Machine Learning

Ensemble learning, Random Forest, Boosting

Simon BERNARD
simon.bernard@univ-rouen.fr

Outline

Chapters :

Introduction to ensemble learning

Supervised learning

- **Inputs** : $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ (often called instances)
- **Outputs**[1] : $y \in \mathcal{Y} = \{\lambda_1, \lambda_2, \ldots, \lambda_c\}, c \geq 2$
- **Core hypothesis** : $\exists$ an unknown function $f : \mathcal{X} \to \mathcal{Y}$ that maps any input to its output
- **Goal :** find an approximation h of f, s.t. $\hat{y} = h(\mathbf{x})$ is a good output prediction $\forall\, \mathbf{x} \in \mathcal{X}$.
- **Core idea :** determine h from a set of n labeled instances :

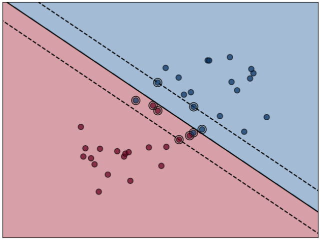
$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$$

where the $\mathbf{x}_i \in \mathcal{D}$ are associated to their "true" output values y_i

1. For simplicity, we focus on classification but most of the concepts in this course are readily transposable to regression tasks.

Supervised learning

- The goal of machine learning is to find the best h possible
- To do so, we rely on
  1. a hypothesis space $\mathcal{H}$
  2. a learning algorithm $\mathcal{A}$ that can output any $h \in \mathcal{H}$



- $\mathcal{H}$: linear models

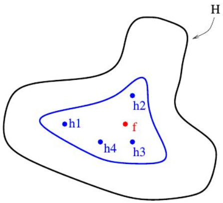
$$h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \quad \forall\, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

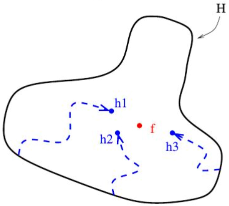
- $\mathcal{A}$: linear SVM for finding the best $h \in \mathcal{H}$

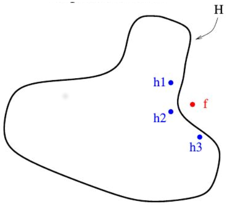
Introduction to ensemble learning

Limitations of learning a single *h*

1. Statistical : if $\mathcal{D}$ is too small compared to $\mathcal{H}$, many h_t that performs equally on $\mathcal{D}$
2. Computational : if $\mathcal{A}$ proceeds by local search, may lead to suboptimal solutions
3. Representational : in many situations, f can not be represented by $\mathcal{H}$







Introduction to ensemble learning

Limitations of learning a single h

1. Statistical : if $\mathcal{D}$ is too small compared to $\mathcal{H}$, many h_t that performs equally on $\mathcal{D}$
2. Computational : if $\mathcal{A}$ proceeds by local search, may lead to suboptimal solutions
3. Representational : in many situations, f can not be represented by $\mathcal{H}$

Solution : Ensemble learning

1. Combining the h_t reduces the risk of choosing the wrong one
2. Learning from different "starting points" and combining is better
3. A weighted combination of several h_t expands $\mathcal{H}$

Introduction to ensemble learning

Generic principle of ensemble learning

- Produce several different $h_t, \forall t = 1, \dots, T$
- Combine these h_t with a combining operator Ω such that

$$\hat{y} = \Omega(h_1(\mathbf{x}), \dots, h_T(\mathbf{x})), \forall \mathbf{x} \in \mathcal{X}$$

For example :

- $\mathcal{Y} = \{-1, 1\}$ and $h(\mathbf{x}) \in [-1, 1], \forall t = 1, \dots, T, \forall \mathbf{x} \in \mathcal{X}$
- Split $\mathcal{D}$ into T disjoints subsets $\mathcal{D}_t$ and learn a h_t from each one
- Ω is typically the sign of the summed score :

$$\hat{y} = sign\left(\sum_{t=1}^{T} h_t(\mathbf{x})\right)$$

Key ingredients : accuracy and diversity

Accuracy vs Diversity :

1. A classifier is said to be accurate if it's better than random guessing, "at least"
2. The diversity between classifiers is high if their errors are mostly different

- Maximize both accuracy and diversity at the same time
- But these two objectives are generally opposed
- The challenge is to create diversity while maintaining sufficient individual accuracy

How to create diversity ?

1. Randomization in the learning algorithm

   ex. : Random init. of parameters (e.g. weights of NN)

2. Manipulate the training instances

   ex. : Use training subsets

3. Manipulate the features

   ex. : Use description subspaces

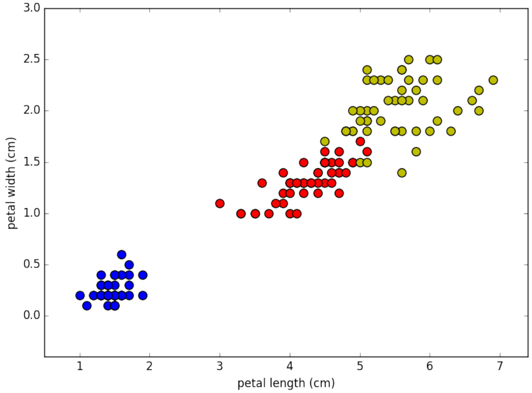
4. Manipulate the output values

   ex. : One-vs-One or One-vs-All strategies

Decision tree classifier

Decision tree classifier

Example : predict the iris species from the width and length of the petals

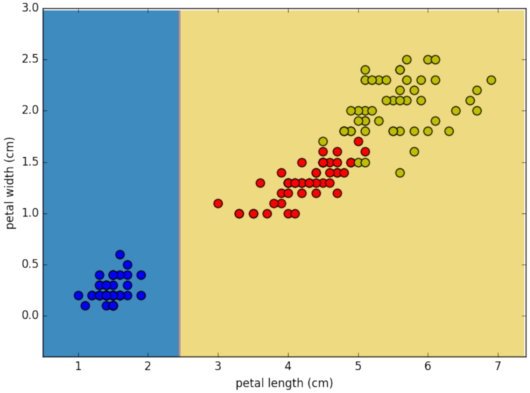


Petal width vs *Petal length* :

→ We build the classifier from these 2 features, with very simple and intuitive decision rules

Decision tree classifier

Example : predict the iris species from the width and length of the petals



If *Petal length* ≤ 2.45

$$h(x) = setosa$$

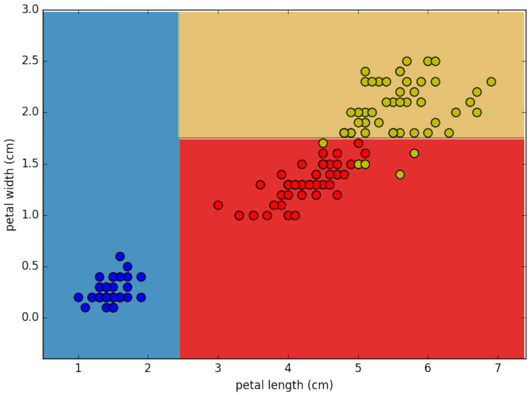
Else

$$h(x) = virginica$$

(or *versicolor*, difficult to decide)

Decision tree classifier

Example : predict the iris species from the width and length of the petals



If *Petal length* ≤ 2.45

$$h(\mathbf{x}) = \textit{setosa}$$

Else if *Petal width* ≥ 1.75

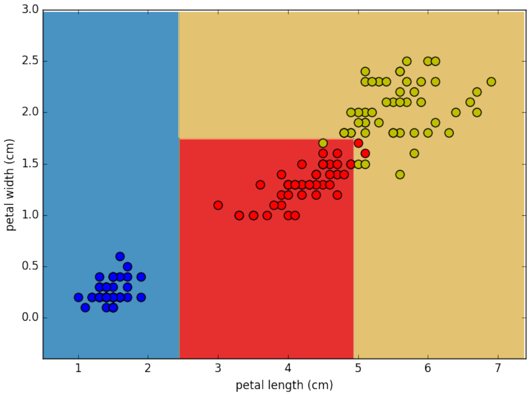
$$h(\mathbf{x}) = \textit{virginica}$$

Else

$$h(\mathbf{x}) = \textit{versicolor}$$

Decision tree classifier

Example : predict the iris species from the width and length of the petals



If *Petal length* ≤ 2.45

$$h(\mathbf{x}) = \mathit{setosa}$$

Else if *Petal width* ≥ 1.75

$$h(\mathbf{x}) = \mathit{virginica}$$

Else if *Petal length* ≤ 4.95

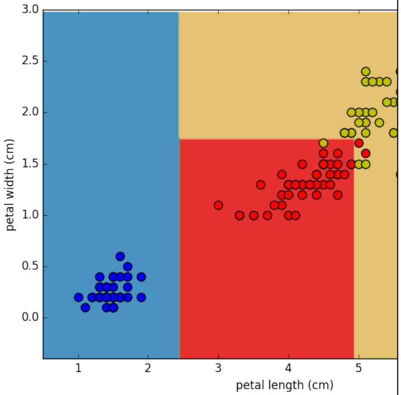
$$h(\mathbf{x}) = \mathit{versicolor}$$

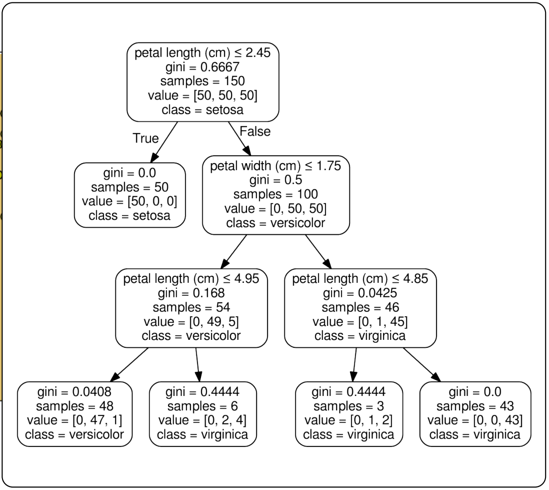
Else

$$h(\mathbf{x}) = \mathit{virginica}$$

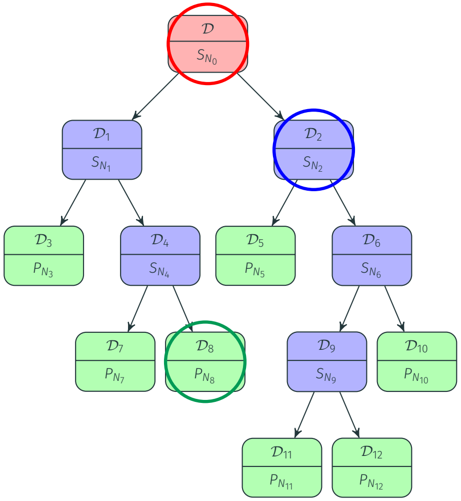
Decision tree classifier

Example : predict the iris species from the width and length of the petals





Decision tree classifier

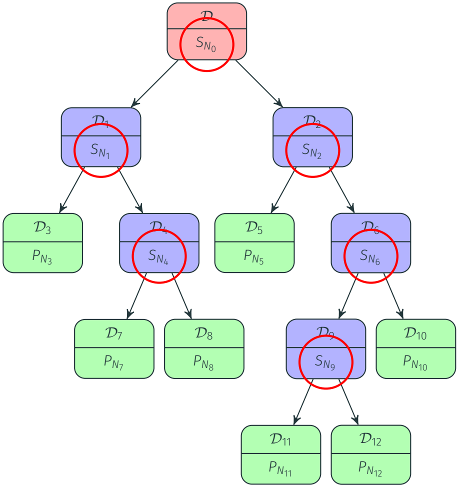


A tree is made up of nodes :

1. Root node or initial node
2. Splitting node or internal node
3. Leaf node or terminal node

The root node contains the whole training set $\mathcal{D}$, the other nodes contains subsets $\mathcal{D}_k$

Decision tree classifier



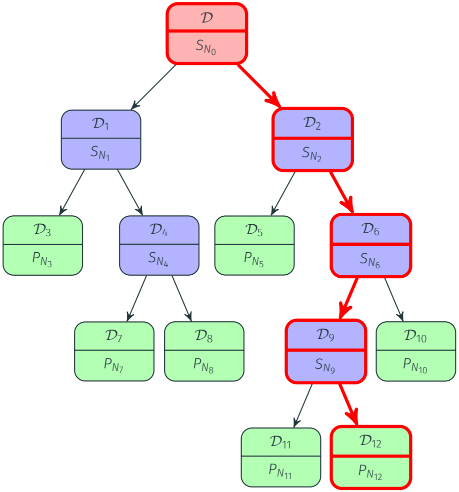
Root and splitting nodes contains splitting rules :

$$S_{N_i} : \mathcal{X} \rightarrow \{N_{left}, N_{right}\}$$

Most often, it is a test on the value of one feature

$$S_{N_i}(\mathbf{x}) = \begin{cases} N_{left} & if\ x^{(i)} \leq \alpha_i \\ N_{right} & otherwise \end{cases}$$

Decision tree classifier



Predict a new instance x :

1. Feed **x** to the root node
2. Evaluate S_{N_0}
3. Redirect to child node (N_2)
4. Evaluate S_{N_2}
5. and so on till reaching a leaf

Leaf node gives a prediction or a vector of class proba. :

$$P_{N_i} = \{\hat{P}(\lambda_1|\mathbf{x}), \hat{P}(\lambda_2|\mathbf{x}), \ldots, \hat{P}(\lambda_C|\mathbf{x})\}$$

Decision tree learning

Decision tree learning

Generic procedure for DT learning

1. Init. the root node N with $\mathcal{D}$
2. To split or not to split ?
    2.1 To split :
        2.1.1 Select a splitting rule S_N
        2.1.2 Build the child nodes N_{left} and N_{right} according to S_N
        2.1.3 Go to step 2 with $N = N_{left}$ and with $N = N_{right}$
    2.2 Not to split : assign a class to predict (or C class proba.) to N

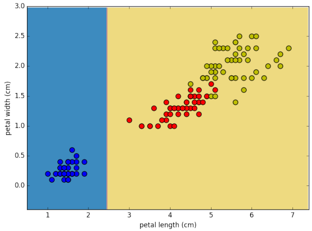
Questions to adress :

- How to select a splitting rule ? (step 2.1.1)
- How to decide if a node should be split or not ? (step 2)
- How to decide which class to predict at a given leaf ? (step 2.2)

Decision tree learning

Splitting rule based on a single feature $x^{(i)}$

- Find the $S_N = (x^{(i)}, \alpha_i)$ that best separates the instances according to their classes
- The classical procedure is an exhaustive search :



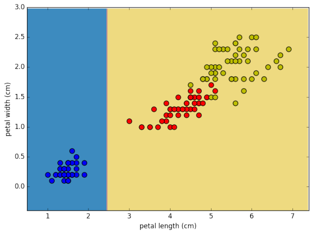
1. At node N_k, S_{N_k} must be set from $\mathcal{D}_k$ only
2. Try all $x^{(i)}$ and for each of them, every possible α_i in $\mathcal{D}_k$
3. Retain the $(x^{(i)}, \alpha_i)$ that optimizes the splitting criterion

Decision tree learning

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Splitting criterion

- The splitting criterion is a measure of the quality of the split
- Based on impurity : a node is said to be pure if all the instances belong to the same class



- For a given impurity measure I (here the Gini index)
- $I(\mathcal{D}_i) = 0.667$ when class distrib. is $\{50, 50, 50\}$
- $I(\mathcal{D}_i) = 0$ for left child since class distrib. is $\{50, 0, 0\}$

Splitting criterion

- For each candidate rule S_{N_i}, the criterion is the impurity gain ($\nearrow$) :

$$\Delta(\mathcal{D}_i, S_{N_i}) = I(\mathcal{D}_i) - \left(\frac{|\mathcal{D}_{left}|}{|\mathcal{D}_i|} I(\mathcal{D}_{left}) + \frac{|\mathcal{D}_{right}|}{|\mathcal{D}_i|} I(\mathcal{D}_{right}) \right)$$

- Two popular impurity measures :
  - *Gini index*

$$G(\mathcal{D}_i) = 1 - \sum_{k=1}^{C} \left(\frac{n_{k.}}{|\mathcal{D}_i|} \right)^2$$

  - *Entropy (Information Gain)*

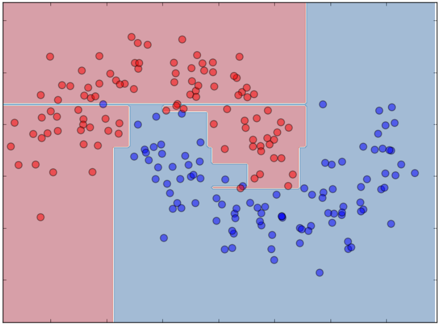
$$E(\mathcal{D}_i) = \sum_{k=1}^{C} -\frac{n_{k.}}{|\mathcal{D}_i|} log_2 \frac{n_{k.}}{|\mathcal{D}_i|}$$

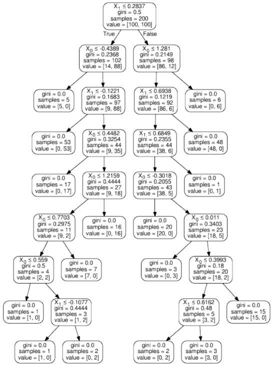
Early stopping and prediction assignment

1. Stop a branch from growing :
   - Never : a node N_k become a leaf node when it's pure
   - Based on threshold on depth, number of instances, impurity gain, etc.
2. which prediction to assign to the leaf node :
   - the majority class in $\mathcal{D}_k$
   - class counts in $\mathcal{D}_k$ + smoothing methods (recommended)
     Example : Laplace smoothing
     $$\hat{P}(\lambda_c|\mathbf{x}) = \frac{n_{c\cdot} + 1}{n_{\cdot\cdot} + C}$$

Decision tree learning

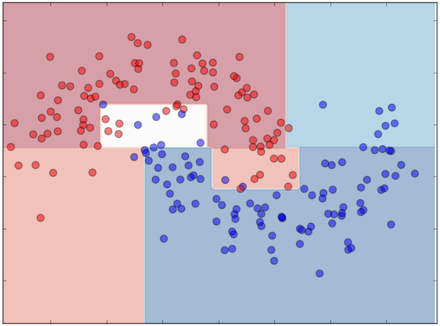
Early stopping and prediction assignment

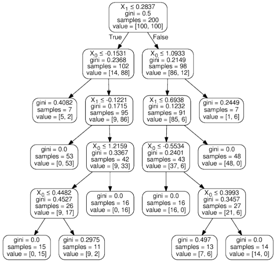



X₁ ≤ 0.2837 gini = 0.5 samples = 200 value = [100, 100]
True
False
X₀ ≤ -0.4389 gini = 0.2368 samples = 102 value = [14, 88]
X₀ ≤ 1.281 gini = 0.2149 samples = 98 value = [86, 12]
gini = 0.0 samples = 5 value = [5, 0]
X₁ ≤ -0.1221 gini = 0.1683 samples = 97 value = [9, 88]
X₁ ≤ 0.6938 gini = 0.1219 samples = 92 value = [86, 6]
gini = 0.0 samples = 6 value = [0, 6]
gini = 0.0 samples = 53 value = [0, 53]
X₀ ≤ 0.4482 gini = 0.3254 samples = 44 value = [9, 35]
X₁ ≤ 0.6849 gini = 0.2355 samples = 44 value = [38, 6]
gini = 0.0 samples = 48 value = [48, 0]
gini = 0.0 samples = 17 value = [0, 17]
X₀ ≤ 1.2159 gini = 0.4444 samples = 27 value = [9, 18]
X₀ ≤ -0.3018 gini = 0.2055 samples = 43 value = [38, 5]
gini = 0.0 samples = 1 value = [0, 1]
X₀ ≤ 0.7703 gini = 0.2975 samples = 11 value = [9, 2]
gini = 0.0 samples = 16 value = [0, 16]
gini = 0.0 samples = 20 value = [20, 0]
X₀ ≤ 0.011 gini = 0.3403 samples = 23 value = [18, 5]
X₀ ≤ 0.559 gini = 0.5 samples = 4 value = [2, 2]
gini = 0.0 samples = 7 value = [7, 0]
gini = 0.0 samples = 3 value = [0, 3]
X₀ ≤ 0.3993 gini = 0.18 samples = 20 value = [18, 2]
gini = 0.0 samples = 1 value = [1, 0]
X₁ ≤ -0.1077 gini = 0.4444 samples = 3 value = [1, 2]
X₁ ≤ 0.6162 gini = 0.48 samples = 5 value = [3, 2]
gini = 0.0 samples = 15 value = [15, 0]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 2 value = [0, 2]
gini = 0.0 samples = 2 value = [0, 2]
gini = 0.0 samples = 3 value = [3, 0]

Max. depth tree always have 0% training error but very poor generalization performances

Early stopping and prediction assignment





Early stopping can help but one can also cut some branches a posteriori (pruning)

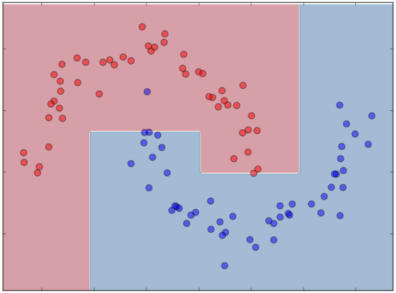
Ensemble of decision trees

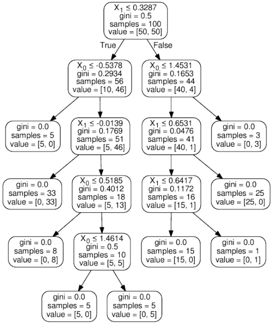
Ensemble of decision trees

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Unstable classifier

A classifier is said to be unstable if small changes in the learning set involves significant changes in the resulting model.



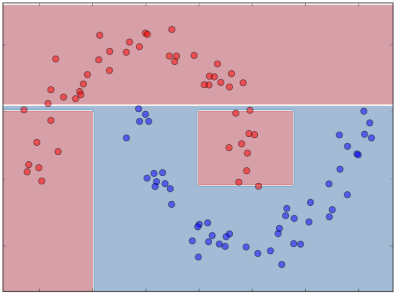


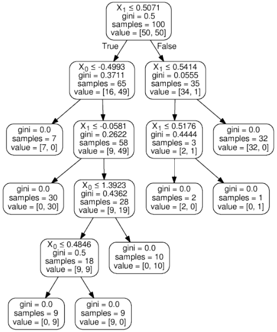
$X_1 \leq 0.3287$ gini = 0.5 samples = 100 value = [50, 50]
True
False
$X_0 \leq -0.5378$ gini = 0.2934 samples = 56 value = [10, 46]
$X_0 \leq 1.4531$ gini = 0.1653 samples = 44 value = [40, 4]
gini = 0.0 samples = 5 value = [5, 0]
$X_1 \leq -0.0139$ gini = 0.1769 samples = 51 value = [5, 46]
$X_1 \leq 0.6531$ gini = 0.0476 samples = 41 value = [40, 1]
gini = 0.0 samples = 3 value = [0, 3]
gini = 0.0 samples = 33 value = [0, 33]
$X_0 \leq 0.5185$ gini = 0.4012 samples = 18 value = [5, 13]
$X_1 \leq 0.6417$ gini = 0.1172 samples = 16 value = [15, 1]
gini = 0.0 samples = 25 value = [25, 0]
gini = 0.0 samples = 8 value = [0, 8]
$X_0 \leq 1.4614$ gini = 0.5 samples = 10 value = [5, 5]
gini = 0.0 samples = 15 value = [15, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 5 value = [5, 0]
gini = 0.0 samples = 5 value = [0, 5]

Ensemble of decision trees

Unstable classifier

A classifier is said to be unstable if small changes in the learning set involves significant changes in the resulting model.





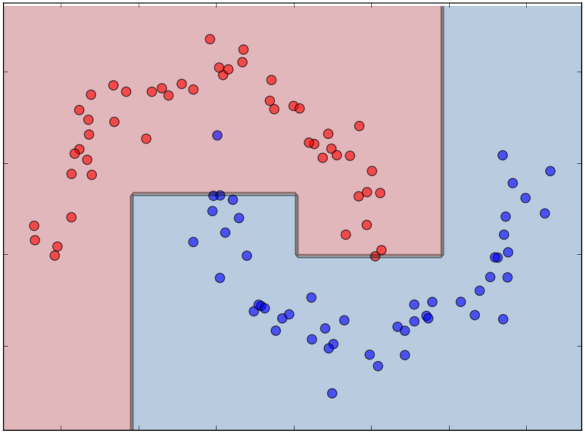
$X_1 \leq 0.5071$ gini = 0.5 samples = 100 value = [50, 50]
True
False
$X_0 \leq -0.4993$ gini = 0.3711 samples = 65 value = [16, 49]
$X_1 \leq 0.5414$ gini = 0.0555 samples = 35 value = [34, 1]
gini = 0.0 samples = 7 value = [7, 0]
$X_1 \leq -0.0581$ gini = 0.2622 samples = 58 value = [9, 49]
$X_1 \leq 0.5176$ gini = 0.4444 samples = 3 value = [2, 1]
gini = 0.0 samples = 32 value = [32, 0]
gini = 0.0 samples = 30 value = [0, 30]
$X_0 \leq 1.3923$ gini = 0.4362 samples = 28 value = [9, 19]
gini = 0.0 samples = 2 value = [2, 0]
gini = 0.0 samples = 1 value = [0, 1]
$X_0 \leq 0.4846$ gini = 0.5 samples = 18 value = [9, 9]
gini = 0.0 samples = 10 value = [0, 10]
gini = 0.0 samples = 9 value = [0, 9]
gini = 0.0 samples = 9 value = [9, 0]

Ensemble of decision trees

- Instability leads to a very high variance (and poor generalization performances)
- Pruning methods are generally not sufficient
- Better solution are ensembles of decision trees (decision forests)
- Why ? Because instability is a real asset to create diversity

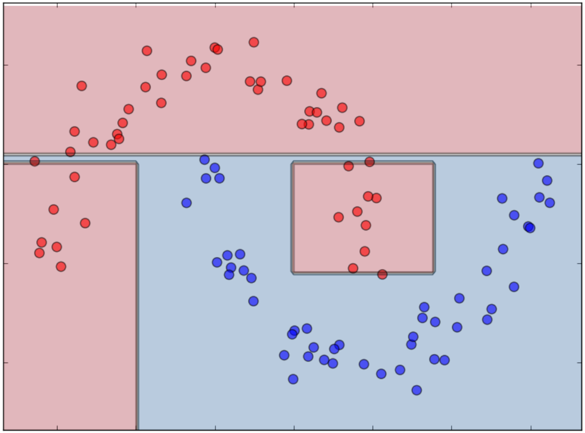
Ensemble of decision trees

A first decision tree built (without pruning) on 100 training instances (50 in each class)



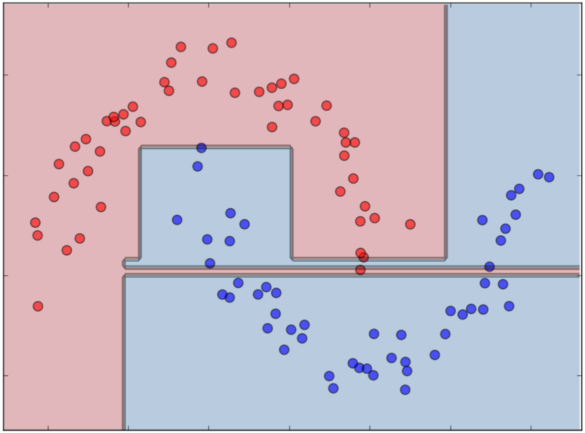
Ensemble of decision trees

A second tree built on 100 additional training instances (from the same distribution)



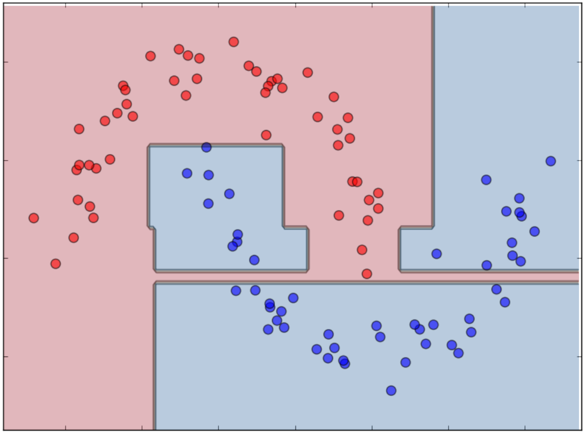
Ensemble of decision trees

Third tree...



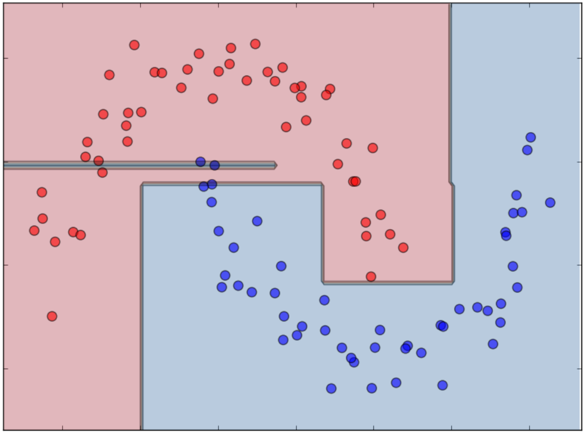
Ensemble of decision trees

Fourth...



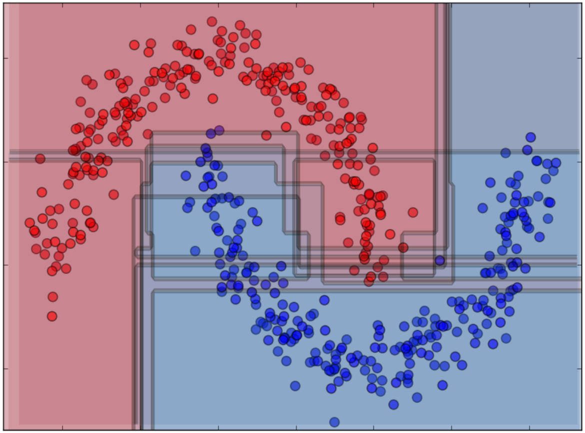
Ensemble of decision trees

And fifth.



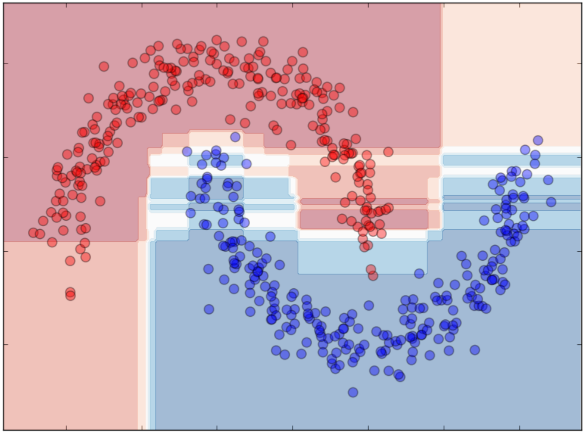
Ensemble of decision trees

All of them in the same plot (with their respective training instances)



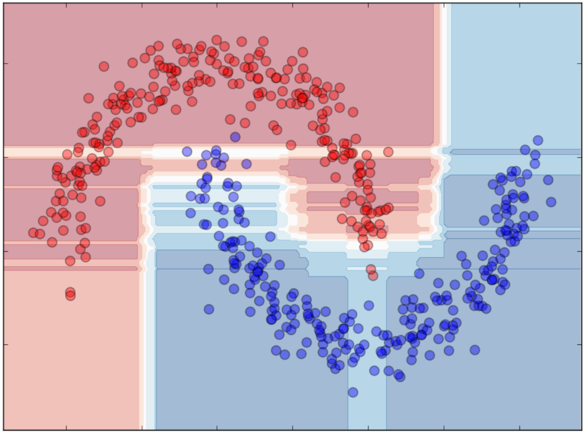
Ensemble of decision trees

And what about making them vote for the final prediction ?



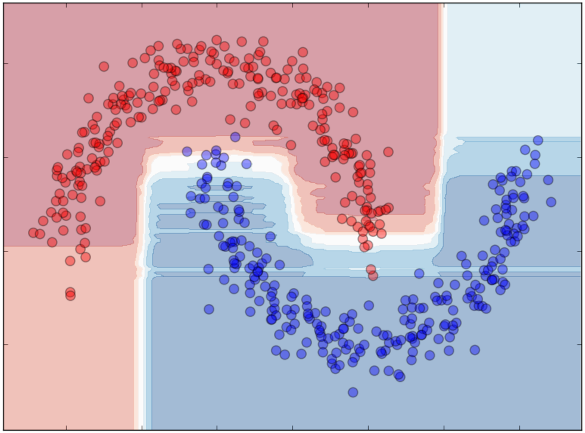
Ensemble of decision trees

And by combining 10 trees ?



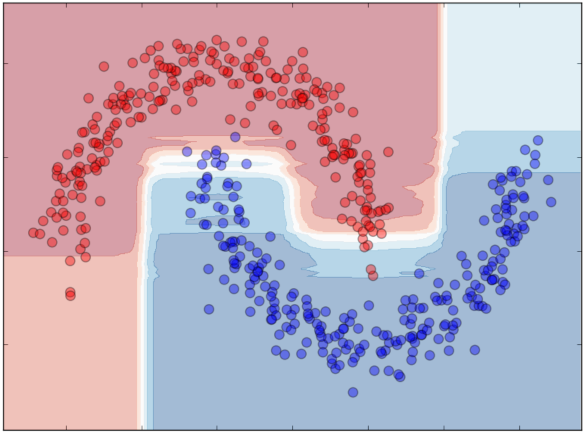
Ensemble of decision trees

And 100 trees ?



Ensemble of decision trees

1000 ?!! ($<$ 4 secondes for training)



Ensemble of decision trees

Takeaways

DT are...

- easy to understand and interpret
- flexible (can handle many different types of data)
- reasonnably fast with reasonnably large datasets

But...

- do not scale up (”non reasonnably” large datasets)
- are unstable (high variance)
- are not very accurate in general

However, they are ideal for ensemble methods