

# **MS BGD**

## **MDI 343 : Arbres pour l'apprentissage**

**Joseph Salmon**

<http://josephsalmon.eu>

Télécom Paristech, Institut Mines-Télécom

# 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 modèles

# 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 modèles

# Classification supervisée et régression

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

$Y$  : variable **à prédire**, aléatoires dans  $\mathcal{Y} = \{C_1, \dots, C_K\}$   
(classification avec  $K$  classes) ou  $\mathcal{Y} = \mathbb{R}$  (régression)

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

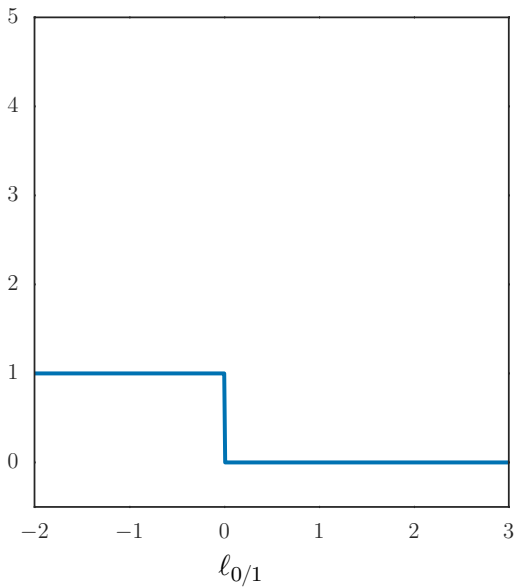
$\mathcal{H}$  : collection de **classifieurs/estimateurs**,  $h : \mathcal{X} \mapsto \mathcal{Y}$

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

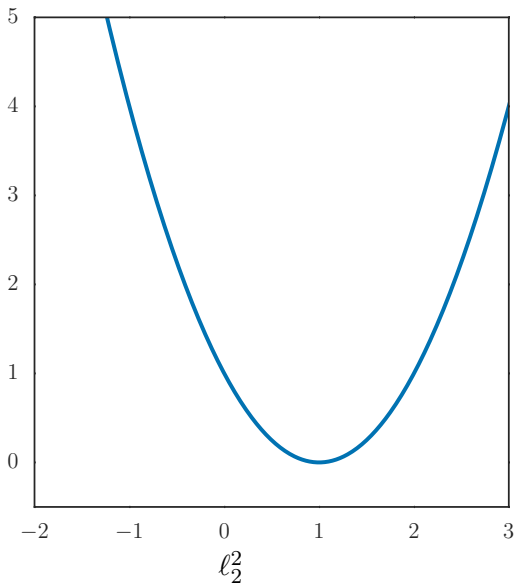
- ▶ Exemple (classification) :  $\ell(\mathbf{x}, y, h(\mathbf{x})) = \begin{cases} 1, & \text{si } h(\mathbf{x}) \neq y, \\ 0, & \text{sinon.} \end{cases}$
- ▶ Exemple (régression) :  $\ell(\mathbf{x}, y, h(\mathbf{x})) = (y - h(\mathbf{x}))^2$

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

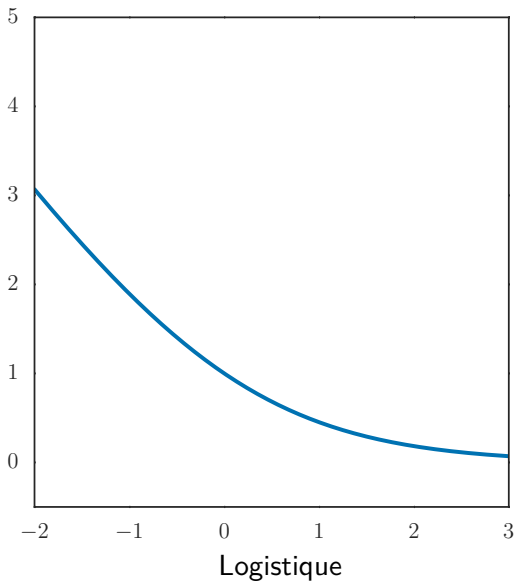
## Divers type d'erreurs



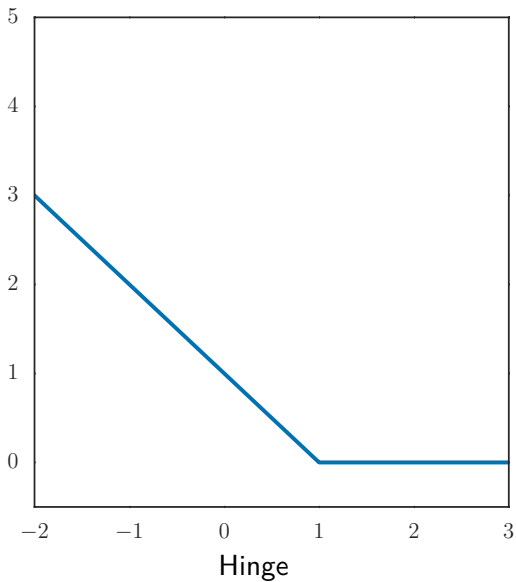
## Divers type d'erreurs



## Divers type d'erreurs

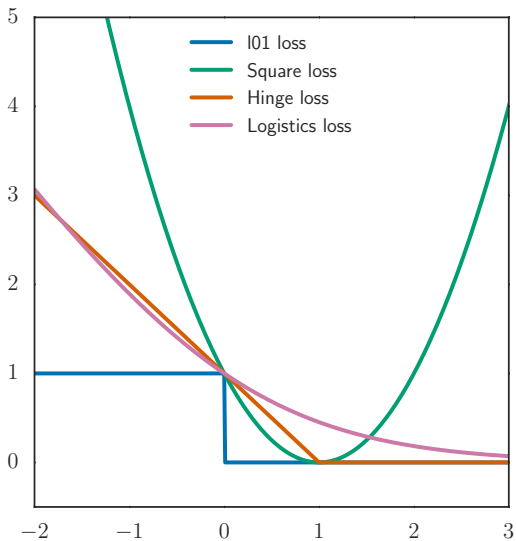


## Divers type d'erreurs





# Divers type d'erreurs



# Apprendre un classifieur/prédicteur

Définir :

- l'**espace de représentation** des données  $\mathcal{X}$
- la **classe des fonctions** considérées  $\mathcal{H}$

# Apprendre un classifieur/prédicteur

Définir :

- ▶ l'**espace de représentation** des données  $\mathcal{X}$
- ▶ la **classe des fonctions** considérées  $\mathcal{H}$
- ▶ la **fonction de coût**  $\ell$  à minimiser pour obtenir la meilleur fonction  $h$

# Apprendre un classifieur/prédicteur

Définir :

- ▶ l'**espace de représentation** des données  $\mathcal{X}$
- ▶ la **classe des fonctions** considérées  $\mathcal{H}$
- ▶ la **fonction de coût**  $\ell$  à minimiser pour obtenir la meilleur fonction  $h$
- ▶ l'**algorithme de minimisation** de cette fonction de coût

# Apprendre un classifieur/prédicteur

Définir :

- ▶ l'**espace de représentation** des données  $\mathcal{X}$
- ▶ la **classe des fonctions** considérées  $\mathcal{H}$
- ▶ la **fonction de coût**  $\ell$  à minimiser pour obtenir la meilleur fonction  $h$
- ▶ l'**algorithme de minimisation** de cette fonction de coût
- ▶ une **méthode de sélection de modèle** pour calibrer les hyper-paramètres (e.g., validation croisée)

# Apprendre un classifieur/prédicteur

Définir :

- ▶ l'**espace de représentation** des données  $\mathcal{X}$
- ▶ la **classe des fonctions** considérées  $\mathcal{H}$
- ▶ la **fonction de coût**  $\ell$  à minimiser pour obtenir la meilleur fonction  $h$
- ▶ l'**algorithme de minimisation** de cette fonction de coût
- ▶ une **méthode de sélection de modèle** pour calibrer les hyper-paramètres (e.g., validation croisée)
- ▶ un protocole **d'évaluation** des performances

# Apprendre un classifieur/prédicteur

Définir :

- ▶ l'**espace de représentation** des données  $\mathcal{X}$
- ▶ la **classe des fonctions** considérées  $\mathcal{H}$
- ▶ la **fonction de coût**  $\ell$  à minimiser pour obtenir la meilleur fonction  $h$
- ▶ l'**algorithme de minimisation** de cette fonction de coût
- ▶ une **méthode de sélection de modèle** pour calibrer les hyper-paramètres (e.g., validation croisée)
- ▶ un protocole **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 modèles



## Classe des fonctions considérées

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

Simplification : les séparations sont **parallèles** aux axes et donc les composantes constantes sont de la forme

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

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

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

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

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

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

et les  $\hat{\alpha}_m \in \mathbb{R}$

# Classifieur/Estimateur associé

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

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

Choix des coefficients  $\hat{\alpha}_m$  (par maximum de vraisemblance) : pour tout  $\mathbf{x} \in \mathcal{X}$ , il existe un  $m \in \llbracket 1, M \rrbracket$  tel que  $\mathbf{x} \in \mathcal{C}_m$ , puis

- Pour la classification :

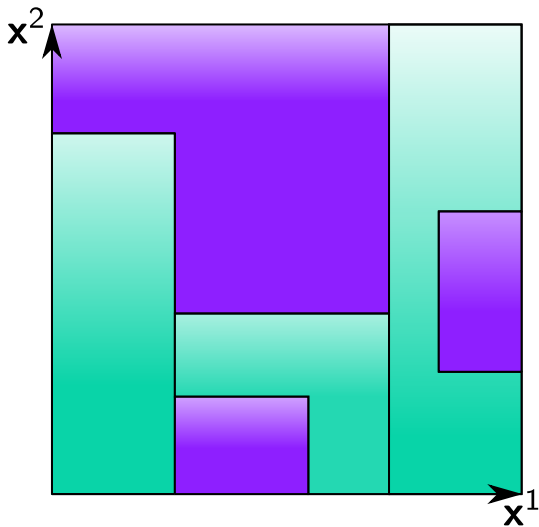
$$\hat{h}(\mathbf{x}) \in \arg \max_{k=1, \dots, K} \sum_{\mathbf{x}_i \in \mathcal{C}_m} \mathbb{1}(y_i = k) \quad (\text{“vote majoritaire”})$$

- Pour la régression :

$$\hat{h}(\mathbf{x}) = \frac{1}{|\{\mathbf{x}_i \in \mathcal{C}_m\}|} \sum_{\mathbf{x}_i \in \mathcal{C}_m} y_i \quad (\text{“moyenne empirique”})$$

Rem: lien avec un estimateur “plug-in”

## 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 ?

---

Alternative : apprendre 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

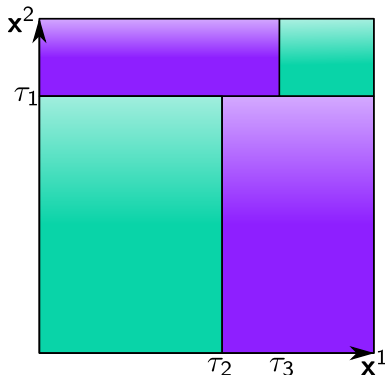
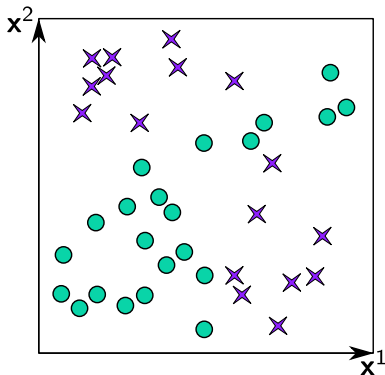
Critères d'arrêt et variantes

Sélection de modèles

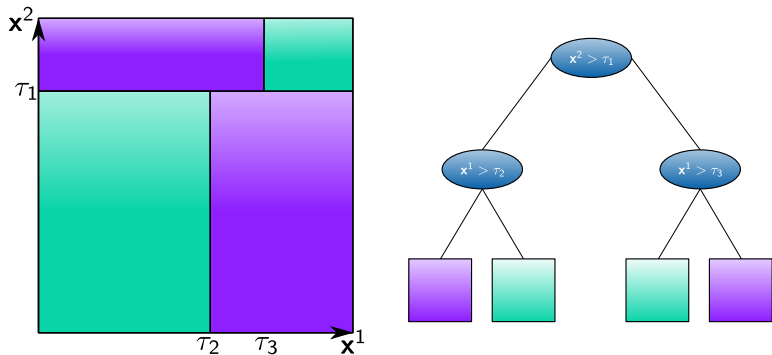
# Arbres de décision

Invention quasi simultanée entre 1979 et 1983

- ▶ CART Breiman *et al.* (1984) ( Berkeley, USA) ; en statistique
- ▶ ID3 Quinlan (1986) (Sydney, Australie) ; en *machine learning*



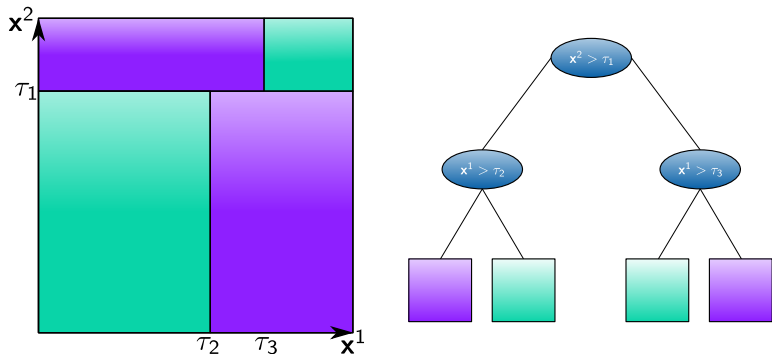
# Arbres de décision



## Première idée :

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

# Arbres de décision

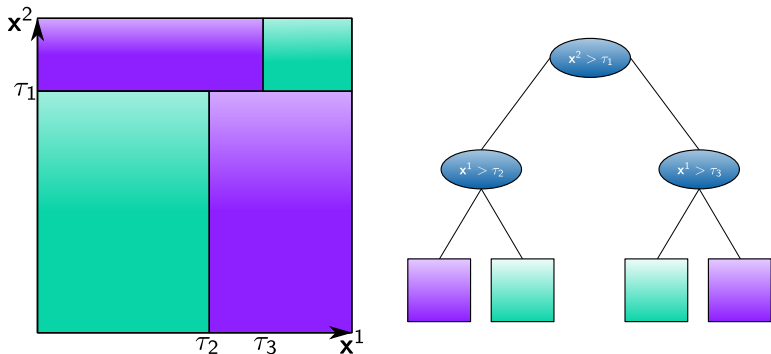


## Deuxième idée :

Utiliser des séparateurs linéaires parallèles aux axes, *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 (donc à 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 modèles

# Règles logiques

Après apprentissage : on connaît les variables explicatives qui interviennent dans la fonction de décision construite

Rem: souvent, une faible partie des variables sont discriminantes, intérêt pour l'**interprétabilité**

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  $k$ ”

Efficacité computationnelle : prédiction très **efficace** une fois la règle apprise, le temps de prédiction ne dépend que du nombre de seuils à tester

Coût : (nombre de nœuds)  $\times$  (coût tester  $\mathbf{x}^{j_1} > \tau_1$  )

# 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  $\tau$  :

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

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

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

Rem: cette dernière version du traitement des variables catégorielles discrimine simplement : “une modalité” vs. “toutes les autres”

Rem: avec `sklearn` il faut utiliser `OneHotEncoder` pour ce cas

# 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 modèles

# Algorithme récursif de construction

Cas d'un arbre binaire :

1. Soit  $\mathcal{D}_n$  l'ensemble d'apprentissage
2. Construire un nœud racine

# 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  $t : \mathcal{X} \mapsto \{-1, 1\}$  à appliquer sur  $\mathcal{D}_n$  telle que le coût local  $L(t, \mathcal{D}_n)$  soit minimal

# 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  $t : \mathcal{X} \mapsto \{-1, 1\}$  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



# 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  $t : \mathcal{X} \mapsto \{-1, 1\}$  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

# 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  $t : \mathcal{X} \mapsto \{-1, 1\}$  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 retour en 3 avec  $\mathcal{D}_n \leftarrow \mathcal{D}_n^d$

# Algorithme récursif de construction

Cas d'un arbre binaire :

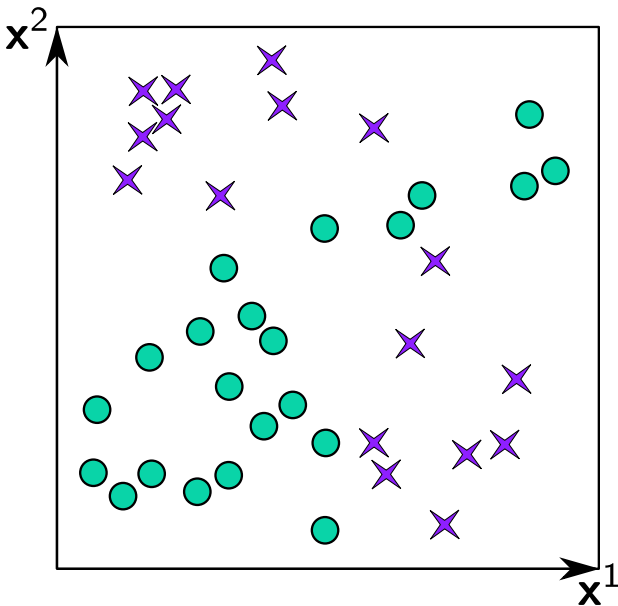
1. Soit  $\mathcal{D}_n$  l'ensemble d'apprentissage
2. Construire un nœud racine
3. Chercher la meilleure séparation  $t : \mathcal{X} \mapsto \{-1, 1\}$  à appliquer sur  $\mathcal{D}_n$  telle que le coût local  $L(t, \mathcal{D}_n)$  soit minimal
4. Associer le séparateur choisi au nœud courant et 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 retour en 3 avec  $\mathcal{D}_n \leftarrow \mathcal{D}_n^d$
7. Mesurer le critère d'arrêt à gauche, s'il est vérifié, le nœud gauche devient une feuille ; sinon retour en 3 avec  $\mathcal{D}_n \leftarrow \mathcal{D}_n^g$

# Algorithme récursif de construction

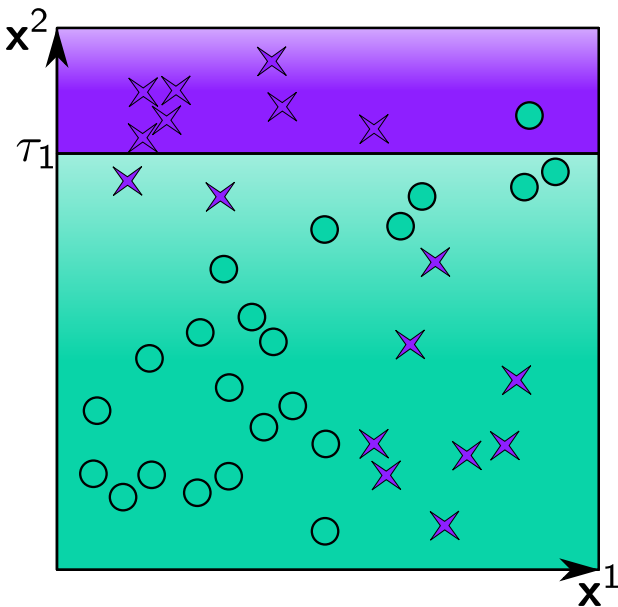
Cas d'un arbre binaire :

1. Soit  $\mathcal{D}_n$  l'ensemble d'apprentissage
2. Construire un nœud racine
3. Chercher la meilleure séparation  $t : \mathcal{X} \mapsto \{-1, 1\}$  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 retour en 3 avec  $\mathcal{D}_n \leftarrow \mathcal{D}_n^d$
7. Mesurer le critère d'arrêt à gauche, s'il est vérifié, le nœud gauche devient une feuille ; sinon retour en 3 avec  $\mathcal{D}_n \leftarrow \mathcal{D}_n^g$

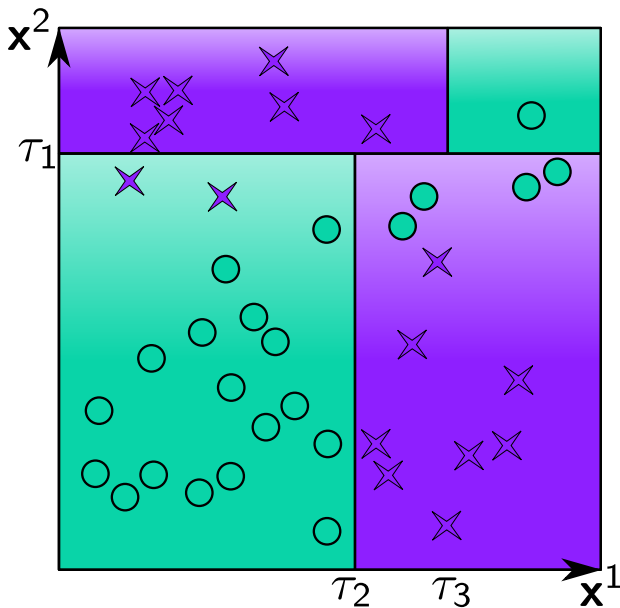
## Exemple visuel



## Exemple visuel



## Exemple visuel



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

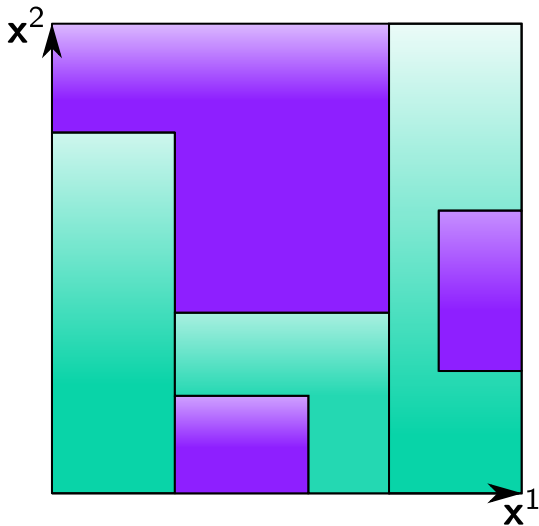
Tout comme les méthodes *stagewise/stepwise*/OMP en régression linéaire, l'algorithme CART (précédent) est **glouton**

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 pour le cas des méthodes gloutonnes pour un modèle de régression



## 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 modèles

# Probabilités / simplexe

Idée principale : définir une notion de pureté/impureté d'une coupure, et faire 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 :

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

Rem: on note le simplexe (de dimension  $K$ )

$$\Delta_K := \left\{ p \in \mathbb{R}^K : \sum_{k=1}^K p_k = 1 \text{ et } \forall k \in \llbracket 1, K \rrbracket, p_k \geq 0 \right\} \text{ ainsi,}$$

$$\hat{p}(\mathcal{D}_n) = (\hat{p}_1(\mathcal{D}_n), \dots, \hat{p}_K(\mathcal{D}_n))^{\top} \in \Delta_K$$

Rem:  $\Delta_K$  identifié aux probabilités discrètes ayant  $K$  modalités

# Coupure

- $\mathcal{D}_n$  : ensemble d'apprentissage
- $t_{j,\tau}$  : fonction de coupure

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

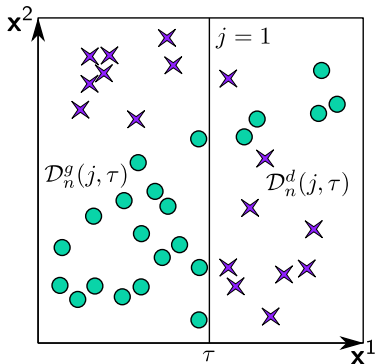
$$\mathcal{D}_n^g(j, \tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) \leq 0\} \quad (\text{partie gauche})$$

# Coupure

- ▶  $\mathcal{D}_n$  : ensemble d'apprentissage
- ▶  $t_{j,\tau}$  : fonction de coupure

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

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

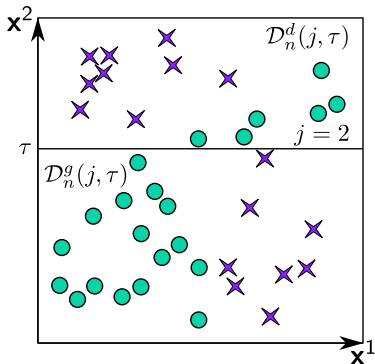
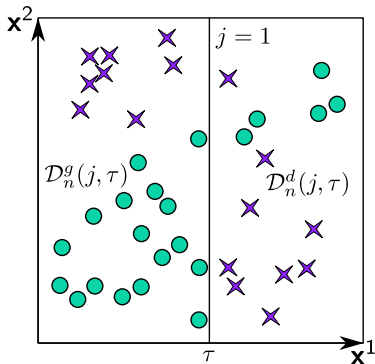


# Coupure

- ▶  $\mathcal{D}_n$  : ensemble d'apprentissage
- ▶  $t_{j,\tau}$  : fonction de coupure

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

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



# 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 :

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

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

$H$  est une fonction mesurant “l'**impureté**” d'une distribution

Propriétés requises :

- ▶ le coût total est la somme de l'impureté de chaque sous parties, pondérée par le nombre d'échantillons
- ▶ un nombre fini de seuils suffit sur l'apprentissage (au plus  $n$ )
- ▶ la notion l'impureté d'un échantillon  $\mathcal{D}_n$  est une fonction seulement la distribution des probabilités  $p(\mathcal{D}_n)$

# 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 modèles



# Fonction d'impureté

Rappel :  $\Delta_K := \left\{ p \in \mathbb{R}^K : \sum_{k=1}^K p_k = 1 \text{ et } \forall k \in \llbracket 1, K \rrbracket, p_k \geq 0 \right\}$

## 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  $p_{\text{unif}} = \left(\frac{1}{K}, \dots, \frac{1}{K}\right)^\top$
2.  $H$  atteint son minimum seulement au point  $(1, 0, \dots, 0)^\top, (0, 1, 0, \dots, 0)^\top, \dots, (0, \dots, 0, 1)^\top$
3.  $H$  est une fonction symétrique en  $p_1, \dots, p_K$

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

1. la distribution la plus impure est l'uniforme
2. les distributions les plus pures sont celles dégénérées
3. toutes les classes ont la même importance

# Critères de coût (I) : Erreur de classification

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

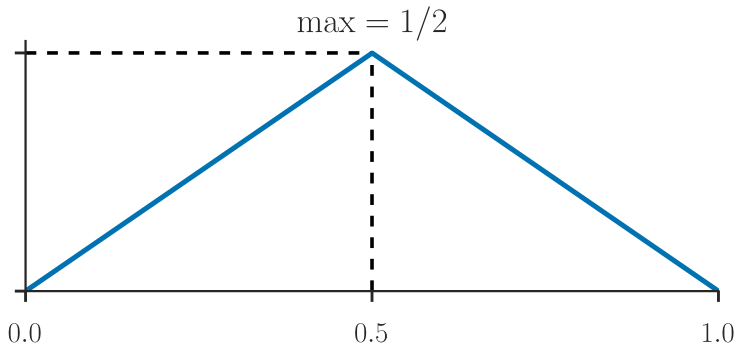
avec  $\hat{k}(\mathcal{D}_n)$  défini comme la classe majoritaire dans  $\mathcal{D}_n$  :

$$\begin{aligned}\hat{k}(\mathcal{D}_n) &= \arg \max_{k=1,\dots,K} \hat{p}_k(\mathcal{D}_n) \\ &= \arg \max_{k=1,\dots,K} \frac{1}{n} \sum_{i=1}^n \mathbb{1}(y_i = k)\end{aligned}$$

# Critères de coût (I) : Erreur de classification

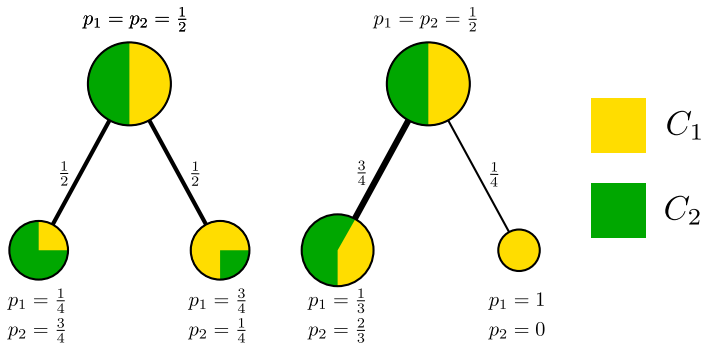
Application dans le cas binaire :

$$H_{\text{mis}}(\mathcal{D}_n) = 1 - \max_{k=1,2} \hat{p}_k(\mathcal{D}_n) = \min(\hat{p}_1(\mathcal{D}_n), 1 - \hat{p}_1(\mathcal{D}_n))$$



## Limites du choix : “erreur de classification”

- ▶ fonction non-différentiable (optimisations plus dure)
- ▶ pour une zone avec une classe très majoritaire il se peut qu’aucune coupure ne produise de réduction d’impureté
- ▶ 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  $p, p'$  dans  $\Delta_K$  avec  $p \neq p'$  et tout  $\alpha \in ]0, 1[$  on a :

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

Interprétation : mélanger ne fait qu'augmenter l'impureté

Conséquence : si  $H$  est une fonction d'impureté pure

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

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

cf. Breiman *et al.* (1984), page 100

## Critères de coût (II) : Entropie

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

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

Rem: liens étroits entre l'entropie de Shannon et celle de Boltzmann (thermodynamique)

---

**Exo:** entropie et divergence de Kullback-Leibler sont liées par  $H_{\text{ent}}(\mathcal{D}_n) = \log(K) - D_{\text{KL}}(\hat{p}(\mathcal{D}_n) \| p_{\text{unif}})$  en définissant pour toutes probabilités  $p, p' \in \Delta_K$  :

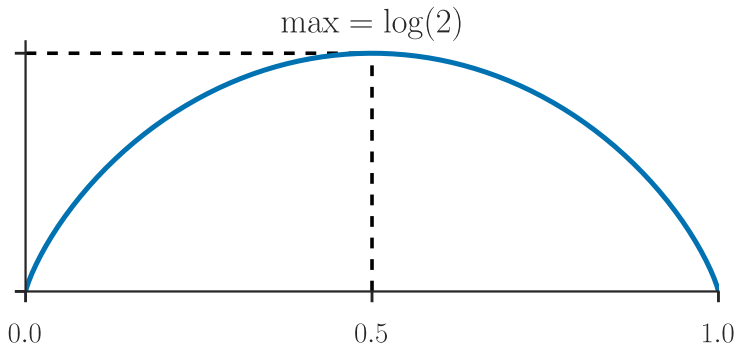
$$D_{\text{KL}}(p \| p') = \sum_{k=1}^K p_k \log \left( \frac{p_k}{p'_k} \right)$$

---

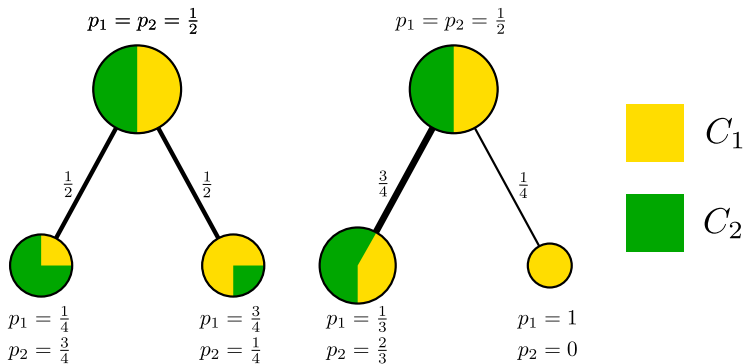
## Critères de coût (II) : Entropie

Application dans le cas binaire :

$$H_{\text{ent}}(\mathcal{D}_n) = -\hat{p}_1(\mathcal{D}_n) \log(\hat{p}_1(\mathcal{D}_n)) - (1 - \hat{p}_1(\mathcal{D}_n)) \log(1 - \hat{p}_1(\mathcal{D}_n))$$



## Retour sur un exemple



---

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

---



# Critères de coût (III) : indice de Gini

## Indice de Gini :

$$H_{\text{Gini}}(\mathcal{D}_n) = \sum_{k=1}^K \hat{p}_k(\mathcal{D}_n)(1 - \hat{p}_k(\mathcal{D}_n)) = \sum_{k=1}^K \sum_{\substack{k'=1 \\ k' \neq k}}^K \hat{p}_k(\mathcal{D}_n)\hat{p}_{k'}(\mathcal{D}_n)$$

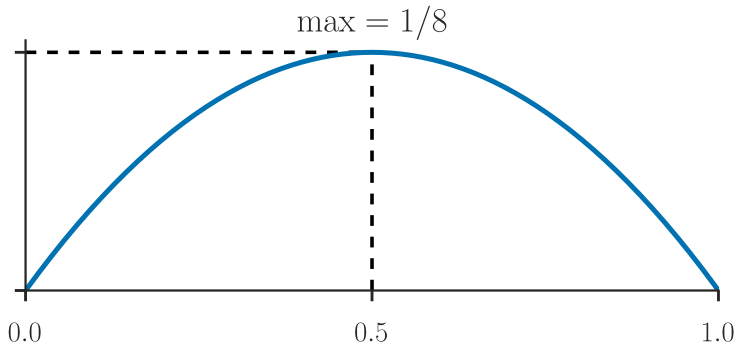
## Interprétation des deux formulations :

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

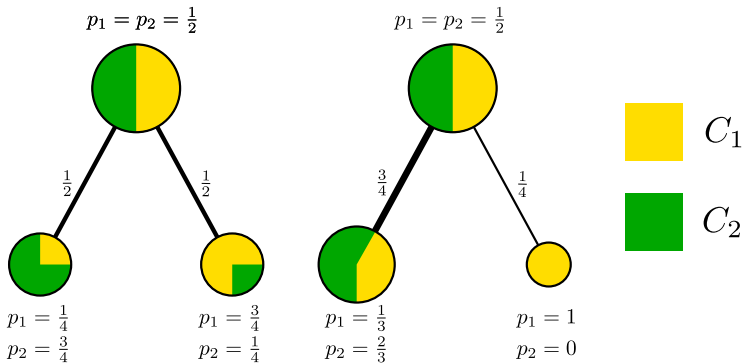
## Critères de coût (III) : indice de Gini

Application dans le cas binaire :

$$H_{\text{Gini}}(\mathcal{D}_n) = 2 \cdot \hat{p}_1(\mathcal{D}_n) (1 - \hat{p}_1(\mathcal{D}_n))$$



## 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 modèles

# Critères d'arrêt

On peut s'arrêter dans une branche dès qu'on atteint :

- ▶ une profondeur maximale
- ▶ un nombre maximale de feuilles
- ▶ un nombre trop faible d'exemples par nœud

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

Rem: le cas de profondeur maximale un est appelé "souche"  
( : *stump*)

# 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 !

## 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  $Y$  :

$$\begin{cases} L_{k,k'} = 0 & \text{si } k = k' \\ L_{k,k'} \geq 0 & \text{si } k \neq k' \end{cases}$$

Erreur moyenne en choisissant la classe  $k$  :

**Indice de Gini :** 
$$\sum_{k=1}^K \sum_{\substack{k'=1 \\ k \neq k'}}^K L_{k,k'} \hat{p}_k(\mathcal{D}_n) \hat{p}_{k'}(\mathcal{D}_n)$$

Rem: dans le cas binaire ( $K = 2$ ), une meilleure approche consiste à pondérer par  $L_{1,2}$  (resp.  $L_{2,1}$ ) selon que l'observation considérée vérifie  $y_i = 1$  (ou  $y_i = 2$ )

# Arbres de régression

Fonctionnement identique pour la régression, 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 veut maximiser l'homogénéité/pureté des sorties, ce qui revient à trouver la partition minimisant le risque quadratique



# 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 modèles

# Sélection de modèles (I)

(1) déterminer un des hyper-paramètres suivant par **validation croisée**

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

## Sélection de modèles (II)

(2) **par 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. Plus de détails dans [Hastie et al. \(2009\)](#)

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.  
<http://www-stat.stanford.edu/~tibs/ElemStatLearn/>.
- ▶ J. R. Quinlan.  
Induction of decision trees.  
*Maching Learning*, 1 :81–106, 1986.

## Références II

- ▶ S. Roman.  
*Coding and information theory*, volume 134 of *Graduate Texts in Mathematics*.  
Springer-Verlag, New York, 1992.
- ▶ C. Scott and R. D. Nowak.  
Minimax-optimal classification with dyadic decision trees.  
*IEEE Trans. Inf. Theory*, 52(4) :1335–1353, 2006.