j}$  et  $\hat{\tau}$  qui minimisent une fonction de coût :

$$L(t_{j,\tau}, \mathcal{D}_n) = \frac{n_g}{n} H(\rho(\mathcal{D}_n^g(j, \tau))) + \frac{n_d}{n} H(\rho(\mathcal{D}_n^d(j, \tau)))$$

avec  $n_g = |\mathcal{D}_n^g(j, \tau)|$  et  $n_d = |\mathcal{D}_n^d(j, \tau)|$

- ▶  $H$  est une fonction “d’impureté”
- ▶ le coût total est la somme de l’impureté de chaque sous-partie, pondérée par son nombre d’échantillons
- ▶ un nombre fini de seuils suffit (au plus  $n$ )
- ▶ la notion d’impureté d’un échantillon  $\mathcal{D}_n$  est une uniquement fonction de la distribution des probabilités  $\rho(\mathcal{D}_n)$

# Sommaire

## Introduction

Rappels de classification

Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

Structure efficace : les arbres

Séparateurs élémentaires

Algorithme efficace

## Détails et variations

Fonction de coût

**Fonction d'impureté**

Critères d'arrêt et variantes

Sélection de modèle

## Fonction d'impureté

Rappel :  $\Delta_K := \left\{ \rho \in \mathbb{R}^K : \sum_{k=1}^K \rho_k = 1 \text{ et } \forall k \in \llbracket 1, K \rrbracket, \rho_k \geq 0 \right\}$   
simplexe de dimension  $K$

### Définition : fonction d'impureté (d'une probabilité)

Une fonction d'**impureté**, est une fonction  $H : \Delta_K \rightarrow \mathbb{R}$  telle que :

1.  $H$  est maximum au point  $(\frac{1}{K}, \dots, \frac{1}{K})^\top$
2.  $H$  atteint son minimum seulement aux points  $(1, 0, \dots, 0)^\top, (0, 1, 0, \dots, 0)^\top, \dots, (0, \dots, 0, 1)^\top$
3.  $H$  est une fonction symétrique en  $\rho_1, \dots, \rho_K$

Interprétation :

- 1) la distribution la plus impure partage les classes uniformément
- 2) les distributions les plus pures sont celles dégénérées
- 3) toutes les classes ont la même importance

cf. Breiman *et al.* (1984, page 32)

## Fonction d'impureté : cas binaire ( $K = 2$ )

En dimension 2, on représente la fonction d'impureté par une fonction :

- ▶  $H : [0, 1] \mapsto [0, 1]$
- ▶  $H$  est maximale en  $\frac{1}{2}$
- ▶  $H$  est minimale en 0 et en 1
- ▶  $H$  est symétrique par rapport à l'axe  $x = \frac{1}{2}$

# Critères de coût (I) : Erreur de classification

**Erreur de classification :**  $H_{\text{mis}}(\mathcal{D}_n) = 1 - \rho_{\hat{k}}(\mathcal{D}_n)$ ,

avec  $\hat{k}(\mathcal{D}_n)$  : classe majoritaire dans  $\mathcal{D}_n$ , *i.e.*,

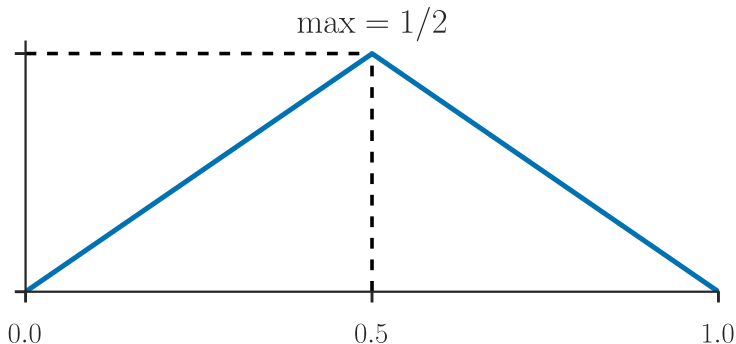
$$\hat{k} = \arg \max_{k=1, \dots, K} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_n} \mathbb{1}(y_i = k)$$

Interprétation : on compte la probabilité de se tromper (*i.e.*, de ne pas choisir la classe majoritaire)

# Critères de coût (I) : Erreur de classification

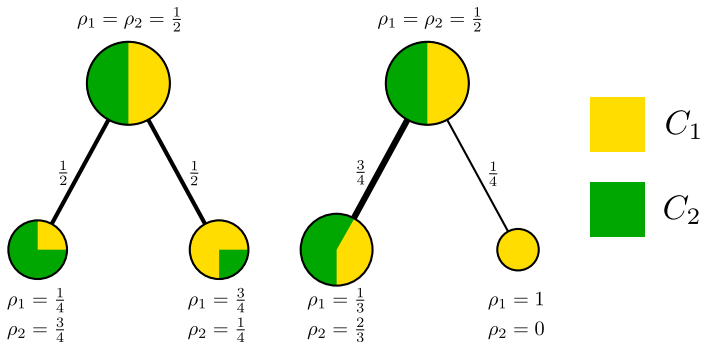
Application dans le cas binaire :

$$H_{\text{mis}}(\mathcal{D}_n) = 1 - \max_{k=1,2} \rho_k(\mathcal{D}_n) = \min(\rho_1(\mathcal{D}_n), 1 - \rho_1(\mathcal{D}_n))$$



## Limites de ce choix

- ▶ pour une zone avec une classe très majoritaire il se peut qu'aucune coupure ne produisent de réduction d'impureté
- ▶ fonction non-différentiable (optimisations plus dure)
- ▶ dans certains cas la pureté induite par des nœuds purs est négligée par ce critère :



$$L_{\text{mis}} = \frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{4} = \frac{3}{4} \cdot \frac{1}{3} + \frac{1}{4} \cdot 0 = \frac{1}{4}$$

# Impureté stricte

## Définition : Impureté stricte

Une fonction d'impureté  $H : \Delta_K \rightarrow \mathbb{R}$  est **stricte** si pour toutes distributions  $\rho, \rho'$  dans  $\Delta_K$  avec  $\rho \neq \rho'$  et tout  $\alpha \in ]0, 1[$  on a :

$$H(\alpha\rho + (1 - \alpha)\rho') > \alpha H(\rho) + (1 - \alpha)H(\rho')$$

Conséquence Breiman *et al.* (1984), page 100, si  $H$  est une fonction d'impureté pure

$$L(t_{j,\tau}, \mathcal{D}_n) = \frac{n_g}{n} H(\rho(\mathcal{D}_n^g(j, \tau))) + \frac{n_d}{n} H(\rho(\mathcal{D}_n^d(j, \tau))) \leq H(\rho(\mathcal{D}_n))$$
$$n_g = |\mathcal{D}_n^g(j, \tau)| \quad \text{et} \quad n_d = |\mathcal{D}_n^d(j, \tau)|$$

et il y a égalité si et seulement si  $\rho(\mathcal{D}_n) = \rho(\mathcal{D}_n^g) = \rho(\mathcal{D}_n^d)$

Interprétation : mélanger ne fait qu'augmenter l'impureté, ce qui traduit la (stricte) concavité de  $H$



## Critères de coût (II) : Entropie

**Entropie :**  $H_{\text{ent}}(\mathcal{D}_n) = - \sum_{k=1}^K \rho_k(\mathcal{D}_n) \log \rho_k(\mathcal{D}_n)$

Pour plus de détails sur l'entropie et ses propriétés caractéristiques, cf. [Roman \(1992\), Chapitre 1](#)

Rem: lien entre l'entropie de Shannon et celle de Boltzmann (physique)

---

**Exo**: L'entropie et la divergence de Kullback-Leibler sont liées par  $H_{\text{ent}}(\mathcal{D}_n) = \log(K) - D_{\text{KL}}(\rho(\mathcal{D}_n) \parallel \rho_{\text{unif}})$ , où pour tout  $\rho, \rho' \in \Delta_K$  :

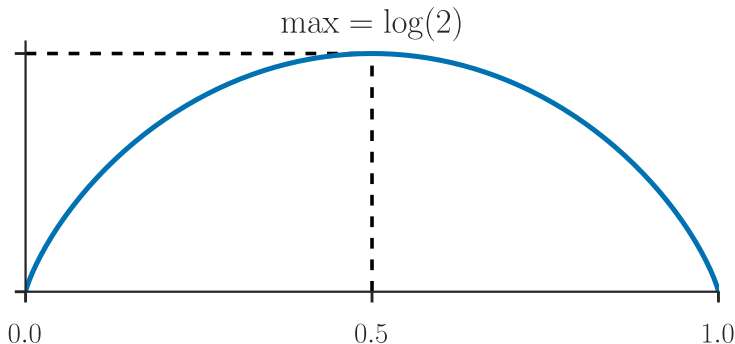
$$D_{\text{KL}}(\rho \parallel \rho') = \sum_{k=1}^K \rho_k \log \left( \frac{\rho_k}{\rho'_k} \right)$$

---

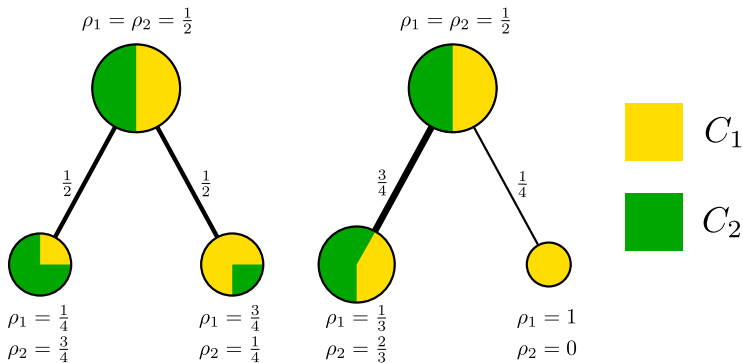
## Critères de coût (II) : Entropie

Application dans le cas binaire :

$$H_{\text{ent}}(\mathcal{D}_n) = -\rho_1(\mathcal{D}_n) \log(\rho_1(\mathcal{D}_n)) - (1 - \rho_1(\mathcal{D}_n)) \log(1 - \rho_1(\mathcal{D}_n))$$



## Retour sur un exemple



---

**Exo:** Calculer  $L_{\text{ent}}$  associée à  $H_{\text{ent}}$ .

---

# Critères de coût (III) : indice de Gini

## Indice de Gini :

$$H_{\text{Gini}}(\mathcal{D}_n) = \sum_{k=1}^K \rho_k(\mathcal{D}_n)(1 - \rho_k(\mathcal{D}_n)) = \sum_{k=1}^K \sum_{\substack{k'=1 \\ k' \neq k}}^K \rho_k(\mathcal{D}_n)\rho_{k'}(\mathcal{D}_n)$$

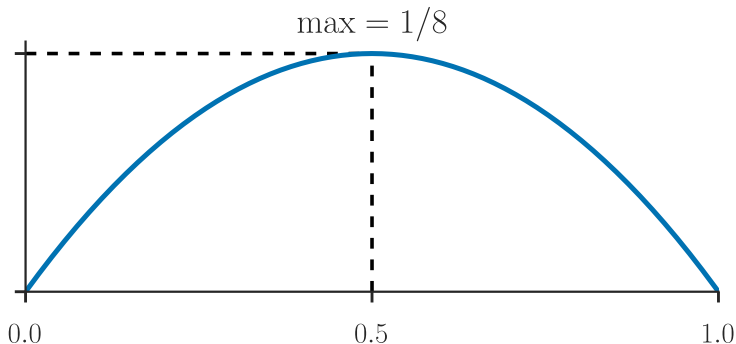
Interprétation des deux formulations :

- ▶ Créer des variables binaires  $X_i^k = \mathbb{1}(y_i = k)$ , pour  $i = 1, \dots, n$  ; leur variance vaut  $\rho_k(\mathcal{D}_n)(1 - \rho_k(\mathcal{D}_n))$ , l'indice de Gini mesure donc la somme/moyenne des variances des classes binarisées
- ▶ Remplacer le vote majoritaire par la règle “Choisir la classe  $k$  avec probabilité  $\rho_k$ ”, l'indice de Gini est alors la probabilité d'erreur pour cette règle Breiman *et al.* (1984), p. 104

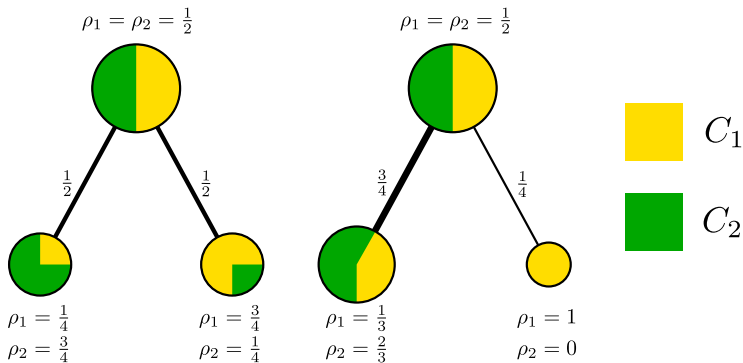
## Critères de coût (III) : indice de Gini

Application dans le cas binaire :

$$H_{\text{Gini}}(\mathcal{D}_n) = 2 \cdot \rho_1(\mathcal{D}_n) (1 - \rho_1(\mathcal{D}_n))$$



## Retour sur un exemple



---

**Exo:** Calculer  $L_{\text{Gini}}$  associée à  $H_{\text{Gini}}$

---

# Sommaire

## Introduction

Rappels de classification

Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

Structure efficace : les arbres

Séparateurs élémentaires

Algorithme efficace

## Détails et variations

Fonction de coût

Fonction d'impureté

Critères d'arrêt et variantes

Sélection de modèle

# Critères d'arrêt

On peut s'arrêter localement (dans une branche), dès :

- qu'on atteint la profondeur maximale
- qu'on atteint le nombre maximale de feuilles
- qu'on atteint le nombre minimal d'exemples dans un nœud (pas assez d'exemples)

Rem: si le nombre minimal d'exemples vaut 1, l'ensemble d'apprentissage est appris jusqu'au bout (dans les limites computationnelles et de mémoire) : risque de **sur-apprentissage** !



# Variables catégorielles

- ▶ Pour avoir un arbre binaire : si une variable catégorielle est à  $M$  valeurs/modalités, on la transforme en  $M$  variables binaires
- ▶ L'algorithme d'apprentissage est approprié pour traiter aussi bien des problèmes binaires que multi-classes
- ▶ Les classes avec beaucoup de modalités ont tendance à être favorisées car plus il y a de classes, plus il y a de chance de trouver une bonne coupure.

Attention donc au sur-apprentissage (éviter si possible de telles variables)

## Matrice de perte / Asymétrie

Quand se tromper entre deux classes n'a pas les mêmes conséquences, (cf. spam, médecine, etc.), on introduit une matrice de coût  $L \in \mathbb{R}^{K \times K}$ , avec  $K$  le nombre de classes possibles pour  $Y$  :

$$C_{k,k'} = 0 \text{ si } k = k'$$

$$C_{k,k'} \geq 0 \text{ si } k \neq k'$$

Erreur moyenne en choisissant la classe  $k$  :

$$\text{Erreur de classification : } \sum_{k=1}^K C_{k,k'} \rho_k(\mathcal{D}_n)$$

$$\text{Indice de Gini : } \sum_{k=1}^K \sum_{\substack{k'=1 \\ k' \neq k}}^K C_{k,k'} \rho_k(\mathcal{D}_n) \rho_{k'}(\mathcal{D}_n)$$

Rem: dans le cas binaire ( $K = 2$ ), une autre approche consiste à pondérer les observations de la classe  $k$  par  $C_{k,k'}$  (avec  $k \neq k'$ ).

# Arbres de régression

Le fonctionnement pour la régression est pratiquement identique, pour construire l'arbre, seul le critère de coût change : on minimise

$$L(t_{j,\tau}, \mathcal{D}_n) = \frac{n_g}{n} H(\mathcal{D}_n^g(j, \tau)) + \frac{n_d}{n} H(\mathcal{D}_n^d(j, \tau))$$

avec la variance comme mesure d'impureté

$$H(\mathcal{D}_n) = \overline{\text{var}}(\mathcal{D}_n) := \frac{1}{|\mathcal{D}_n|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_n} (y_i - \bar{y}_n)^2$$

où

$$\bar{y}_n =:= \frac{1}{|\mathcal{D}_n|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_n} y_i$$

Rem: on cherche à maximiser l'homogénéité des sorties, ce qui dans ce cas revient à trouver la partition de risque quadratique minimale

# Sommaire

## Introduction

Rappels de classification

Estimateurs/Classifieurs constants par morceaux

## Arbres de décision

Structure efficace : les arbres

Séparateurs élémentaires

Algorithme efficace

## Détails et variations

Fonction de coût

Fonction d'impureté

Critères d'arrêt et variantes

Sélection de modèle

# Sélection de modèle

On s'intéressera à déterminer un des hyper-paramètres suivants :

- Profondeur maximale
- nombre maximal de feuilles
- maxima minimal d'exemples dans une feuille/nœud

→ potentiellement par **validation croisée**

## Élagage ( : *pruning*)

On utilise un ensemble de validation pour re-visiter un arbre appris sans limite sur un ensemble d'apprentissage. On ne garde que les branches qui apportent une amélioration en validation *cf. Hastie et al. (2009)* pour plus de détails

Rem: utile pour l'interprétation, mais coûteux et inutile si l'on combine plusieurs arbres (*cf. "forêts aléatoires"*)

Rem: l'élagage n'est pas disponible dans `sklearn` (utiliser si besoin `rpart` de R)

# Avantages et inconvénients des arbres de décision

## Avantages

- ▶ Construit une fonction de décision non linéaire, interprétable
- ▶ Consistance des arbres (cf. Scott et Nowak (2006) pour une revue détaillée)
- ▶ Fonctionne pour le multi-classe
- ▶ Prise de décision efficace :  $O(\log F)$ ,  $F$  : nombre de feuilles
- ▶ Fonctionne pour des variables continues et catégorielles

# Avantages et inconvénients des arbres de décision

## Inconvénients

- ▶ Estimateur à large variance, instabilité : une petite variation dans l'ensemble d'apprentissage engendre un arbre complètement différent → d'où l'intérêt des combinaisons linéaires d'arbres (*bagging*, forêt, *boosting*)
- ▶ Pas d'optimisation globale



# Références I

- ▶ L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone.  
*Classification and regression trees*.  
Wadsworth Statistics/Probability Series. Wadsworth Advanced Books and Software, Belmont, CA, 1984.
- ▶ T. Hastie, R. Tibshirani, and J. Friedman.  
*The Elements of Statistical Learning*.  
Springer Series in Statistics. Springer, New York, second edition, 2009.
- ▶ J. R. Quinlan.  
Induction of decision trees.  
*Maching Learning*, 1 :81–106, 1986.
- ▶ S. Roman.  
*Coding and information theory*, volume 134 of *Graduate Texts in Mathematics*.  
Springer-Verlag, New York, 1992.

## Références II

- ▶ C. Scott and R. D. Nowak.  
Minimax-optimal classification with dyadic decision trees.  
*IEEE Trans. Inf. Theory*, 52(4) :1335–1353, 2006.