SD 210 : Decision Trees

Pietro Gori, Joseph Salmon, Florence d'Alché-Buc
Télécom Paristech

Plan

Summary

Supervised Classification and regression

X : input data x_i^j, random variable in $\mathcal{X} = \mathbb{R}^p$ with $i = 1, \ldots, n$ and $j = 1, \ldots, p$ where n and p are the number of observations and variables respectively
Y : response (to predict), random variable in $\mathcal{Y} = \{C_1, \ldots, C_K\}$ (classification with K classes) or $\mathcal{Y} = \mathbb{R}$ (regression)
P : joint probability distribution of (X, Y), fixed but **unknown**
$\mathcal{D}_n = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$: i.i.d.samples drawn from P
$\mathcal{H}$: collection of classifiers $h \in \mathcal{H}$
ℓ : loss function which measures the error of the classifier/model

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

<u>**Goal**</u> : estimate from $\mathcal{D}_n$ the function $h \in \mathcal{H}$ which minimizes the risk (cost) function $R(h) = \mathbb{E}_P[\ell(X, Y, h(X)]$

Estimate a classifier

We need to define :

- **input and output data space** ($\mathcal{X}$ $\mathcal{Y}$)
- **type of classifier** ($\mathcal{H}$)

Estimate a classifier

We need to define :

- **input and output data space** ($\mathcal{X}$ $\mathcal{Y}$)
- **type of classifier** ($\mathcal{H}$)
- **cost function** (ℓ) to minimize for finding the best h

Estimate a classifier

We need to define :

- **input and output data space** ($\mathcal{X}$ $\mathcal{Y}$)
- **type of classifier** ($\mathcal{H}$)
- **cost function** (ℓ) to minimize for finding the best h
- **minimization algorithm** for ℓ

Estimate a classifier

We need to define :

- **input and output data space** $(\mathcal{X}\ \mathcal{Y})$
- **type of classifier** $(\mathcal{H})$
- **cost function** (ℓ) to minimize for finding the best h
- **minimization algorithm** for ℓ
- **method for model selection** to estimate hyper-parameters

Estimate a classifier

We need to define :

- **input and output data space** ($\mathcal{X}$ $\mathcal{Y}$)
- **type of classifier** ($\mathcal{H}$)
- **cost function** (ℓ) to minimize for finding the best h
- **minimization algorithm** for ℓ
- **method for model selection** to estimate hyper-parameters
- **method to evaluate performance**

Estimate a classifier

We need to define :

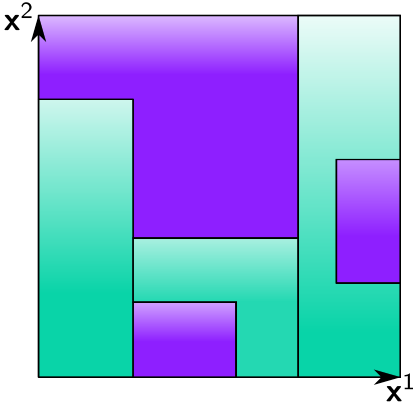
- **input and output data space** $(\mathcal{X}\ \mathcal{Y})$
- **type of classifier** $(\mathcal{H})$
- **cost function** (ℓ) to minimize for finding the best h
- **minimization algorithm** for ℓ
- **method for model selection** to estimate hyper-parameters
- **method to evaluate performance**

Summary

Constant piecewise model/classifier

$\mathcal{H}$ belongs to the set of **constant piecewise functions**.

We divide the input space $\mathcal{X}$ in M disjoint partitions $\mathcal{C}_m$ with $m = 1, \ldots, M$. To simplify things, we assume that the separation lines are parallel to the coordinate axes :



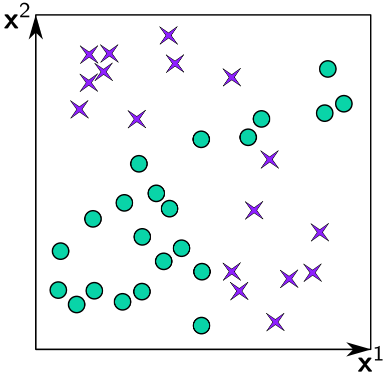
Constant piecewise model/classifier

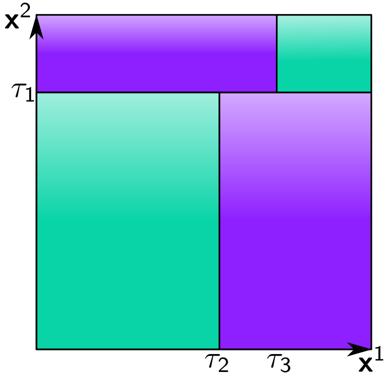
- Motivation : easy to interpret
- Limitations :
  - regions are difficult to describe
  - If the partition is fixed beforehand, many regions might end up being empty
- Possible solution : learn the partitions from the data ! How to avoid curse of dimensionality ?

Summary

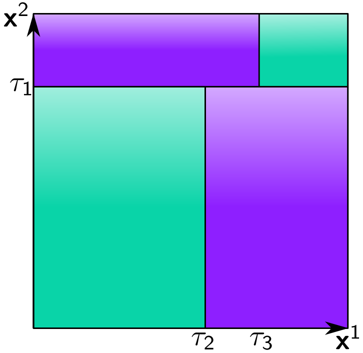
Decision trees

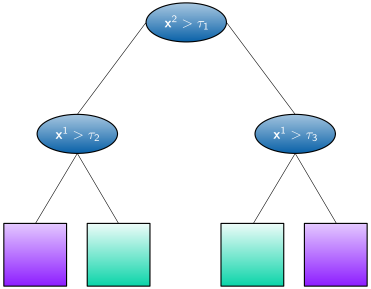
Presented almost simultaneously between 1979 and 1983 by Breiman *et al.* (1984) (CART, Berkeley, USA) and Quinlan (1986) (ID3, Sydney, Australie) in two different communities : statistics (CART), and *machine learning* (ID3)





Decision trees

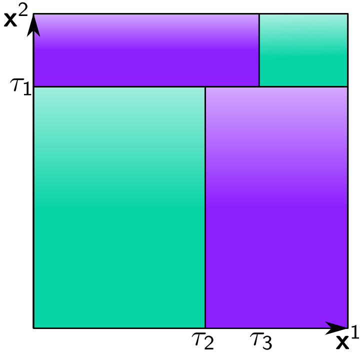


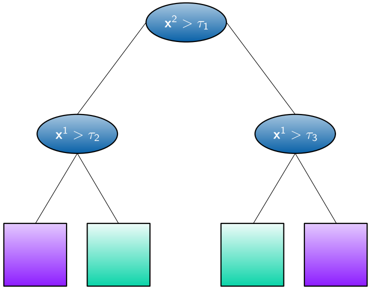


First idea :

Use several separation lines (and not only one) to build non linear decision boundaries.

Decision trees

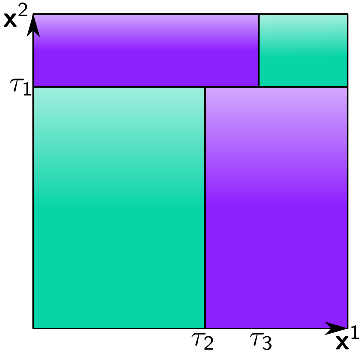


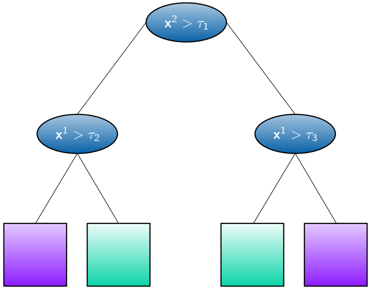


Secod idea :

Use separation lines orthogonal to the coordinate axis, *i.e.*, hyperplanes $\{\mathbf{x} \in \mathcal{X} : \mathbf{x}^j = \tau\}$ to ease interpretation.

Decision trees





Third idea :

Use binary decision trees : The full data-set sits at the top of the tree. Every node (junction) is associated to a separating hyperplane $\{\mathbf{x} \in \mathcal{X} : \mathbf{x}^j = \tau\}$. The terminal nodes correspond to the regions.

Summary

Linear partition orthogonal to the axes

Let $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^p)$ with p variables. We define :

- Continuous or binary variable $\mathbf{x}^j$ and threshold τ :

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

- Categorical variable with M categories $\{v_1^j, \ldots, v_M^j\}$:

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

Summary

Efficient recursive algorithm

For a binary tree :

1. Given $\mathcal{D}_n$
2. The entire data-set is the root node
3. Look for the best separator $t_{j,\tau}$ on $\mathcal{D}_n$ such that the local cost function $L(t, \mathcal{D}_n)$ is minimal. This means looking for the "best" direction j and threshold τ.
4. Split $\mathcal{D}_n$ to $\mathcal{D}_n^d$ and $\mathcal{D}_n^g$ using the estimated separator.
5. It results two nodes, a left ($\mathcal{D}_n^g$) and a right ($\mathcal{D}_n^d$) one
6. Evaluate the stopping criteria for the right node, if it is verified, the nodes becomes a terminal node, otherwise go to 3 using $\mathcal{D}_n^d$ as input space
7. Evaluate the stopping criteria for the left node, if it is verified, the nodes becomes a terminal node, otherwise go to 3 using $\mathcal{D}_n^g$ as input space

Examples

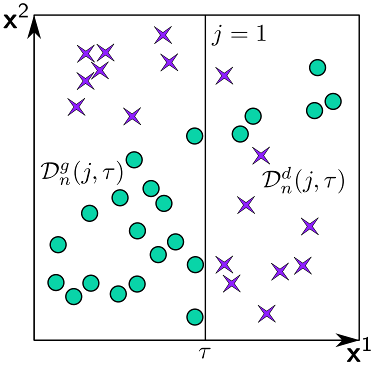
Given the input data-set $\mathcal{D}_n$ and a binary separator $t_{j,\tau}$, we have

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

Examples

Given the input data-set $\mathcal{D}_n$ and a binary separator $t_{j,\tau}$, we have

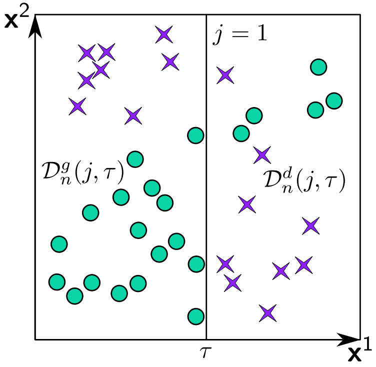
$$\mathcal{D}_n^d(j,\tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) > 0\}$$
$$\mathcal{D}_n^g(j,\tau) = \{(\mathbf{x}, y) \in \mathcal{D}_n, t_{j,\tau}(\mathbf{x}) \leq 0\}$$

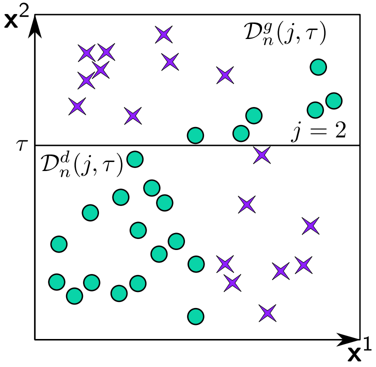


Examples

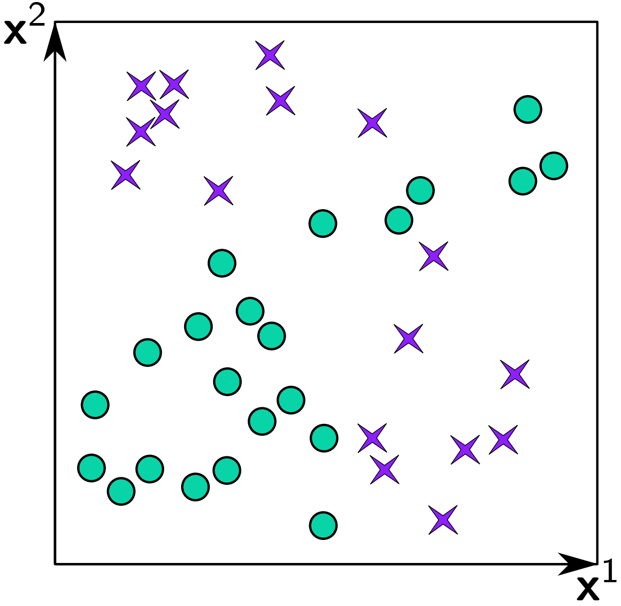
Given the input data-set $\mathcal{D}_n$ and a binary separator $t_{j,\tau}$, we have

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

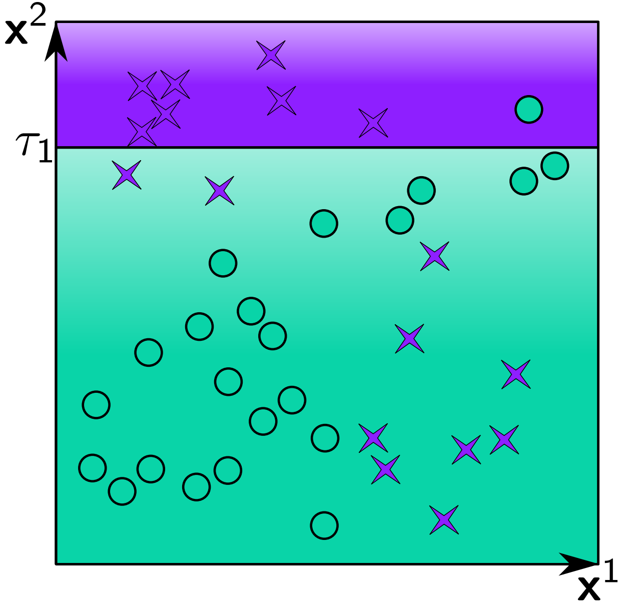




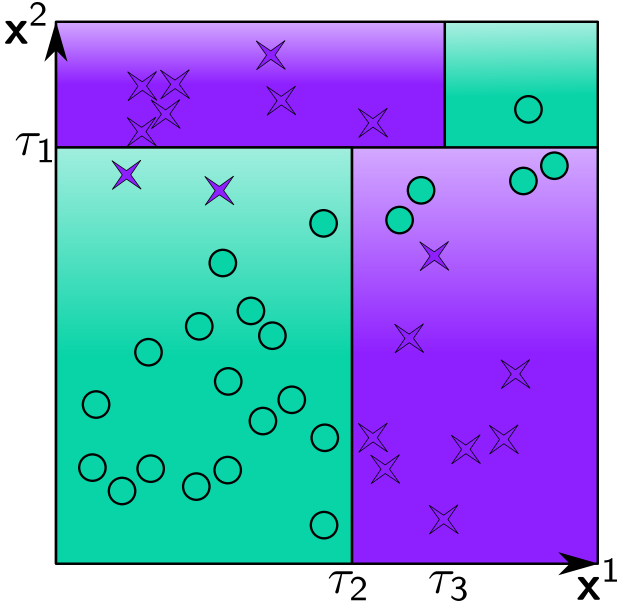
Example



Example



Example



Greedy method

The presented algorithm (called CART) is greedy as the methods *stagewise*/*stepwise*/OMP in linear regression.

We do not optimize a global criteria (course of dimensionality !). Instead, we locally look for an optimal separator (with respect to L), which means at every direction j independently.

cf. MDI 720 / SD204 for greedy algorithms in linear regression

Summary

Local cost function

Now we need to define an impurity measure for splitting nodes.

Given the input data $\mathcal{D}_n$ divided in K classes, we define the proportion of observations belonging to class k (*i.e.*, categorical distribution) as :

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

Note that $\rho(\mathcal{D}_n) = (\rho_1(\mathcal{D}_n), \ldots, \rho_K(\mathcal{D}_n))^\top \in \Delta_{K-1}$ where $\Delta_{K-1} := \left\{ \rho_k \in \mathbb{R}^K : \sum_{k=1}^{K} \rho_k = 1 \text{ and } \forall k \in [\![1, K]\!], \rho_k \geq 0 \right\}$ is the (K-1)-simplex.

Local cost function

Among all parameters $(j, \tau) \in \{1, \ldots, p\} \times \{\tau_1, \ldots, \tau_m\}$, we look for $\hat{j}$ and $\hat{\tau}$ which minimizes the following cost function :

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

- H is an “impurity” function that evaluate the splitting and it depends on the categorical distributions of each class. Pure means a node with observations from the same class.
- the total cost is the sum of the impurity of each child node ($\mathcal{D}_n^g$ and $\mathcal{D}_n^d$) weighted by the proportion of its observations (n_g and n_d)
- we evaluate a finite number of thresholds (max n)

Summary

Impurity function

Definition : impurity function

An **impurity** function $H : [0:1]^K \to \mathbb{R}$ is a function defined on Δ_{K-1} for which the following properties hold :

1. H becomes maximum at points $(\frac{1}{K}, \dots, \frac{1}{K})^\top$, *i.e.*, all ρ_k are equal
2. H becomes minimum at points $(1, 0, \dots, 0)^\top, (0, 1, 0, \dots, 0)^\top, \dots, (0, \dots, 0, 1)^\top$, *i.e.*, the probability of being in a certain class is 1 and 0 for all other classes. These are the vertices of Δ_{K-1}.
3. H is symmetric with respect to its arguments $\rho_1, \dots, \rho_K$, *i.e.*, even if we permute the classes H does not change, that is to say that all classes have the same importance

Impurity function : binary case ($K = 2$)

When $K = 2$:

- Δ_{K-1} is the line segment joining $(1, 0)$ and $(0, 1)$ in $\mathbb{R}^2$
- H becomes maximum at $(\frac{1}{2}, \frac{1}{2})$
- H becomes minimum at $(0, 1)$ or $(1, 0)$

Misclassification error

Given the data of a node $\mathcal{D}_n$ (it might be the root node or a child node), we assign the observations in $\mathcal{D}_n$ to the majority class $\hat{k}$:

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

Then we define :

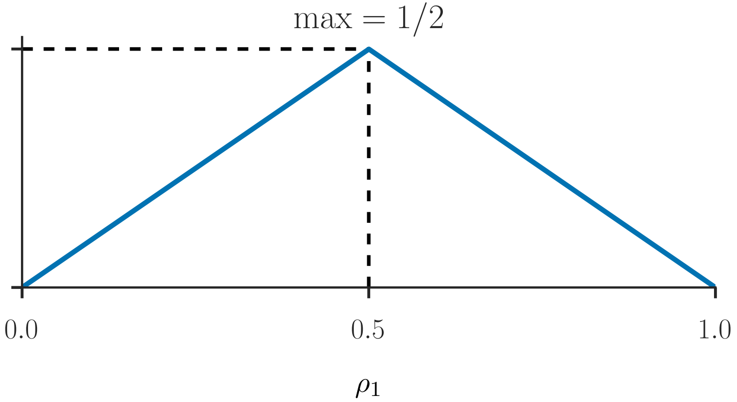
Misclassification error : $\quad H_{\text{mis}}(\mathcal{D}_n) = 1 - \rho_{\hat{k}}(\mathcal{D}_n),$

This is the error that we commit by assigning the observations to the class $\hat{k}$. Remember that $\sum_{k=1}^{K} \rho_k = 1$.

Misclassification error

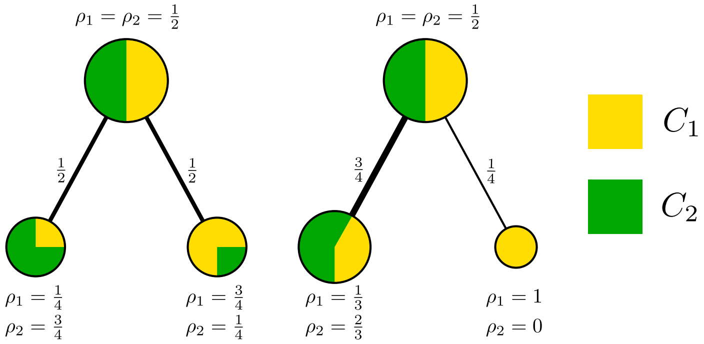
When the number of classes K is 2

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



Limitations of the misclassification error

- for a partition where a class has a clear majority, we might not find a split which reduces L
- The function is not differentiable (optimization is harder)
- It might underestimate pure nodes :



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

Strict impurity function

Definition : strict impurity function

Let $H : [0:1]^K \to \mathbb{R}$ be an impurity function, ρ, ρ' two distributions in Δ_{K-1} with $\rho \neq \rho'$ and $\alpha \in]0,1[$. Then H is called **strict**, if it is strictly concave :

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

If H is strict then it follows that

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

the equality is given iff $\rho_k(\mathcal{D}_n) = \rho_k(\mathcal{D}_n^g) = \rho_k(\mathcal{D}_n^d)$ for all k

Remark : The impurity function of the misclassification error is concave, but it is not strictly concave. L might be equal for all possible splittings.

Entropy

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

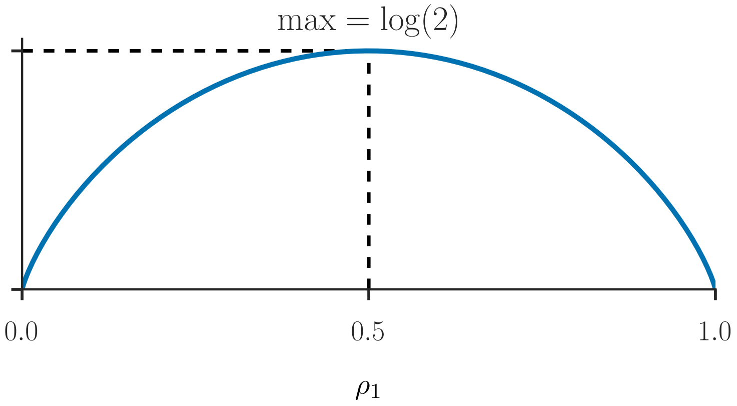
- if we use $\log_2$, it is called Shannon entropy
- $-\log \rho(\mathcal{D}_n)$ is the information content of $\mathcal{D}_n$
- Entropy is defined as the expected value of the information content (average amount of information). It measures the randomness
- when an event is certain, entropy is 0
- information gain is defined as reduction in entropy
- it is differentiable

For more information see *cf.* Roman (1992), Chapter 1

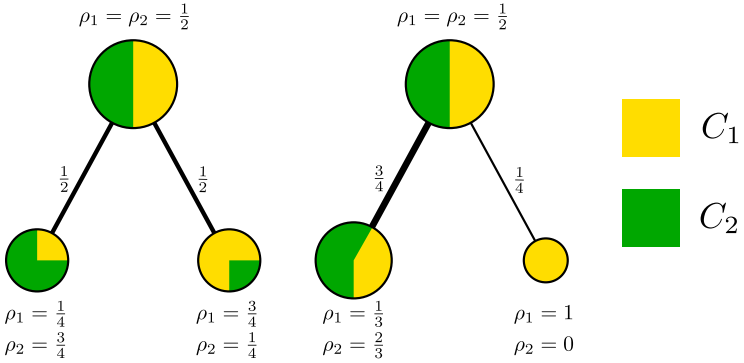
Entropy

When the number of classes K is 2

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



Example



<ins>Question</ins>: Compute L_{ent} associated to H_{ent}. Which split is better ?

Gini index

Gini index :

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

Two different interpretations :

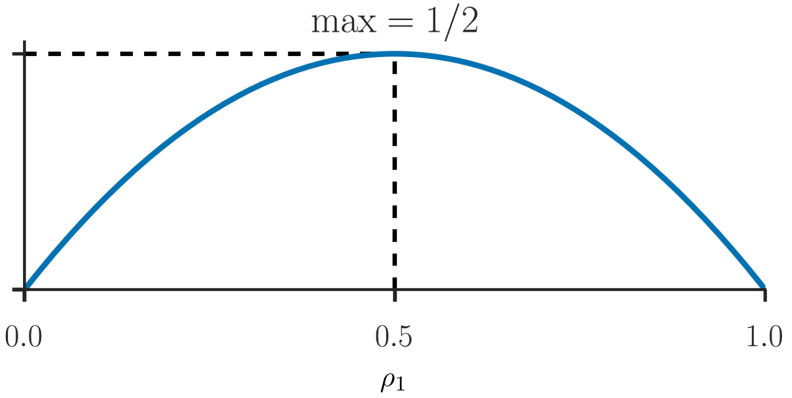
- If we code each observation as 1 for class k and zero otherwise (bernoulli variable) the variance is $\rho_k(\mathcal{D}_n)(1 - \rho_k(\mathcal{D}_n))$. The Gini index is the sum of the variances of the “binarized” classes
- We do not assign observations to the majority class (as for H_{mis}) but we classify them to class k with probability $\rho_k(\mathcal{D}_n)$. The training error rate of this rule is $\sum_{\substack{k'=1 \\ k' \neq k}}^{K} \rho_k(\mathcal{D}_n)\rho_{k'}(\mathcal{D}_n)$.

  The Gini index is the sum over all classes.

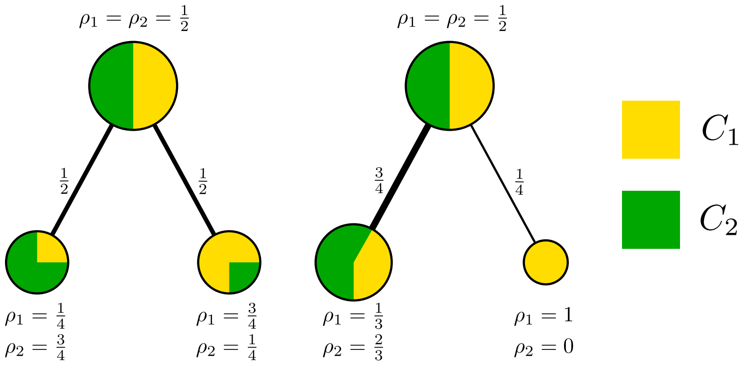
Gini index

When the number of classes K is 2

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



Example



<u>Question</u>: Compute L_{Gini} associated to H_{Gini}. Which split is better ?

Summary

Stopping criteria

Without stopping criteria, we could grow a tree until a situation where each observation represents a terminal node. This would be computationally expensive, difficult to interpret and prone to over-fitting. Instead, we could use one or more of the following stopping criteria :

- maximal depth
- maximum number of terminal nodes
- a node becomes a terminal node when it reaches a maximum number of observations
- degree of purity of a node (*i.e.*, threshold on $\rho_k(\mathcal{D}_n)$)

Categorical variables

- For a binary tree : if we have a categorical variable x which can take up to M values, we transform it into M binary variables
- Warning : The partitioning algorithm tends to favor categorical variables with many values since the number of possible partitions grows exponentially with M. This means that we have more choices to find a good partition. This can lead to over-fitting ! Try to avoid such variables.

Loss matrix

In some cases, the consequences of misclassifying observations can be very serious (*i.e.,* medicine). To account for that, we introduce a loss matrix $C \in \mathbb{R}^{K \times K}$, with $C_{k,k'}$ being the loss incurred for classifying observations of class k as belonging to class k'

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

We can then modify the Gini index as follows :

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

Note: This works for $K > 2$ but it has no effect in the binary case (Why ?). A different approach consists of weighting the observations of class k by $C_{k,k'}$. In a terminal node we classify the observations to $k' = \arg\min_k' \sum_k C_{k,k'} \rho_k(\mathcal{D}_n)$

Regression trees

For regression the process is almost identical, we only change the impurity function. We use the squared error (or variance) :

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

where

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

and we minimize as before

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

Note: as before we want to maximize the homogeneity (purity) of the terminal nodes

Summary

Model selection (1)

We can compute one (or more) of the following hyper-parameters instead than fixing them as stopping criteria :

- maximal depth of the tree
- maximum number of terminal nodes
- maximum number of observations in a node to become a terminal node

$\rightarrow$ we could use **cross validation**

Pruning (2)

What's the optimal size of a tree ? A large tree might overfit the data, while a small tree might not capture important structures.

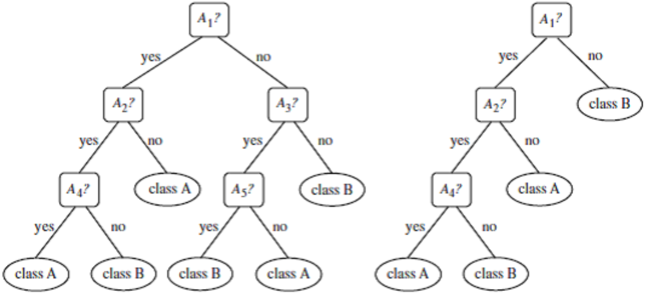
A possible solution is to grow a large tree in a training set stopping the splitting only when the terminal nodes have reached a minimum number of observations or a certain purity. Then, we produce several new trees by pruning the original tree at different nodes. When we want to prune with respect to node t, we delete all successor nodes of t in the original tree.

The new trees are then tested in a validation set. We select the tree that gives the best performance.

Trying all possible trees might be computationally unfeasible. Several greedy techniques exist. See Hastie *et al.* (2009) and Scott et Nowak (2006) for more details.

Note: pruning is not currently supported in `sklearn` (use `rpart` in `R` if needed)

Example of pruning



Advantages and drawbacks of decision trees

Advantages

- Build a non-linear and intepretable decision function
- Invariant under scaling and other linear transformations of the input data X
- Robust to the inclusion of (few) irrelevant features x
- It works for multi-class
- Computationally efficient : $O(\log F)$, where F is the number of terminal nodes
- It works for continuous and categorical variables

Advantages and drawbacks of decision trees

Drawbacks

- Low bias but very high variance. A small change in the input data can bring to a completely different tree ! This instability is due to the hierarchical nature of the process. $\rightarrow$ averaging trees reduces the variance (bagging, random forests)
- No global optimization
- Lack of smoothness of the prediction surface. It can degrade performance in regression.

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.
  Mach. Learn., 1(1) :81–106, March 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 Transactions on Information Theory, 52(4) :1335–1353, 2006.