Machine Learning in High Dimension
IA317
Ensemble Methods

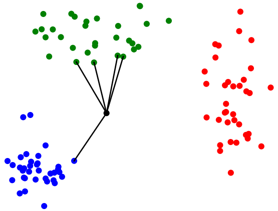
Thomas Bonald

2023 – 2024

Back to nearest neighbors

A set of **methods** for

- Classification
- Regression
- Clustering
- Anomaly detection



Advantages

- Simple
- Explainable

Issues

- Choice of distance
- Complexity

Nearest neighbor search

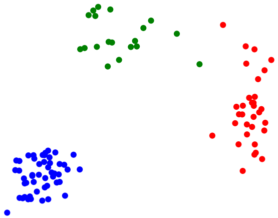
Exact search:

- Exhaustive search $O(n)$
- Tree search $O(\log n) \to O(n)$

Approximate search:

- Locally sensitive hashing $O(1)$

What if data have a **simple** structure?



Outline

- **Decision trees**
- Bagging methods → Random forests, ExtraTrees
- Boosting methods → Gradient boosting, XGBoost

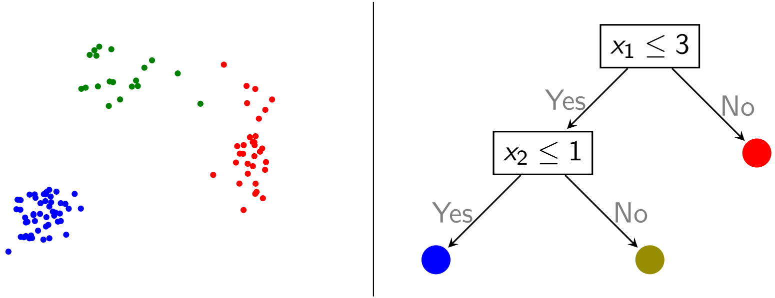
Decision tree

Consider data samples in $\mathbb{R}^d$

Definition

A decision tree is a binary tree providing a **partition** of $\mathbb{R}^d$.
Each node of the tree corresponds to a **split** along a **single** feature.

Note: Same as KD-trees, but supervised!

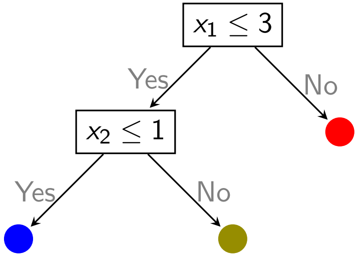


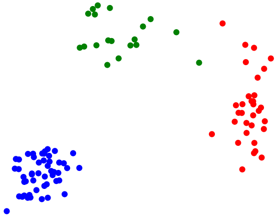
Classification tree

Objective

Build the **most compact** decision tree whose leaves are **pure**.

Problem: A NP-hard problem!
Solution: Greedy algorithm





Gini impurity

Let $p_1, \ldots, p_K$ be a probability distribution over $\{1, \ldots, K\}$.

Definition

$$G = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

Interpretation: Probability of **error** by a random prediction drawn from $p_1, \ldots, p_K$

- $G = 0$ for a **pure** distribution (no randomness)
- $G = 1 - \frac{1}{K}$ for a **uniform** distribution

Application: Impurity of a **multi-set**, e.g., $\{1, 1, 2, 3\}$

Entropy

Let $p_1, \ldots, p_K$ be a probability distribution over $\{1, \ldots, K\}$.

Definition

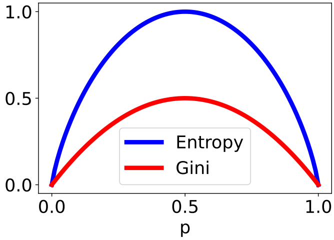
$$H = -\sum_{k=1}^{K} p_k \log p_k$$

Interpretation: Average number of bits per symbol to encode a sequence of i.i.d. symbols $Y_1, Y_2, \ldots$

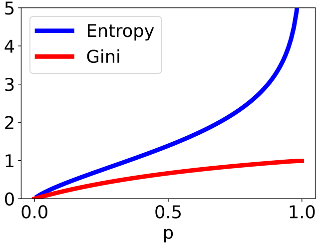
- $H = 0$ for a **pure** distribution (no randomness)
- $H = \log K$ for a **uniform** distribution

Application: Entropy of a **multi-set**, e.g., $\{1, 1, 2, 3\}$

Gini impurity vs. Entropy



Bernoulli distribution

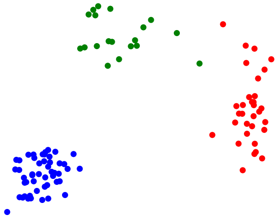


Geometric distribution

Tree construction

Training data = n samples with labels, d features

$$X \in \mathbb{R}^{n \times d}, \quad y \in \{1, \ldots, K\}^n$$



Tree construction

Training data = n samples with labels, d features

$$X \in \mathbb{R}^{n\times d}, \quad y \in \{1,\dots,K\}^n$$

Algorithm

$S \leftarrow \{1,\dots,n\}$
$G \leftarrow$ **randomness** of $\{y_i, i \in S\}$ (Gini or entropy)

While $G > 0$, find the **split** (feature j, threshold t) maximizing

$$\boxed{\Delta G = G - (\alpha_L G_L + \alpha_R G_R)}$$

with

- $S_L \leftarrow \{i \in S : X_{ij} \leq t\}$, $S_R \leftarrow \{i \in S : X_{ij} > t\}$
- $\alpha_L \leftarrow \frac{|S_L|}{|S|}$, $\alpha_R \leftarrow \frac{|S_R|}{|S|}$
- $G_L \leftarrow$ rand. of $\{y_i, i \in S_L\}$, $G_R \leftarrow$ rand. of $\{y_i, i \in S_R\}$

Interpretation

Each split maximizes the **information gain**:

$$\boxed{\Delta G = G - (\alpha_L G_L + \alpha_R G_R)}$$

$$\begin{tikzcd} & \boxed{G} \arrow[dl] \arrow[dr] & \\ \boxed{G_L} & & \boxed{G_R} \end{tikzcd}$$

How to interpret $\alpha_L G_L + \alpha_R G_R$?

- **Gini impurity**
  Probability of error by a random prediction **given** the split (S_L or S_R)
- **Entropy**
  Conditional entropy of the labels given the decision (S_L or S_R)

Exercise

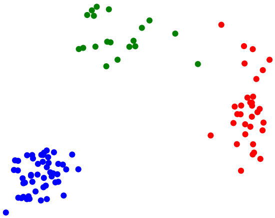
$$x \in \mathbb{R},\ y \in \{1, 2, 3\}$$

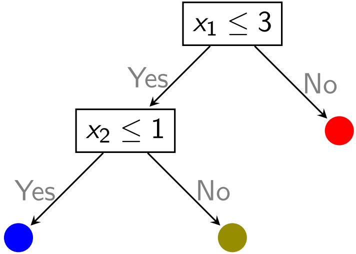
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
y	3	1	1	1	2	2	3	3	2

What is the **gain** of the split $x \leq 0.4$?

Computational cost

The samples must be **sorted** for each feature $\rightarrow O(dn \log n)$

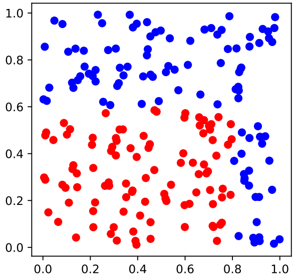


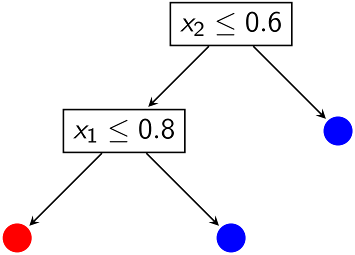


Example (rectangle)

$$x \in \mathbb{R}^2,\ y \in \{0, 1\}$$

Training data

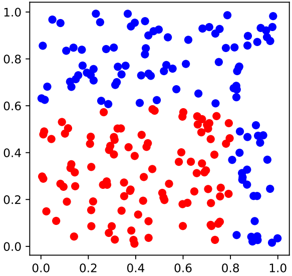




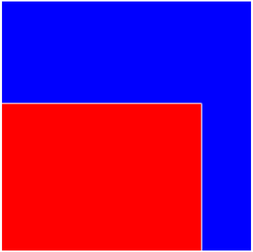
Example (rectangle)

$$x \in \mathbb{R}^2,\ y \in \{0, 1\}$$

Training data



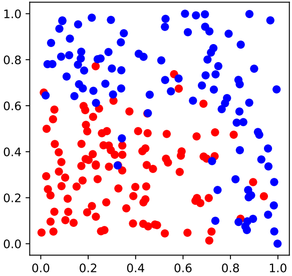
Model (depth 2)



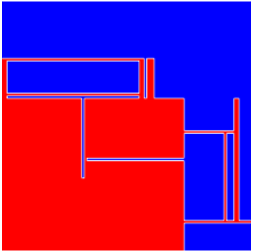
Example (noisy rectangle)

$$x \in \mathbb{R}^2,\ y \in \{0, 1\}$$

Training data



Model (depth 10)



Pruning

Idea

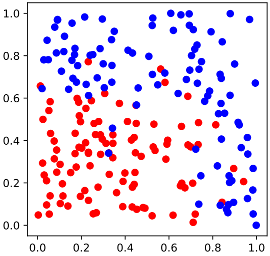
Limit the **depth** to avoid **over-fitting**

Problem: Leaves can now contain **several** labels...
Solution : Return the **most frequent** label of the leaf!

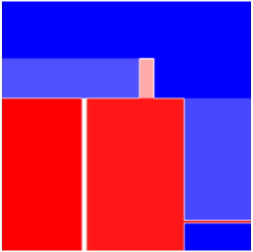
Example (noisy rectangle)

$$x \in \mathbb{R}^2,\ y \in \{0, 1\}$$

Training data



Model (max depth 4)



Feature importance

Idea: A feature is important if it reduces **randomness**.

Definition

The **importance** of feature j is:

$$\boxed{F_j = \sum_{s \text{ using } j} \alpha_s \Delta G_s}$$

where

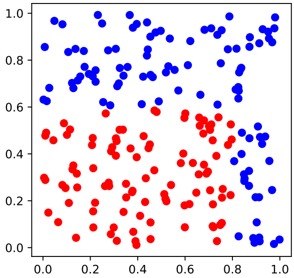
- s is a split (internal node of the tree)
- α_s is the **proportion of samples** concerned by the split s
- ΔG_s is the **information gain** brought by the split s

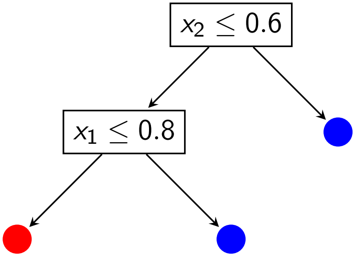
Note: In the absence of pruning, $G = \sum_{j=1}^{d} F_j$

Exercise

$$x \in \mathbb{R}^2,\ y \in \{0, 1\}$$

Training data





What is the most important feature?

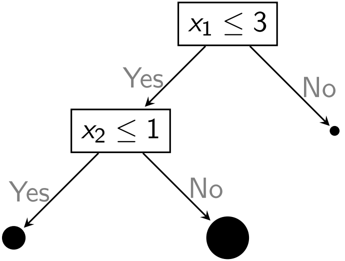
Regression tree

Training data = n samples with **values**, d features

$$X \in \mathbb{R}^{n \times d}, \quad y \in \mathbb{R}^n$$

Objective

Build a decision tree whose leaves have **similar** values.



Tree construction

Training data = n samples with values, d features

$$X \in \mathbb{R}^{n \times d}, \quad y \in \mathbb{R}^n$$

Tree construction

Training data = n samples with values, d features

$$X \in \mathbb{R}^{n \times d}, \quad y \in \mathbb{R}^n$$

Algorithm

$S \leftarrow \{1, \ldots, n\}$
$V \leftarrow$ **variance** of $\{y_i, i \in S\}$

While $V > 0$, find the **split** (feature j, threshold t) maximizing

$$\boxed{\Delta V = V - (\alpha_L V_L + \alpha_R V_R)}$$

with

- $S_L \leftarrow \{i \in S : X_{ij} \leq t\}$, $S_R \leftarrow \{i \in S : X_{ij} > t\}$
- $\alpha_L \leftarrow \frac{|S_L|}{|S|}$, $\alpha_R \leftarrow \frac{|S_R|}{|S|}$
- $V_L \leftarrow$ rand. of $\{y_i, i \in S_L\}$, $V_R \leftarrow$ rand. of $\{y_i, i \in S_R\}$

Feature importance

Idea: A feature is important if it reduces **variance**.

Definition

The **importance** of feature j is:

$$\boxed{F_j = \sum_{s \text{ using } j} \alpha_s \Delta V_s}$$

where

- s is a split (internal node of the tree)
- α_s is the proportion of samples concerned by this split
- ΔV_s the gain in variance

Note: In the absence of pruning, $V = \sum_{j=1}^{d} F_j$

Outline

- Decision trees
- **Bagging methods** $\rightarrow$ Random forests, ExtraTrees
- Boosting methods $\rightarrow$ Gradient boosting, XGBoost

Bagging[1]

Idea: Generate a strong learner by **aggregating** the decisions of several weak learners (typically, decision trees)

- **Classification**
  through **majority vote**
- **Regression**
  through **averaging**

Require **diversity** in learners $\rightarrow$ sampling

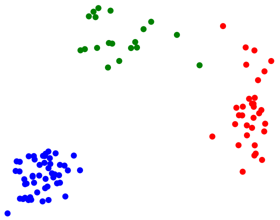
[1]Bootstrap Aggregating

Random Forests

Training data = n samples, d features

Principle

Use m decision trees, each for n data samples selected uniformly at random with replacement $\rightarrow$ **bootstrap**



Random Forests

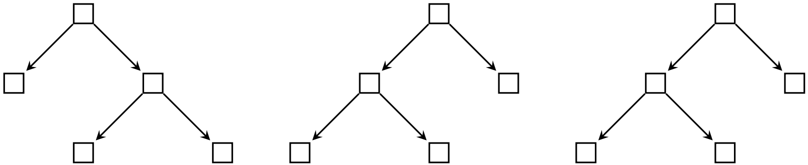
Training data $=$ n samples, d features

Principle

Use m decision trees, each for n data samples selected uniformly at random with replacement $\rightarrow$ **bootstrap**

Notes:

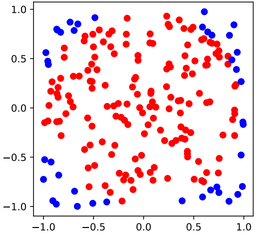
- Each bootstrap contains a proportion $1-(1-\frac{1}{n})^n \approx 1-e^{-1} \approx \frac{2}{3}$ of samples
- **Deep** trees (no pruning)
- For **each** tree and **each** split, use k features selected uniformly at random (typically, $k=\sqrt{d}$)



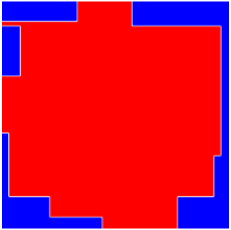
Example (disk)

$$x \in \mathbb{R}^2,\ y \in \{0, 1\}$$

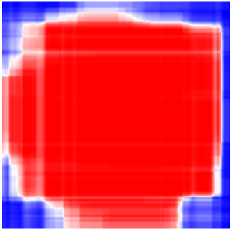
Training data



Decision tree



Random forest



Real data: Digits

1797 pictures of digits
8×8 pixels
16 grey levels

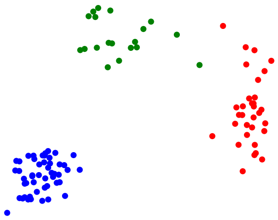
Algorithm	Accuracy
Decision tree	86%
Forest (1 tree)	76%
Forest (10 trees)	94%
Forest (100 trees)	97%

ExtraTrees[1]

Idea: Add randomness in the splits.

Principle

Same as Random Forests, but with a **random threshold** for each feature. The best split (for these random thresholds) is selected.



[1] Extremely Randomized Trees

Real data: Digits

1797 pictures of digits
8×8 pixels
16 grey levels

Algorithm	Accuracy
Decision tree	86%
Forest (1 tree)	76%
Forest (10 trees)	94%
Forest (100 trees)	97%
ExtraTrees (1 tree)	81%
ExtraTrees (10 trees)	97%
ExtraTrees (100 trees)	99%

Outline

- Decision trees
- Bagging methods $\rightarrow$ Random forests, ExtraTrees
- **Boosting methods** $\rightarrow$ Gradient boosting, XGBoost

Boosting

Idea: Improve the estimator **sequentially** by **correcting** the errors

Let $y_1, \ldots, y_n \in \mathbb{R}$ the values to predict
Let $f : x \mapsto y$ be the estimator

Principle

Init $f \leftarrow \frac{1}{n} \sum_{i=1}^{n} y_i$
Repeat

- $h \leftarrow \arg\min_h \mathcal{L}(f + \alpha h)$
- $f \leftarrow f + \alpha h$

Notes:

- A **regression** problem, also applicable to **binary** classification
- $\mathcal{L}$ is the **loss function**
- $\alpha \in (0, 1]$ is the **learning rate**

Gradient Boosting

Let $y_1, \ldots, y_n \in \mathbb{R}$ the values to predict
Let $f : x \mapsto y$ be the estimator

Algorithm

Init $f \leftarrow \frac{1}{n} \sum_{i=1}^{n} y_i$
Repeat

- $h \leftarrow$ **regression tree** for values $y_1 - f(x_1), \ldots, y_n - f(x_n)$
- $f \leftarrow f + \alpha h$

Notes:

- The regression tree has **small depth**
- Equivalent to **gradient descent** for the loss function:

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

XGBoost[1]

Key idea: Add **regularization**

Let $y_1, \dots, y_n \in \mathbb{R}$ the values to predict
Let $f : x \mapsto y$ be the estimator

Principle

Init $f \leftarrow \frac{1}{n} \sum_{i=1}^{n} y_i$
Repeat

- $h \leftarrow$ **regression tree** minimizing $\mathcal{L}(f + \alpha h) + \Omega(h)$
- $f \leftarrow f + \alpha h$

Regularization for regression tree h, with partition $A_1, \dots, A_L$:

$$\boxed{\Omega(h) = \sum_{l=1}^{L} (\gamma + \frac{\lambda}{2} v_l^2)} \quad \text{for} \quad h = \sum_{l=1}^{L} v_l 1_{A_l}$$

[1] eXtreme Gradient Boosting

Regression tree in XGBoost

Consider the loss function:

$$\mathcal{L}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{y}_i, y_i) \qquad \hat{y}_1 = f(x_1), \dots, \hat{y}_n = f(x_n)$$

For the partition $A_1, \dots, A_L$, the criterion to minimize is:

$$\boxed{J = \sum_{l=1}^{L} \left(\gamma - \frac{1}{2}\frac{G_l^2}{\lambda + H_l}\right)}$$

with

$$G_l = \sum_{i \in A_l} \frac{\partial \ell}{\partial \hat{y}}(\hat{y}_i, y_i) \quad H_l = \sum_{i \in A_l} \frac{\partial^2 \ell}{\partial \hat{y}^2}(\hat{y}_i, y_i)$$

The regression tree is built recursively, with a split whenever the criterion J **decreases**.

Loss functions

The regression tree depends on the loss function:

- **Regression**

$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

$$\frac{\partial \ell}{\partial \hat{y}}(\hat{y}, y) = \hat{y} - y, \quad \frac{\partial^2 \ell}{\partial \hat{y}^2} = 1$$

- **Binary classification**

$$\ell(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \quad \text{with} \quad \hat{y} = \frac{e^z}{1 + e^z}$$

$$\frac{\partial \ell}{\partial z}(\hat{y}, y) = -|\hat{y} - y|, \quad \frac{\partial^2 \ell}{\partial z^2} = \hat{y}(1 - \hat{y})$$

Real data: Digits

1797 pictures of digits
8×8 pixels
16 grey levels

Algorithm	Accuracy
Decision tree	86%
Forest (1 tree)	76%
Forest (10 trees)	94%
Forest (100 trees)	97%
Gradient boosting	95%
XGBoost	96%

Summary

Ensemble methods

- **Bagging** → Random Forests
  Parallel training
  Aggregation
- **Boosting** → Gradient Boosting, XGBoost
  Sequential training
  Correction