

Ensemble learning from theory to practice

# Lecture 1:

## Part 1: Introduction - Part 2: Decision Trees

Myriam Tami

[myriam.tami@centralesupelec.fr](mailto:myriam.tami@centralesupelec.fr)

January - March, 2022



CentraleSupélec

## About me

- Undergrad at the University of Montpellier (UM), France
- Ph.D. in IMAG (Institut Montpelliérain Alexander Grothendieck) lab at UM, France
- Postdoc researcher in LIG (Laboratoire Informatique de Grenoble) lab at University of Grenoble Alpes, France
- Associate Professor Maître de Conférences (MCF) at CentraleSupélec & MICS (Mathematics Interacting with Computer Science) lab since February 2019

**Research interests:** ML, Ensemble methods, Domain adaptation, Few-supervised ML methods, Heterogeneous data, Multi-modal learning, Causality.

## Contact:

Email: myriam.tami@centralesupelec.fr

Office: Bouygues, MICS Lab, Room SC.116

## Course organization

- **9 class hours**
- **12 practical class hours** (practical exercises, labs using Python, 3 hours for the Python project)
- **3 hours for each group oral defense** of an evaluated Python project
- Only one instructor:
  - ▶ Instructor of the group 1 & group 2: Myriam Tami
  - ▶ Office hours: send me an email and we will find a good time to meet
  - ▶ **Best way to get a quick answer: by email (answering delay: 3-7 days)**
- **Evaluation:** project using python (with code → GIT, report delivery and oral defense), an individual assignment (if pandemic complications → a quiz).

## Teaching tools

- **Edunao:**

- ▶ More details about the course and the organization (timeline, etc.)
- ▶ Teaching material (slides, Jupyter Notebooks, PDF with practical exercices, corrections.)
- ▶ Resources (books, papers)
- ▶ Quiz sessions
- ▶ Submit your work

- **Microsoft Teams:**

- ▶ No Teams team created
- ▶ If the pandemic situation requires it, information will be communicated to you later

Context  
oo  
oooooooo

Basic idea and reminders  
oooooooooooo  
ooo

Decision tree model and building  
oooooooo  
oooooooooooo  
oooooooo

Discussion  
ooo

References  
o

## Teaching tools: Edunao



Figure: Edunao screenshot

## Content & Timeline (Subject to Change)

The screenshot shows a course page on the Edunao platform. The sidebar on the left contains a navigation menu with the following items:

- Ensemble Learning (2021-2022)** (highlighted)
- Participants
- Badges
- Compétences** (highlighted)
- Notes
- Tableau de bord
- Accueil du site
- Calendrier
- Fichiers personnels
- Banque de contenus
- Tableau de bord de l'enseignant
- Mes cours
- 2048 et autres jeux
- 2048 et autres jeux (2021/2022)

The main content area is titled "Course". It includes sections for "Structure", "Content", "Course Timeline", and "Prerequisites".

**Structure:**  
A session (3 hours) of the course will comprise either a lecture or a lab, including practical exercises.

**Content:**

1. Introduction and Decision Trees
2. Bagging
3. Boosting
4. --In progress--

**Course Timeline:**

- 3rd january: **Introduction and Decision Trees** (3h)
- 14th january: **Deadline** for submitting a project proposal on edunao ([template](#))
- 14th january: **Bagging and Quizz** on the previous courses
- 31th january: **Boosting and Presentations + vote to select the best project proposal** (1 bonus point for the winner)
- 11 march: **Deadline** for submitting your final report and git link containing your Python code.
- 14th march: **Project Evaluation (oral restitution and code submission)** - TD1 Group (note: final report can be submitted on 23rd march)
- 18th march: **Project Evaluation (oral restitution), code submission and final report submission** - TD2 Group

**Prerequisites:**

Figure: Edunao screenshot - Course part and the course timeline

## Coursework and grading (subject to change)

Wght	Details
Assignment 1 (individual)  Python project (teams of 4-5 students)	25%  Theoretical questions or exercises; quiz on edunao (it will be held remotely); planed for lecture 2 or 3  <b>Part1: Project proposal</b> (to submit on edunao before lecture 2 (14th january); a vote to select the best project subject planed for 31st january; final report (PDF) and code (GIT link) to submit on edunao before the 11th march; <b>oral defense in class planed for the 14th march (group 1) and 18th march (group 2)</b>
	<b>Part 2:</b> Each group has to chose between <b>option a.</b> "Implement a Decision Tree from scratch on Python" or <b>option b.</b> "Select a recent research paper on Ensemble Learning area and, after the teacher validation of the paper (before 31st january), do a summary of the authors contributions and an implementation of their experiments"

## Previous project (2020)

### Text Classification with Rakuten France Product Data

- Topic of large-scale product type code text classification
- This project is derived from a data challenge (details available in this [link](#))
- Goal: predict each product's type code as defined in the catalog of Rakuten France
- The above data challenge focuses on multimodal product type code classification using text and image data

## Previous project (2020)

Tasks for the Rakuten project:

- After a preprocessing step, build a TF-IDF matrix that will constitute the data set
- The project guidelines was:
  - ▶ Apply all adapted approaches of the course and handled during labs (Decision Trees, Bagging, Random forests, Boosting, Gradient Boosted Trees, AdaBoost, etc.) on this data set in the goal to predict the target variable (product's type code).
  - ▶ Compare their performances thanks all scores you can output.
  - ▶ Conclude about the most appropriate approach on this data set for the predictive task.
  - ▶ Write a report in .tex format that address all these guidelines. We will take into account the quality of writing and presentation of the report.
  - ▶ **Provide a GIT link of your team Python code and allow to show your work progress.**

## Project of this year

**I propose to you to submit the project subject you are interested to work on it!**

The guidelines:

- Take inspiration from the previous project or any datachallenge from the web
- Try to propose a project i.e.: an ML task on a reasonable dataset: a huge dataset could be too hard to work on it (too time consuming)
- The project guidelines stay the same (apply all approaches of the course, compare and present the results and performances, conclude, write a report, provide the GIT link of your code...)

## Prerequisites in ML

Given an ML problem,

- Decide as what type of ML problem you can formalize it:
  - ▶ Supervised? Unsupervised?
  - ▶ Regression? Classification? Clustering?
  - ▶ Dimensionality reduction?
- Describe it formally in terms of
  - ▶ A design matrix  $X$  ( $n, d$ ) dim
  - ▶ A target variable/matrix  $Y$
  - ▶ A loss function to optimize
- Apply the ML pipeline (data preprocessing, model selection, evaluation, make prediction; etc.)
- Programming (Python, scikit-learn, numpy, pandas, scipy, etc)
- Deal with real-world data challenges

## Prerequisites in Decision Trees (DT)

### DT (based classification) building

- A simple, greedy, recursive approach (builds up tree node-by-node)
  - ▶ Split a set of data points based on an attribute test condition that optimizes a specific criterion
  - ▶ How to determine the best split (i.e., the best attribute test condition and the best split value)
  - ▶ Stopping criteria for DT growing and overfitting problem
- Know some measures of node impurity (e.g., Gini index)
- Read a DT visualization (diagram)

## Learning objectives of the course

Given an ML problem,

- Know if you can use an ensemble method to solve it
- Know as what type of ensemble methods can correspond it
- Can formalize it by using a model from ensemble methods models
- Can apply/learn/tune different models from ensemble methods
- For each of them, can make an evaluation and comparison to choose the more appropriate model
- Know the pros and cons of each ensemble method or tree-based approach
- Understand and know the theoretical process/algorithms of each ensemble method
- Can implement each of them from scratch (option a.) or by using Python libraries
- Be able to read, understand and restitute the content of a research paper (option b.)

## Learning objectives of the today's class

- Reminders (ML, DT) and introduce our ML notations, formalizations
- Present DT building process:
  - ▶ mathematical rules for regression AND classification cases
  - ▶ the algorithms
- Be able to formalize and understand in details the DT building process
- Know more about the DT limitations
- Be able to implement DT algorithms by yourself (by using scikit-learn library or from scratch):
  - ▶ know how to train and visualize a DT
  - ▶ learn how to predict with a DT and how to regularize it
  - ▶ know how to use DT for regression/classification tasks

## 1 Context & motivations

- Machine learning field and weak learners
- Decision Tree: a supervised weak learner

## 2 Basic idea and reminders

- Example of basic decision tree
- Vocabulary and conventions

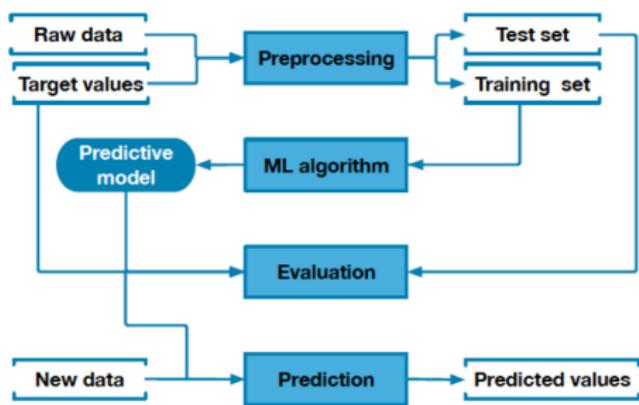
## 3 Decision tree model and building

- Decision tree model
- Building regression trees
- Building classification trees

## 4 Discussion

## Traditional ML process,

- You have a data set (both input and target)
- You have to achieve a task (i.e., classification, regression)
- To tackle it: you learn a predictor (a predictive model)!

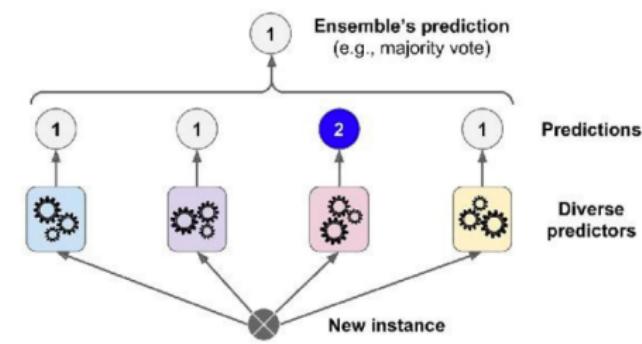


But, **What happen if you obtain a weak learner?**

meaning it does only slightly better than random guessing

Together we stand but divided we fall!

- Ensemble learning principle: **combine multiple weak learners in order to improve robustness and prediction performances**



**Figure:** source: Hands-On Machine Learning with Scikit-Learn and TensorFlow, A. Géron

- A decision tree is a weak learner
- Intuition: an aggregation of weak Decision Trees (DT) leads to a strong random forest

Before, we have to come back to DT in details

## Introduction and history

- Decision tree method was introduced by Leo Breiman in 1984
- There are many algorithms:
  - ▶ CART (Classification And Regression Trees)  
[Breiman et al., 1984]
  - ▶ ID3, C4.5
  - ▶ Hunt
- Python library:  
`scikit-learn`  
(`sklearn` module)



## Introduction and notations

- A decision tree is a weak **supervised** learner
- **Goal:** predict a dependent variable  $y \in \mathcal{Y}$  explained by input variables  $x \in \mathcal{X}$  via a set of data  $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$  iid

### Related goals:

Construct a predictor  $\hat{f}(x)$  of a function  $f$  mapping  $x$  to  $y$

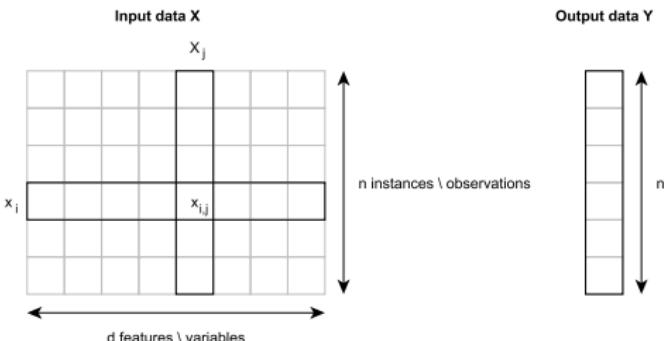
$$f: \mathcal{X} \rightarrow \mathcal{Y}$$

$$x \mapsto y$$

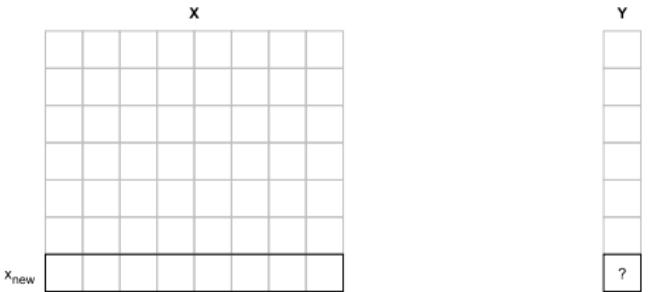
- for supervised **regression**:  $\mathcal{Y} = \mathbb{R}$  and  $\mathcal{X} = \mathbb{R}^d$
- for supervised **classification**:  $\mathcal{Y} = \{1, \dots, C\}$  and  $\mathcal{X} = \mathbb{R}^d$   
where  $C$  is the number of classes and  $d$  the dimension of the input space  $\mathcal{X}$

## Supervised learning problematic

- The data



- Related problematic:  
For a new observation  $x_{new}$  I want to predict the linked unobserved  $y$



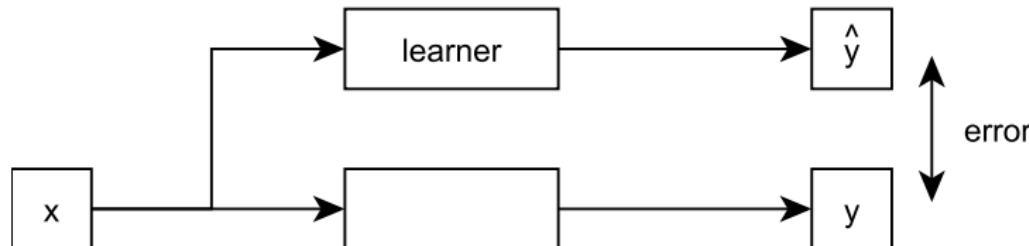
## Example of supervised learning problematic



## Problem: Is your date good or evil?

	mask	cape	tie	ears	smokes	height	class
<b>Batman</b>	y	y	n	y	n	180	good
<b>Robin</b>	y	y	n	n	n	176	good
<b>Alfred</b>	n	n	y	n	n	185	good
<b>Penguin</b>	n	n	y	n	y	140	evil
<b>Catwoman</b>	y	n	n	y	n	170	evil
<b>Joker</b>	n	n	n	n	n	179	evil
<b>Batgirl</b>	y	y	n	y	n	165	?
<b>Riddler</b>	y	n	n	n	n	182	?
<b>Someone New</b>	n	y	y	y	y	181	?

## Supervised learning



- ➊ Set of couples  $(x_i, y_i)$ ,  $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$  learning set
- ➋  $x_i \in \mathcal{X}$  e.g.,  $\mathcal{X} = \mathbb{R}^d$  and  $x_i$  a  $d$ -length vector  
 $y_i \in \mathcal{Y}$  e.g.,  $\mathcal{Y} = \mathbb{R}$  if  $y$  a temperature feature or  $\mathcal{Y} = \{0, 1\}$  if  $y$  encodes the class **evil** v.s **good**
- ➌ The learner select among a given set of functions  $\mathcal{F}$ , the best  $f$  such as  $\hat{y} = f(x)$

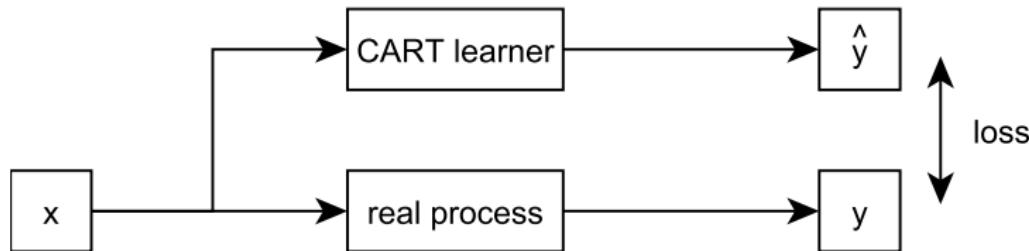
→ What is  $\mathcal{F}$ ? How to select  $f$ ?

## Supervised learning

- The set  $\mathcal{F}$ , called **the hypothesis space**:
  - ▶ depends on the learning method used
    - ex: linear regression,  $f(x) = ax + b$  with  $a, b$  the parameters of  $f$
  - ▶ learn  $f$  means learn the parameters of the kind of function chosen
- The best  $f$  to select:
  - ▶ need of a tool that could measure the quality of the learned  $f$
  - ▶ **objective function, loss function, risk function** (expectation of the loss)
  - ▶ risk function allows to **measure the error** made when we use the learned  $f$
- **During the learning process** we use at each step a loss function:  
we want to **learn  $f$  under the constraint of a minimal loss**

## Decision Tree: a supervised learner

- Decision tree is a supervised learning method
- It is also called CART (Classification And Regression Trees)



- To characterize Decision Tree method we have to answer,
  - ▶ What is  $\mathcal{F}$ ?
  - ▶ How to learn the best  $f$  from  $\mathcal{F}$ ?

Let's start by an illustration: a basic example

# Plan

## 1 Context & motivations

- Machine learning field and weak learners
- Decision Tree: a supervised weak learner

## 2 Basic idea and reminders

- Example of basic decision tree
- Vocabulary and conventions

## 3 Decision tree model and building

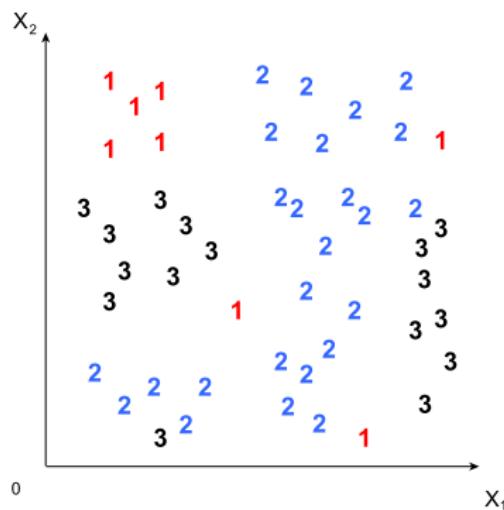
- Decision tree model
- Building regression trees
- Building classification trees

## 4 Discussion

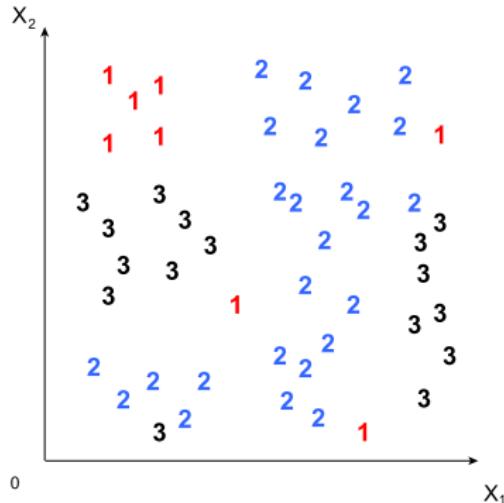
## Example of a basic DT: case of a basic classification

- Learning set

$\mathcal{D} = \{(x_{1,1}, x_{1,2}), y_1\}, \dots, ((x_{n,1}, x_{n,2}), y_n)\} \subset \mathbb{R}^2 \times \mathcal{Y}$   
where  $x_i = (x_{i,1}, x_{i,2}) \in \mathbb{R}^2$  and  $y \in \mathcal{Y} = \{1, 2, 3\}$



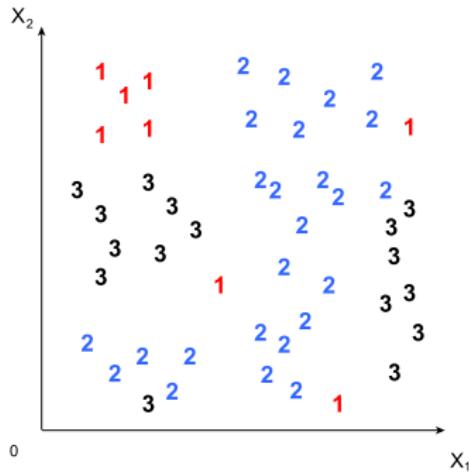
## Example of a basic DT: case of a basic classification



- Problematic: segment  $\mathbb{R}^2$  in pertinent classes using  $\mathcal{D}$
- Goal: predict for each (new)  $x$  the well associated class
- Assumptions: the learning set  $\mathcal{D}$  is representative and  $x_i$  are iid

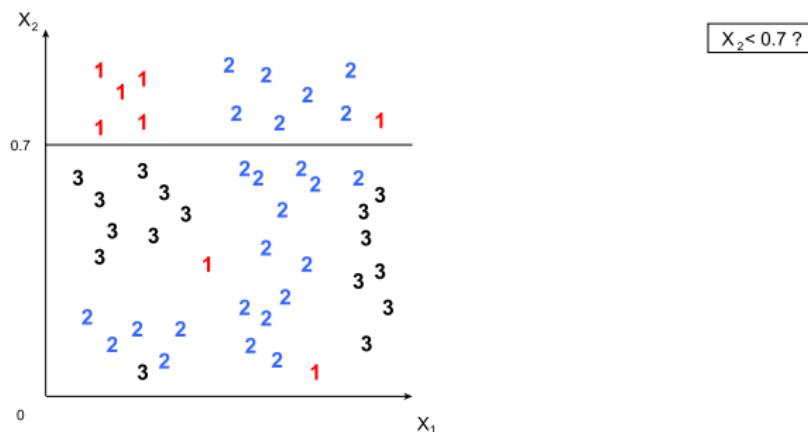
## A basic DT building and the hypothesis space

- We can't see any kind of "line" → linear model
- We need another and more appropriate hypothesis space



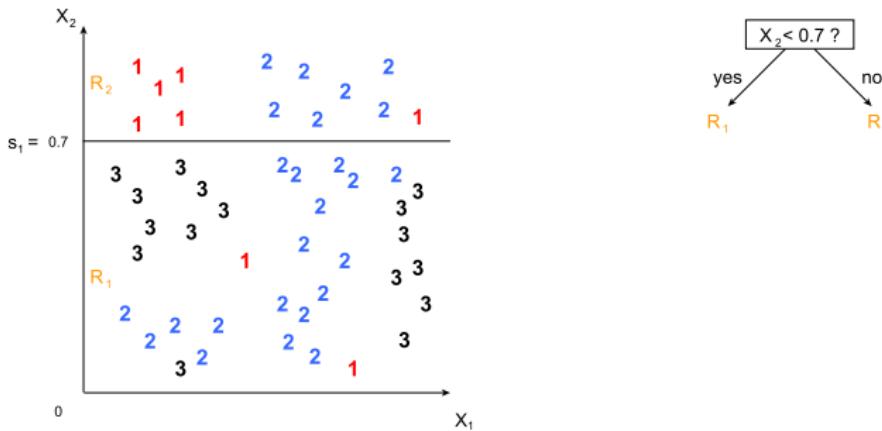
## A basic DT building (the root)

CART is an appropriate model that proceeds by using a recursive binary partition to isolate each class region



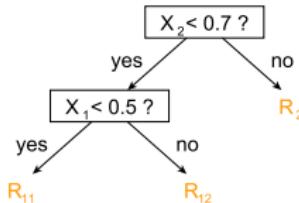
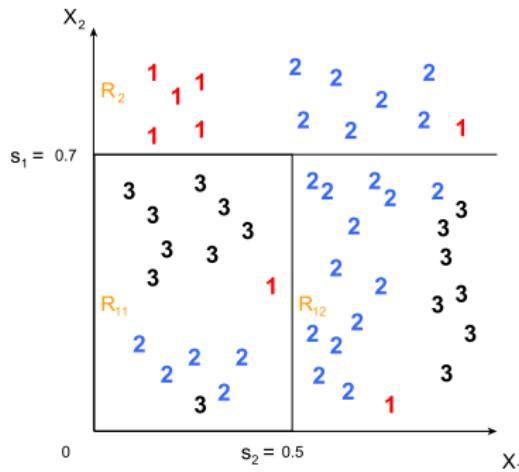
- The horizontal line represents the decision boundary of the root node (depth 0):  $X_2 = 0.7$

## A basic DT building (depth = 1)



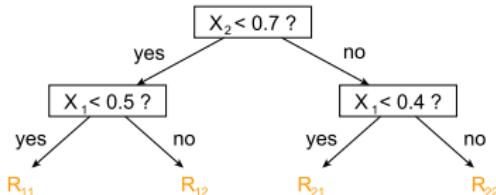
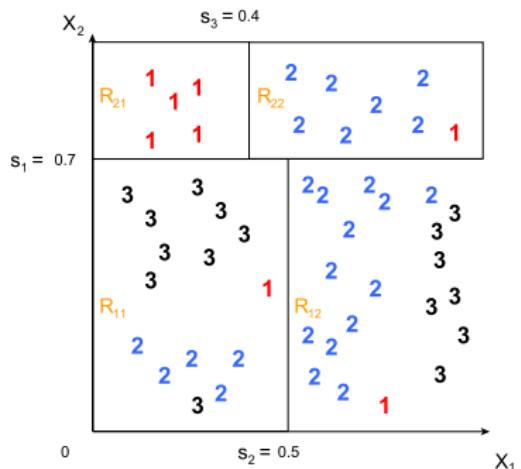
- However, the  $R_1$  and  $R_2$  areas are impure, so that we will grow a depth-2

## A basic DT building (depth = 2)



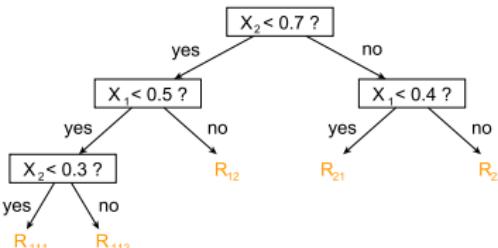
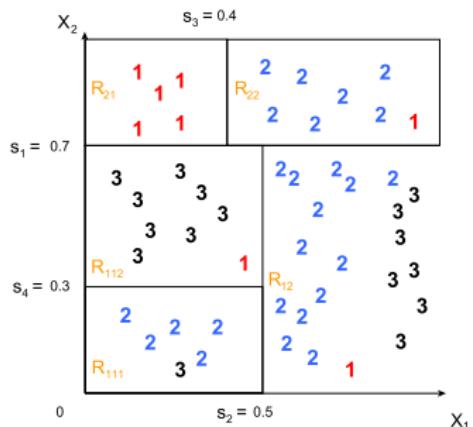
- Start splitting  $R_1$ : the depth-1 left node splits it at  $X_1 = 0.5$  represented by the vertical segment

## A basic DT building (depth = 2)



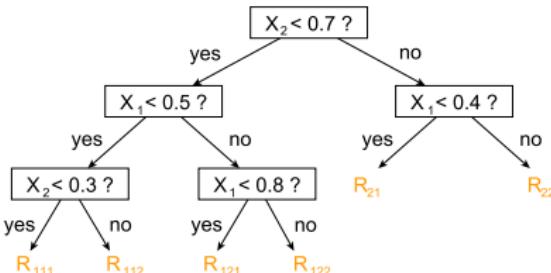
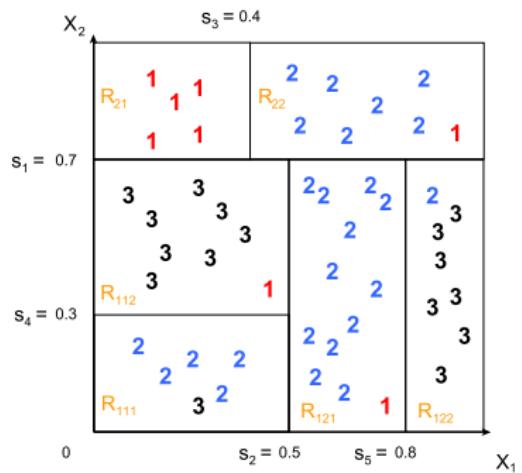
- Similarly,  $R_2$  is split: the depth-1 right node splits it at  $X_1 = 0.4$  represented by the upper vertical segment
- The two upper regions are pure enough; it won't be split any further
- However, the down regions are still impure

## A basic DT building (depth = 3)



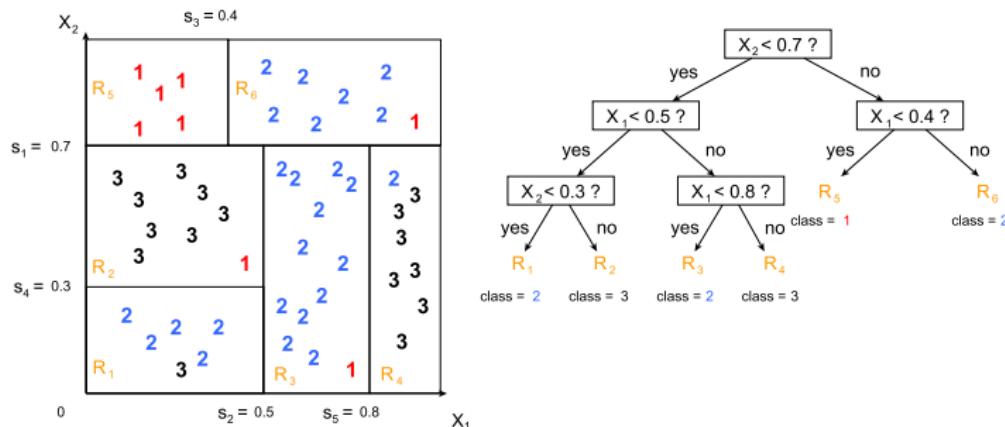
- The depth-2 left node splits it at  $X_2 = 0.3$
- i.e. the region  $R_{11}$  is split and a new horizontal segment appears giving two regions  $R_{111}$ ,  $R_{112}$  pure enough
- However, the down-right region  $R_{12}$  needs another split

## A basic DT building (depth = 3)



- The depth-2 right node splits it at  $X_1 = 0.8$
- i.e. the region  $R_{12}$  is split and a new vertical segment appears giving two regions  $R_{121}$ ,  $R_{122}$  pure enough

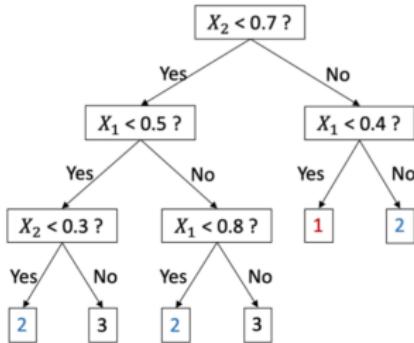
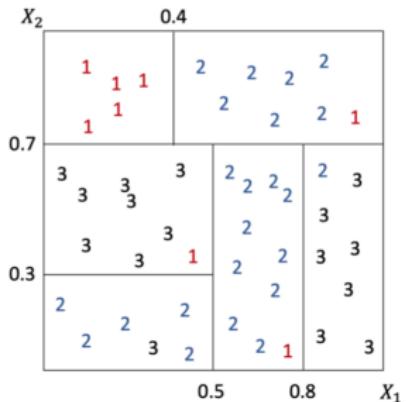
## A basic DT building (depth = 3)



**Figure:** A classification tree predicting labels on a two-dimensional input space (left: the spatial segmentation, right: the visualized tree).

- The DT stops right there
- In practical case, you can parameterize the "pure enough" concept by setting "max\_depth=3"
- We will introduce multiple stopping criterion

## The basic DT obtained

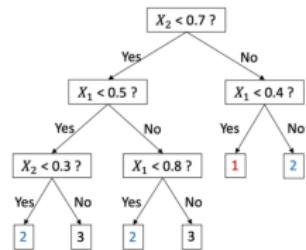
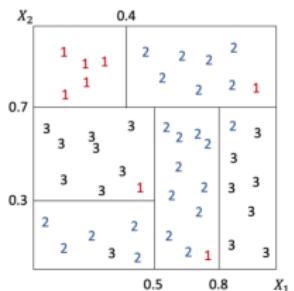


**Figure:** A classification tree predicting labels on a two-dimensional input space (left: the spatial segmentation, right: the visualized tree). Source: [Zhou, 2019].

- CART provides a partition of  $\mathbb{R}^2$  in classes by horizontal/vertical line segments

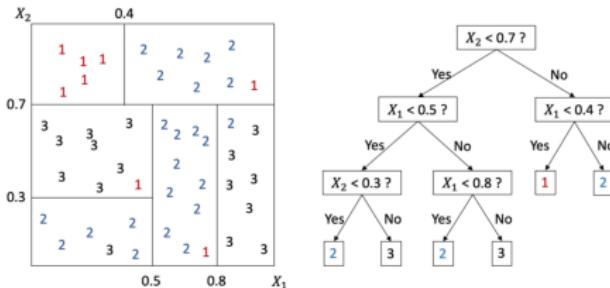
## How this DT makes predictions?

Suppose you have a new  $x_{new}$  instance and you want to classify it



- ➊ You start at the root node (depth 0, at the top):  
this node asks whether  $x_{new}$  has a  $X_2$  value smaller than 0.7
- ➋ If it is not, then you move down to the root's right child node (depth 1, right)
- ➌ This is not a leaf node  $\Rightarrow$  It asks another question:  
Is the  $X_1$  value of  $x_{new}$  smaller than 0.4?
- ➍ If it is, then  $x_{new}$  is most likely in the class 1. If not, it is likely in the class 2.

## How estimate the class probabilities of an instance?



- ① First, the instance traverses the tree to find the leaf node for it
- ② It returns the ratio of training instances of each class  $c$  in this node

E.g.,  $x_i = (0.45, 0.05)$ : the corresponding leaf node is the depth-3 left node ( $R_1$ )

So, DT should output the probabilities: 0 for the class **1**,  $5/6=0.83$  for the majority class **2** and  $1/6=0.17$  for the class **3**

Of course, DT predicts the class **2** for  $x_i$

## Model interpretation: white box versus black box

- As you can see DTs are intuitive and their decisions are easy to interpret!
- Such models are often called white box models
- In contrast, as we will see in next a lecture, Random Forests are generally considered black box models
- Black box models make great predictions, but it is usually hard to explain in simple terms why the predictions were made

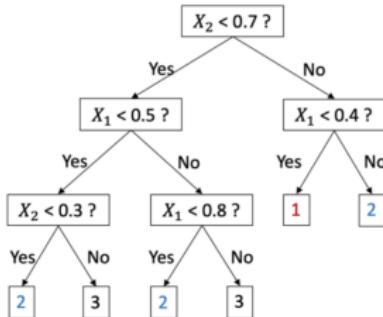
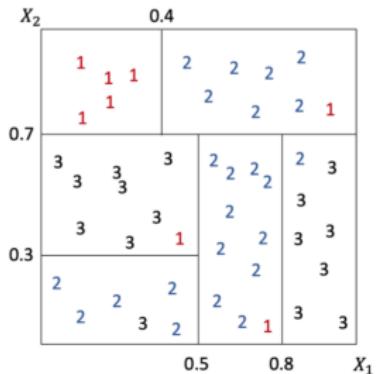
E.g., if a neural network says that a particular person appears on a picture, it is hard to know what actually contributed to this prediction: did the model recognize that person's eyes? Her nose?

- Conversely, DTs provide nice and simple prediction rules that can even be applied manually if need be

## Vocabulary used and to use

- Partitioning has a simple description like  $X_2 = s_1$
- $s_1, s_2, s_3, s_4, s_5$  are the five **splits**
- Each **node** of the tree is characterized by a split and represent a **region**
- We have 6 final regions  $R_1, R_2, R_3, R_4, R_5, R_6$
- The first node is called the **root**
- The two following nodes
  - are the **left and right childs**
  - are located at the **depth=1** of the tree
- The final regions are called **leaves** (or terminal nodes) and give the final partition of the **feature space**  
$$\mathcal{X} = R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5 \cup R_6$$

## The prediction rule



- The structure of the learned DT gives the classification rule
- Its formula is the function  $\hat{f}$  learned to be as close as possible to  $f$

### Exercise 1

Write the  $\hat{f}(x)$  formula associated to the classification tree example?

## Vocabulary and conventions

- A decision tree is characterized by the regions partitioning  $\mathcal{X}$  and the classes (or values) predicted for each region

### Exercise 2

Rewrite  $\hat{f}(x)$  formula with the regions  $R_1, R_2, R_3, R_4, R_5, R_6$ ?

### Exercise 3

Rewrite  $\hat{f}(x)$  formula with a sum symbol from  $k = 1$  to  $k = K = 6$ ?

Remark:  $k$  also indexes the nodes

## Decision tree model

### Definition (Decision tree)

A decision tree is a binary tree that recursively segment the input space  $\mathcal{X}$ , also called the feature space, along the coordinate directions to create a set of  $K$  hyper-rectangles  $R_k$ ,  $k \in \{1, \dots, K\}$  as basic prediction units for fitting constant values  $\gamma_k$  within each of them. The decision tree model is formulated as follow:

$$f(x) := \sum_{k=1}^K \gamma_k \mathbb{1}_{\{\mathbf{x} \in R_k\}} \quad (1)$$

where  $x \in \mathcal{X}$  and  $\gamma_k$  the corresponding predicted value or class for the dependent variable (called also output variable and denoted  $Y$ ).

We call  $\gamma_k$  and  $R_k$  the "parameters" of the DT model.

## Parameters of a decision tree model

Elements of intuition:

- The set of  $K$  hyper-rectangles  $R_k$  is determined by a set of  $K - 1$  chosen splits and their order during the recursive binary segmentation
- Each split is determined by a greedy strategy to best discriminate the observations in its two branches respectively
- The fitted value  $\gamma_k$  assigned to each terminal node  $R_k$  is usually decided by the majority vote (classification task) or the average (regression task)
- Several decision tree models are possible: as much as input space partitions and set of  $\gamma_k$  (instability)

## Parameters of a decision tree model

- We speak about DT as no-parametric model
- It's not because it does not have any but because the number of parameters is not determined prior to training
- So the model structure is free to stick closely to the data
- Conversely, a linear model is a parametric one with parameters allowing to don't stick closely to the data

That's why DT come from the no-parametric models family but have parameters

## How estimate the parameters of a DT model

- We wish the best decision tree predictor  $\hat{f}(x) = \sum_{k=1}^K \hat{\gamma}_k \mathbb{1}_{\{x \in \hat{R}_k\}}$
- In practice, this is realized by **minimizing** a given **risk function** or **impurity measure** (this is called the **main optimization problem**)
- Allowing to evaluate the error made by the learning predictor

### Definition (Risk function)

$$\mathcal{R}(f) := P(y \neq f(x))$$

## Examples of risk function (Classification case)

- $\mathcal{R}_n(f) := \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{\{y_i \neq f(x_i)\}}$  is the empirical misclassification loss
- The misclassification loss is illustrated below with another losses

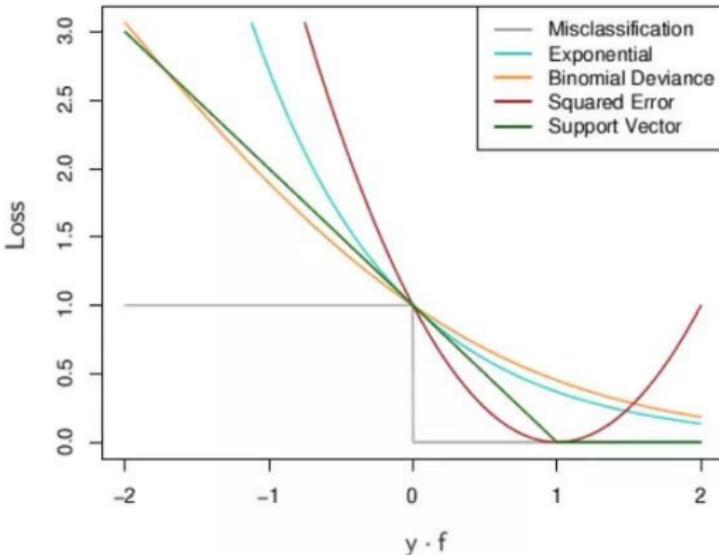


Figure: Illustration of some losses for binary classification

## Examples of risk function (Regression case)

### Definition (MSE - Mean Squared Error)

$\mathcal{R}_n(f) := \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$  is the empirical and squared loss also called L2-loss or MSE.

Each empirical risk has a no-empirical formula:

$\mathcal{R}(f) := \int_{\mathcal{X} \times \mathcal{Y}} (y - f(x))^2 dP(x, y)$  is the squared loss

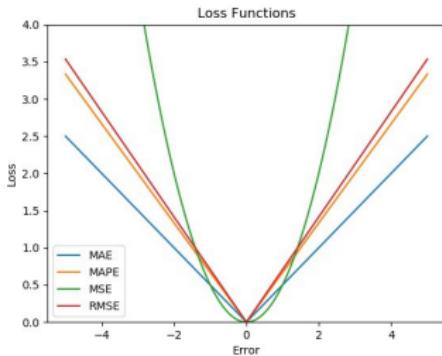


Figure: Illustration of some popular losses for regression

## Building a regression DT

- The minimization of the risk function allows the construction of a decision tree by determining the best values for the sets of  $R_k$  and  $\gamma_k$
- Given a learning set  $\mathcal{D} = \{(x_i, y_i)_{1 \leq i \leq n}\} \subset \mathcal{X} \times \mathcal{Y}$  and using the empirical quadratic risk function, one should solve:

### Optimization problem (empirical case)

$$\min_{\gamma_1, \dots, \gamma_K, R_1, \dots, R_K} \mathcal{R}_n(f) = \min_{\gamma_1, \dots, \gamma_K, R_1, \dots, R_K} \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2$$

- Another version of optimization problem:

$$\min_{\gamma_1, \dots, \gamma_K, R_1, \dots, R_K} \mathbb{E} \left[ (y_i - f(x_i))^2 \right]$$

## Building a regression DT: predictive rule

Let's solve the optimization problem ( $\gamma_k$  case)

$$\min_{\gamma_1, \dots, \gamma_K} \frac{1}{n} \sum_{i=1}^n \left( y_i - \sum_{k=1}^K \gamma_k \mathbb{1}_{\{x_i \in R_k\}} \right)^2$$

$$\forall k \in \{1, \dots, K\}, \quad \hat{\gamma}_k = \operatorname{argmin}_{\gamma_k} \frac{1}{n} \sum_{i=1}^n (y_i - \gamma_k \mathbb{1}_{\{x_i \in R_k\}})^2 \text{ (disjoint regions)}$$

$$\forall k \in \{1, \dots, K\}, \text{ given } x_i \in R_k, \quad \hat{\gamma}_k = \operatorname{argmin}_{\gamma_k} \frac{1}{n} \sum_{i=1}^n (y_i - \gamma_k)^2$$

Then,  $\forall k \in \{1, \dots, K\}$ , given  $x_i \in R_k$ ,

$$\begin{aligned} \frac{\partial}{\partial \gamma_k} \mathcal{R}_n(f) &= 0 \Leftrightarrow \sum_{i=1}^n -2(y_i - \gamma_k) = 0 \text{ (formula derivates)} \\ \Leftrightarrow \hat{\gamma}_k &= \frac{1}{n} \sum_{i=1}^n y_i \quad \text{given } x_i \in R_k \end{aligned}$$

## Building a regression DT: predictive rule

Thus, we obtain

### the predictive rule

For each  $x_i \in R_k \implies \hat{f}(x_i) = \frac{1}{n} \sum_{i=1}^n y_i \mathbb{1}_{\{x_i \in R_k\}}$

- This is the empirical expectation of outputs corresponding to inputs belonging to region  $R_k$

Notations,

$$\hat{\gamma}_k = \frac{1}{n} \sum_{i=1}^n y_i \mathbb{1}_{\{x_i \in R_k\}}$$

$$\Leftrightarrow \hat{\gamma}_k = \frac{\sum_{i=1}^n y_i \mathbb{1}_{\{x_i \in R_k\}}}{\sum_{i=1}^n \mathbb{1}_{\{x_i \in R_k\}}}$$

$$\Leftrightarrow \hat{\gamma}_k = \text{average}(y_i | x_i \in R_k)$$

## Building a regression DT: split rule

- Finding the best binary partition in the term of risk  $\mathcal{R}_n$  is more complicated  $\Rightarrow$  proceed with a greedy algorithm
- Let  $j$  be a splitting variable (index of feature  $x^j$  from the input),  $s$  a split point and define the pair of half-planes

$$R_1(j, s) = \{x_i \in \mathcal{D} : x_{i,j} \leq s\} \text{ and } R_2(j, s) = \{x_i \in \mathcal{D} : x_{i,j} > s\}$$

- Then, we search for the couple  $(j, s)$  that produces the purest subsets, i.e., solving,

### Definition (CART cost function for regression)

$$\min_{j,s} \left[ \min_{\gamma_1} \sum_{x_i \in R_1(j,s)} (y_i - \gamma_1)^2 + \min_{\gamma_2} \sum_{x_i \in R_2(j,s)} (y_i - \gamma_2)^2 \right] \quad (2)$$

where, for any choice of  $(j, s)$ , the inner minimization is solved by

$$\hat{\gamma}_1 = \text{average}(y_i | x_i \in R_1(j, s)) \text{ and } \hat{\gamma}_2 = \text{average}(y_i | x_i \in R_2(j, s))$$

## Building a regression DT: split rule

- To compute  $\hat{\gamma}_1$  and  $\hat{\gamma}_2$  we need to determine  $R_1(j, s)$  and  $R_2(j, s)$ ,  
ie: the couple  $(j, s)$

### The splitting process of one region in two child regions

- We scan each dimension of all the inputs  $j \in \{1, \dots, d\}$  and each  $n - 1$  split points for each fixed  $j$
- For each split  $(j, s)$  we compute the risk value thanks to  $\hat{\gamma}_1$  and  $\hat{\gamma}_2$  formula
- We stock smartly the risk values computed for each couple  $(j, s)$
- Choose the couple  $(j, s)$  that associated to the minimal risk value
- Partition the data into the two resulting regions
- This splitting process is repeated for each child regions, and so on
- That's why we call it a "greedy" process

## Building a regression DT: stopping criterion and pruning

How much should we let the tree grow?

- Too big a tree may **overfit** the data
- Too small a tree risk of **not capturing their structure**
- The optimal size is a **hyper-parameter to calibrate** on the data

Solutions:

- Grow as long as the variance within the leaves decreases more than a fixed threshold
- Too short-term vision:** a bad cut can sometimes lead to very good cuts
- **Pruning:** we make the tree grow to the maximum and then we remove the uninteresting nodes.

## Building a regression DT: stopping criterion

- The repetition is made until a stopping criterion is met
- To avoid overfitting the training data, you need to restrict the DT's freedom during training
- Examples of stopping criterion or "regularization by setting hyperparameters",
  - ▶ maximum depth
  - ▶ maximum number of leaves
  - ▶ minimum number of sample per leaf
- They can help to avoid overfitting by avoiding, to grow the maximal tree that overfits the training data

Suggestion: During the lab or at home, try to observe it with moon dataset:

```
from sklearn.datasets import make_moons
```

## Building a DT: pruning

Another way to avoid overfitting

We re-examine the tree and remove the nodes caused by the noise present in the learning set

### Penalized criterion

- The optimal tree is a pruned subtree **minimizing** the prediction error penalized by the complexity of the model (length of the tree)  $|T|$ ,

$$\mathcal{R}_\alpha(f_T) = \mathcal{R}_n(f_T) + \alpha \frac{|T|}{n}$$

$\mathcal{R}_n(f_T)$  is the error term linked to the decision subtree  $f_T$  (ex: Mean Squared error or the rate of misclassifications)

- The complexity parameter  $\alpha \in [0, 1]$  penalizes large trees

## Evaluate a regression DT model

- The decision tree is learned on the **learning set**
- Hyper-parameters could be calibrate on a **validation set**
- To evaluate the accuracy, we could compute the error made on a **test set** via an error measure (MSE is the most popular for regression)

**MSE** (Mean Squared Error) is a risk function, corresponding to the expected value of the squared error loss,

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2$$

- Usually, a 5-fold cross validation process is realized on the data set

## Summary: the greedy building of a regression DT

### CART algorithm [Breiman et al., 1984]

- 1 Start at the root node corresponding to the full input space  $\mathcal{X}$
- 2 Given a node, enumerate all possible candidate splits by going over all features and collecting all possible split values
- 3 Choose a split solving the optimization problem (2) to separate the node, thus the corresponding feature sub-space, into two child branches
- 4 Work recursively in the child node to further split until a stopping criterion is met
- 5 Calculate the fitted value in each of the terminal nodes

This algorithm is generic to classification DT by adapting the optimization problem

# Plan

## 1 Context & motivations

- Machine learning field and weak learners
- Decision Tree: a supervised weak learner

## 2 Basic idea and reminders

- Example of basic decision tree
- Vocabulary and conventions

## 3 Decision tree model and building

- Decision tree model
- **Building regression trees**
- Building classification trees

## 4 Discussion

## Building a classification DT: Formalization

- Output variable:  $y \in \mathcal{Y} = \{1, \dots, c, \dots, C\}$ ,  $C$  classes
- Learning set  $\mathcal{D} = \{(x_i, y_i)_{1 \leq i \leq n}\} \subset \mathcal{X} \times \mathcal{Y}$

### Proportion of class $c$ observations in a node $k$

For a node  $k$  representing a region  $R_k$  with  $n_k = \sum_{i=1}^n \mathbb{1}_{\{x_i \in R_k\}}$  data points, we define:

$$\hat{p}_{k,c} := \frac{1}{n_k} \sum_{x_i \in R_k} \mathbb{1}_{\{y_i=c\}}$$

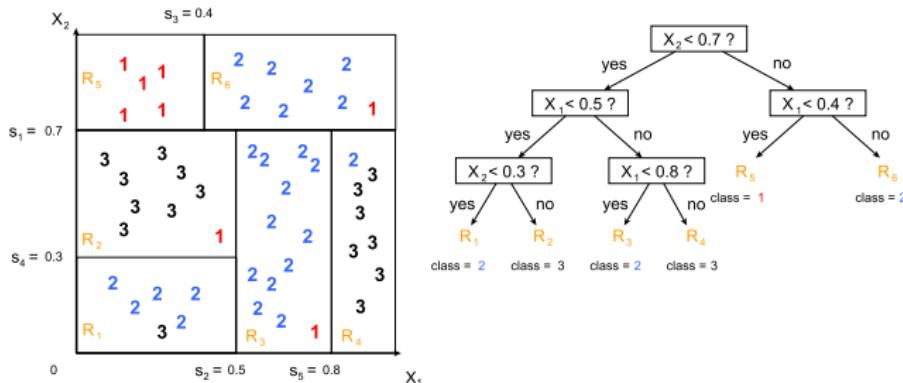
the proportion of class  $c$  observations in node  $k$

### Classification rule (majority vote)

We classify the observations in node  $k$  to class  $c(k)$  such as

$$c(k) := \operatorname{argmax}_{1 \leq c \leq C} \hat{p}_{k,c} \quad (\text{the majority class})$$

## Illustration of the classification rule



- The proportion of class  $c = 1$  observations in node  $k = 1$  is  $\hat{p}_{1,1} = \frac{1}{6} \sum_{x_i \in R_1} \mathbb{1}_{\{y_i=1\}} = 0$
- For  $c = 2$ , we have  $\hat{p}_{1,2} = \frac{1}{6} \sum_{x_i \in R_1} \mathbb{1}_{\{y_i=2\}} = 5/6$
- For  $c = 3$ , we have  $\hat{p}_{1,3} = \frac{1}{6} \sum_{x_i \in R_k} \mathbb{1}_{\{y_i=3\}} = 1/6$

The classification rule is given by the majority class,

$$c(1) = \operatorname{argmax}_{1 \leq c \leq C} \hat{p}_{1,c} = 2$$



## Building a classification DT

- Let's write  $\mathcal{D}_k$  the data points in a node  $k$ ,

$$\mathcal{D}_k = \{(x_i, y_i) \in \mathcal{D} : x_i \in R_k\}$$

- We consider risk functions adapted to classification
- To split nodes, we want to choose a **split yielding a maximal impurity reduction** to solve the optimization problem
- Several measures  $\mathcal{Q}(\mathcal{D}_k)$  of node impurity can be considered
  - Misclassification error**

$$\mathcal{Q}(\mathcal{D}_k) = \frac{1}{n_k} \sum_{x_i \in R_k} \mathbb{1}_{\{y_i \neq c(k)\}} = 1 - \hat{p}_{k,c(k)} \quad (3)$$

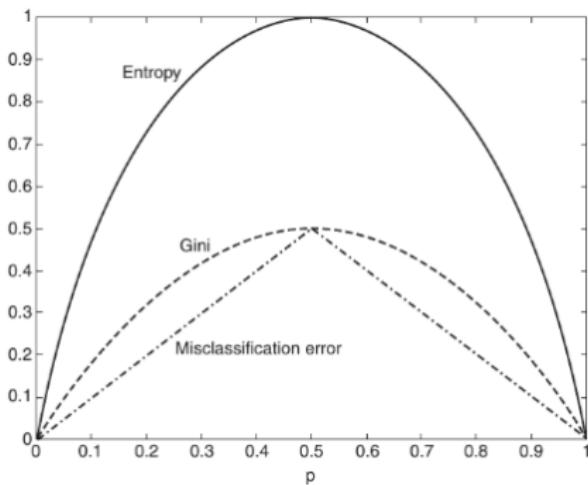
- Gini index**

$$\mathcal{Q}(\mathcal{D}_k) = \sum_{c \neq c'} \hat{p}_{k,c} \hat{p}_{k,c'} = \sum_{c=1}^C \hat{p}_{k,c} (1 - \hat{p}_{k,c}) \quad (4)$$

- Cross-entropy**

$$\mathcal{Q}(\mathcal{D}_k) = - \sum_{\substack{c=1; \hat{p}_{k,c} \neq 0}}^C \hat{p}_{k,c} \ln(\hat{p}_{k,c}) \quad (5)$$

## Building a classification decision tree



**Figure:** Illustration of node impurity measures (binary classification case), as a function of the proportion  $p$  in one class

- We don't want to be in the situation where two observations are affected to two different classes with the same probability 0.5 (as random)

## Building a classification DT: Gini or Cross-entropy?

- Most of the time it does not make a big difference and lead to similar trees
- Gini impurity is slightly faster to compute  $\Rightarrow$  it is a good default
- However, when they differ, Gini impurity tends to isolate the most frequent class in its own branch of the tree, while entropy tends to produce slightly more balanced trees

## Building a classification tree: split rule

The tree is grown as per the regression tree presented before

- A region  $R_k$  is given
- Consider a couple  $(j, s)$  of splitting variable and split point
- Let's write  $\mathcal{D}_{k,L}(j, s)$  the resulting data set of left node of size  $n_{k,L}$  and  $\mathcal{D}_{k,R}(j, s)$  for the right node, of size  $n_{k,R}$
- The tree is growing by solving the following optimization problem,

### Definition (CART cost function for classification)

$$\min_{j,s} \{ n_{k,L} \mathcal{Q}(\mathcal{D}_{k,L}(j, s)) + n_{k,R} \mathcal{Q}(\mathcal{D}_{k,R}(j, s)) \}$$

where  $\mathcal{Q}()$  is one of the impurity measures (3),(4),(5)

## Building a classification tree: stopping criterion, pruning, evaluation

All notions about

- stopping criterion
- pruning
- evaluation

presented previously are extendable to classification decision trees

The error measure used for the classification case is the misclassification error Eq. (3)

## Several point of views of a decision tree

- Decision tree (DT) is a piece-wise constant estimate and a finite linear combination of hyper-rectangular indicators functions
- Among Lebesgue theory deeper is a DT, better is the accuracy
- DT is an adaptive nearest neighbor smooth where the distance measure between 2 points corresponds to the likelihood of them being in the same terminal node
- A DT is able to create a topology on the feature space describing the similarity between observations
- A DT is a combination of binary decision paths mimicking human decision, where a decision rule is a dichotomous evaluation (of if-then predicates)

## Decision tree pros

- DT is a powerful machine learning tool without relying on structural assumptions regarding the underlying space
- DTs are interpretable predictors and easy to train
- DT can handle categorical values (binerazing them)
- DT can handle missing input values ([[Hastie et al., 2009](#)], chapter 9)

## Decision tree cons

- The construction process is greedy (cf. DT's complexity in [Géron, 2017], chapter 6)
- CARTs are neither smooth, Liepshitz nor even continuous
- DTs are adaptive to training sample through their building algorithm
  - Analysis have to be conditioned by the learning data
- Thus, trees are **unstable**: DTs have a high variance
  - A small change on data can result on a different predictor
- A way to reduce the related variance: **ensemble methods**
  - ▶ Bagging
  - ▶ Random forests
  - ▶ Boosting

Random Forests can limit this instability by averaging predictions over many DTs, as we will see in the next lecture!



# References I

-  Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984).  
*Classification and regression trees.*  
CRC press.
-  Géron, A. (2017).  
Hands-on machine learning with scikit-learn and tensorflow: Concepts.  
*Tools, and Techniques to build intelligent systems.*
-  Hastie, T., Tibshirani, R., and Friedman, J. (2009).  
*The elements of statistical learning: data mining, inference, and prediction.*  
Springer Science & Business Media.
-  Zhou, Y. (2019).  
*Asymptotics and Interpretability of Decision Trees and Decision Tree Ensembles.*  
PhD thesis, Cornell University.