Machine Learning

Nathalie Villa-Vialaneix - nathalie.villa@univ-paris1.fr
http://www.nathalievilla.org

IUT STID (Carcassonne) & SAMM (Université Paris 1)

Formation INRA, Niveau 3

Outline

Outline

Background

- **Purpose**: predict Y from X;

Background

- **Purpose**: predict Y from X;
- **What we have**: n observations of (X, Y), (x_1, y_1), ..., (x_n, y_n);

Background

- **Purpose**: predict Y from X;
- **What we have**: n observations of (X, Y), (x_1, y_1), ..., (x_n, y_n);
- **What we want**: estimate unknown Y from new X: x_{n+1}, ..., x_m.

Background

- **Purpose**: predict Y from X;
- **What we have**: n observations of (X, Y), (x_1, y_1), ..., (x_n, y_n);
- **What we want**: estimate unknown Y from new X: x_{n+1}, ..., x_m.

X can be:

- numeric variables;
- or factors;
- or a combination of numeric variables and factors.

Background

- **Purpose**: predict Y from X;
- **What we have**: n observations of (X, Y), (x_1, y_1), ..., (x_n, y_n);
- **What we want**: estimate unknown Y from new X: x_{n+1}, ..., x_m.

X can be:

- numeric variables;
- or factors;
- or a combination of numeric variables and factors.

Y can be:

- a numeric variable ($Y \in \mathbb{R}$) $\Rightarrow$ **(supervised) regression** *régression*;
- a factor $\Rightarrow$ **(supervised) classification** *discrimination*.

Basics

From $(x_i, y_i)_i$, definition of a **machine**, Φ^n s.t.:

$$\hat{y}_{\mathrm{new}} = \Phi^n(x_{\mathrm{new}}).$$

Basics

From $(x_i, y_i)_i$, definition of a **machine**, Φ^n s.t.:

$$\hat{y}_{\text{new}} = \Phi^n(x_{\text{new}}).$$

- if Y is numeric, Φ^n is called a **regression function** *fonction de classification*;
- if Y is a factor, Φ^n is called a **classifier** *classifieur*;

Basics

From $(x_i, y_i)_i$, definition of a **machine**, Φ^n s.t.:

$$\hat{y}_{\text{new}} = \Phi^n(x_{\text{new}}).$$

- if Y is numeric, Φ^n is called a **regression function** *fonction de classification*;
- if Y is a factor, Φ^n is called a **classifier** *classifieur*;

Φ^n is said to be **trained** or **learned** from the observations $(x_i, y_i)_i$.

Basics

From $(x_i, y_i)_i$, definition of a **machine**, Φ^n s.t.:

$$\hat{y}_{\text{new}} = \Phi^n(x_{\text{new}}).$$

- if Y is numeric, Φ^n is called a **regression function** *fonction de classification*;
- if Y is a factor, Φ^n is called a **classifier** *classifieur*;

Φ^n is said to be **trained** or **learned** from the observations $(x_i, y_i)_i$.

Desirable properties

- **accuracy to the observations**: predictions made on **known** data are close to observed values;

Basics

From $(x_i, y_i)_i$, definition of a **machine**, Φ^n s.t.:

$$\hat{y}_{\text{new}} = \Phi^n(x_{\text{new}}).$$

- if Y is numeric, Φ^n is called a **regression function** *fonction de classification*;
- if Y is a factor, Φ^n is called a **classifier** *classifieur*;

Φ^n is said to be **trained** or **learned** from the observations $(x_i, y_i)_i$.

Desirable properties

- **accuracy to the observations**: predictions made on **known** data are close to observed values;
- **generalization ability**: predictions made on **new** data are also accurate.

Basics

From $(x_i, y_i)_i$, definition of a **machine**, Φ^n s.t.:

$$\hat{y}_{\text{new}} = \Phi^n(x_{\text{new}}).$$

- if Y is numeric, Φ^n is called a **regression function** *fonction de classification*;
- if Y is a factor, Φ^n is called a **classifier** *classifieur*;

Φ^n is said to be **trained** or **learned** from the observations $(x_i, y_i)_i$.

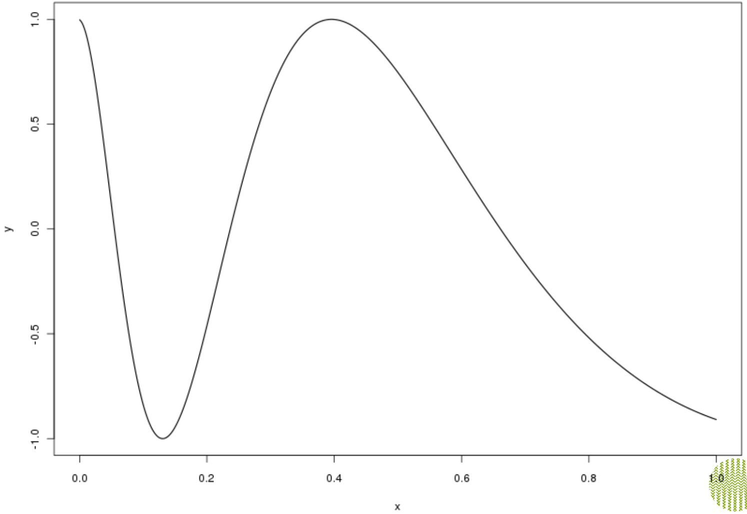
Desirable properties

- **accuracy to the observations**: predictions made on **known** data are close to observed values;
- **generalization ability**: predictions made on **new** data are also accurate.

Conflicting objectives!!

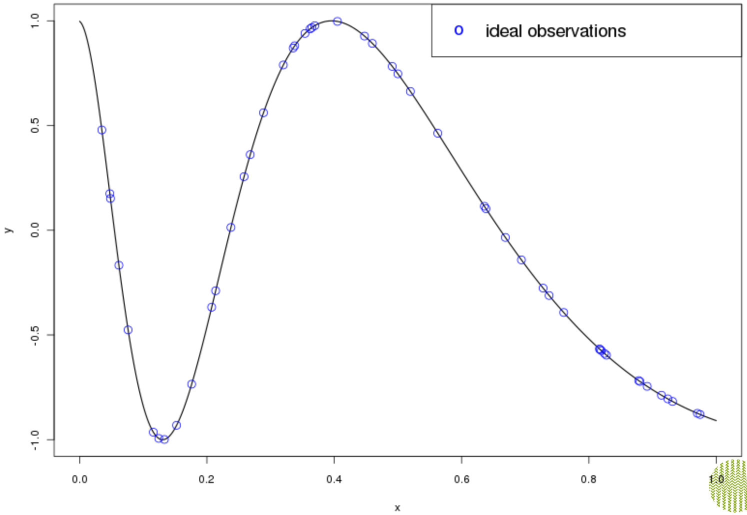
Underfitting/Overfitting *sous/sur - apprentissage*

Function $x \to y$ to be estimated



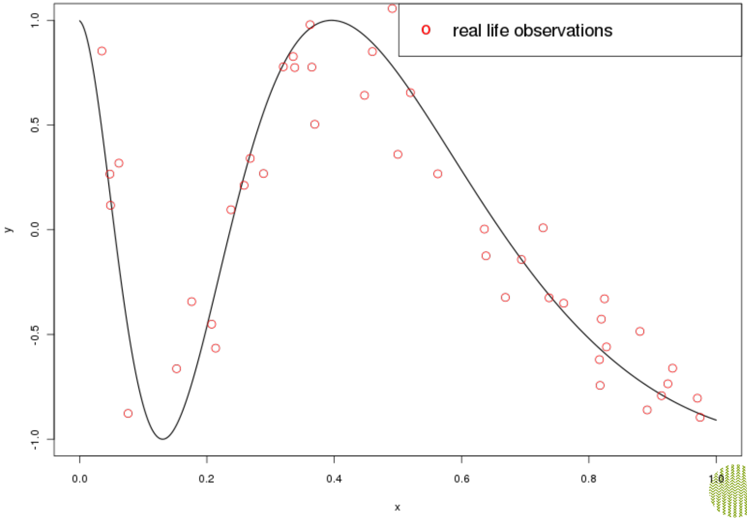
Underfitting/Overfitting *sous/sur - apprentissage*

Observations we might have



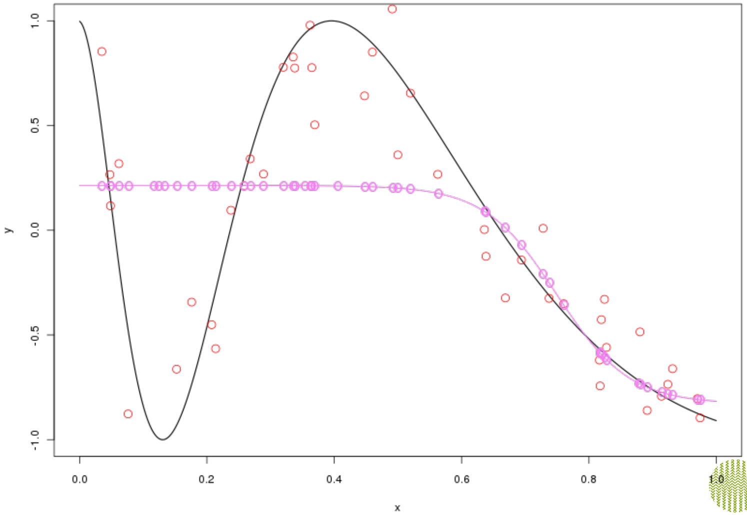
Underfitting/Overfitting *sous/sur - apprentissage*

Observations we do have



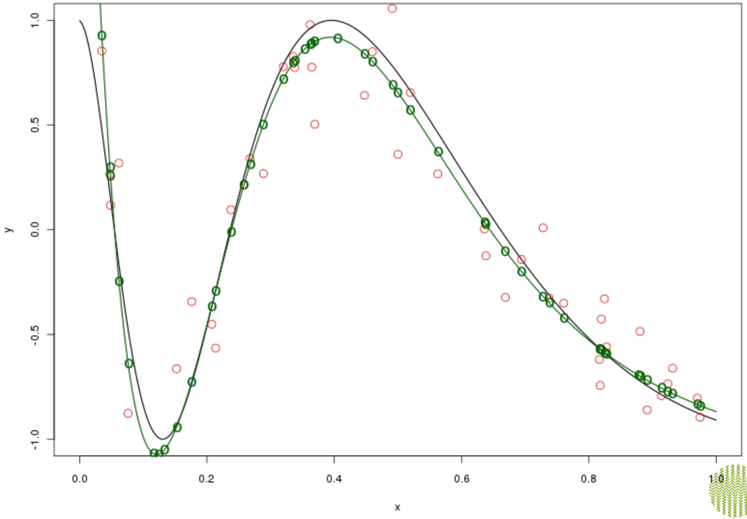
Underfitting/Overfitting *sous/sur - apprentissage*

First estimation from the observations: **underfitting**



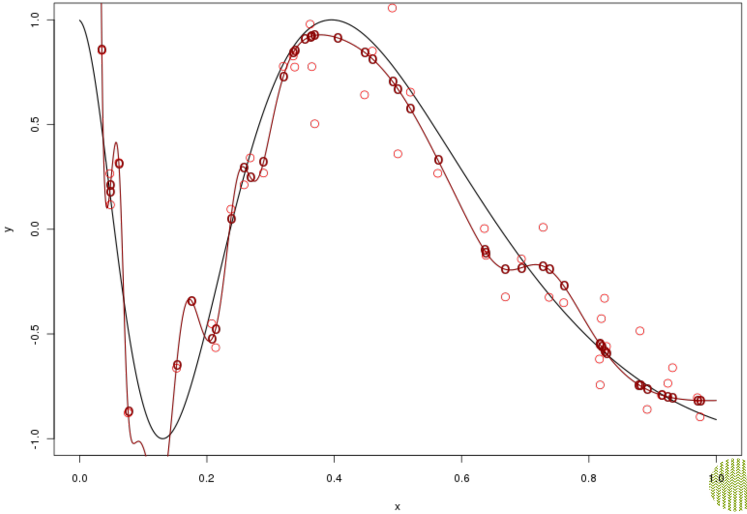
Underfitting/Overfitting *sous/sur - apprentissage*

Second estimation from the observations: accurate estimation

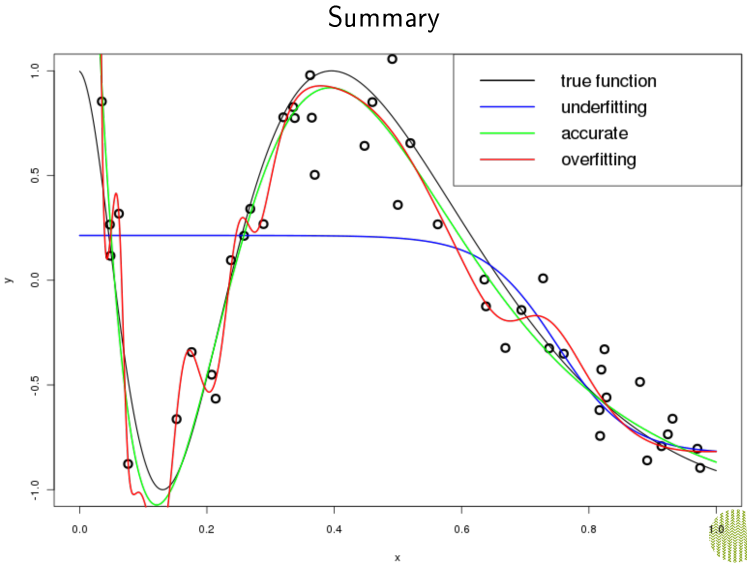


Underfitting/Overfitting *sous/sur - apprentissage*

Third estimation from the observations: **overfitting**



Underfitting/Overfitting *sous/sur - apprentissage*



Errors

- **training error** (measures the accuracy to the observations)

Errors

- **training error** (measures the accuracy to the observations)
  - if y is a factor: **misclassification rate**

$$\frac{\sharp\{\hat{y}_i \neq y_i,\ i = 1, \ldots, n\}}{n}$$

Errors

- **training error** (measures the accuracy to the observations)
  - if y is a factor: **misclassification rate**

$$\frac{\sharp\{\hat{y}_i \neq y_i,\ i = 1, \ldots, n\}}{n}$$

  - if y is numeric: **mean square error (MSE)**

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

Errors

- **training error** (measures the accuracy to the observations)
  - if y is a factor: **misclassification rate**

$$\frac{\sharp\{\hat{y}_i \neq y_i,\ i = 1, \ldots, n\}}{n}$$

  - if y is numeric: **mean square error (MSE)**

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

or root mean square error (RMSE) or pseudo-R^2: $1 - \mathrm{MSE}/\mathrm{Var}((y_i)_i)$

Errors

- **training error** (measures the accuracy to the observations)
  - if y is a factor: **misclassification rate**

$$\frac{\sharp\{\hat{y}_i \neq y_i,\ i = 1, \ldots, n\}}{n}$$

  - if y is numeric: **mean square error (MSE)**

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

    or root mean square error (RMSE) or pseudo-R^2: $1 - \mathrm{MSE}/\mathrm{Var}((y_i)_i)$
- **test error**: a way to prevent overfitting (estimates the generalization error)

Errors

- **training error** (measures the accuracy to the observations)
  - if y is a factor: **misclassification rate**

$$\frac{\sharp\{\hat{y}_i \neq y_i,\ i = 1, \ldots, n\}}{n}$$

  - if y is numeric: **mean square error (MSE)**

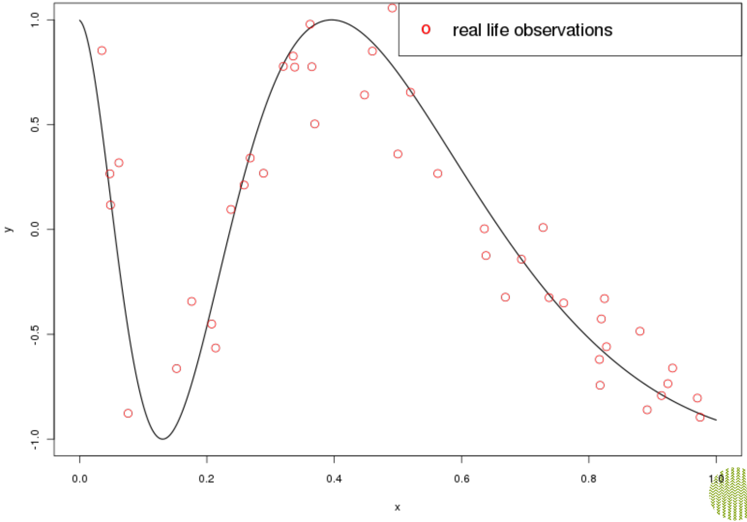
$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

  or root mean square error (RMSE) or pseudo-R^2: $1 - \mathrm{MSE}/\mathrm{Var}((y_i)_i)$
- **test error**: a way to prevent overfitting (estimates the generalization error)
  1. split the data into training/test sets (usually 80%/20%)
  2. train Φ^n from the training dataset
  3. calculate the test error from the remaining data

simple validation

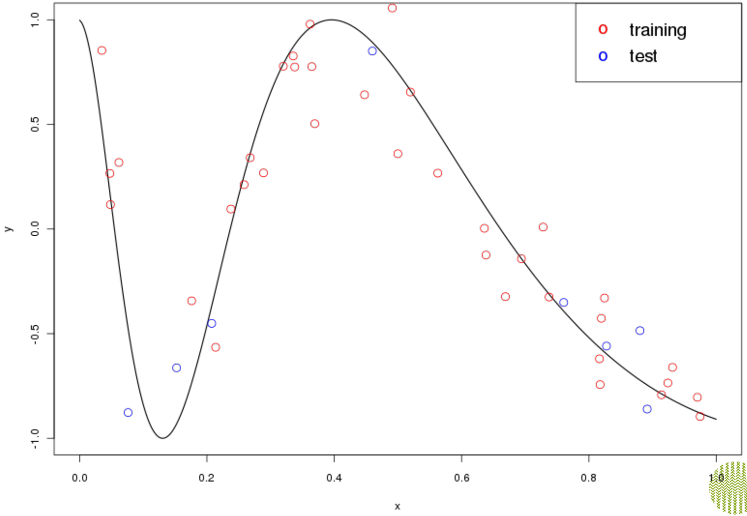
Example

Observations



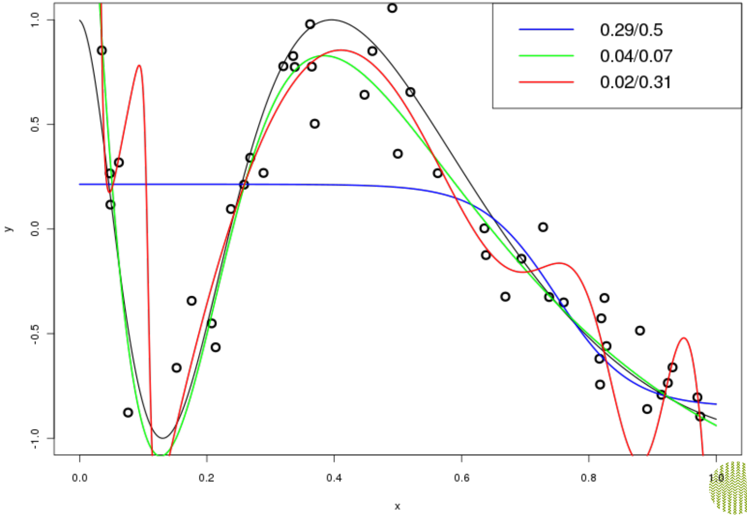
Example

Training/Test datasets



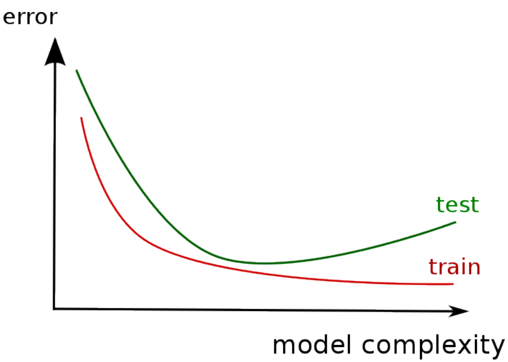
Example

Training/Test errors



Example

Summary



Linear vs Nonparametric

Linear methods:

$$Y = \beta^T X + \epsilon$$

(a priori on the type of link between X and Y)

Linear vs Nonparametric

Linear methods:

$$Y = \beta^T X + \epsilon$$

(a priori on the type of link between X and Y)
Here: nonparametric methods:

$$Y = \Phi(X) + \epsilon$$

where Φ is totally unknown.

Linear vs Nonparametric

Linear methods:

$$Y = \beta^T X + \epsilon$$

(a priori on the type of link between X and Y)

Here: nonparametric methods:

$$Y = \Phi(X) + \epsilon$$

where Φ is totally unknown.

(ML) Objective: Build Φ^n from the observations such that its generalization error $\mathbb{E}L\Phi^n$ is (asymptotically) optimal.

Example (regression framework)

$$\mathbb{E}L\Phi^n := \mathbb{E}\left[(\Phi^n(X) - Y)^2\right] \xrightarrow{n\rightarrow+\infty} \inf_{\Phi} EL\Phi$$

whatever (X, Y) distribution.

Use case description

Data kindly provided by Laurence Liaubet described in [Liaubet et al., 2011]:

- microarray data: expression of 272 selected genes over 56 individuals (pigs);
- a phenotype of interest (muscle pH) measured over the 56 invididuals (numerical variable).

file 1: genes expressions
file 2: muscle pH

Outline

Basics [Bishop, 1995]

Common properties

- (artificial) **"Neural networks"**: general name for supervised and unsupervised methods developed in analogy to the brain;

Basics [Bishop, 1995]

Common properties

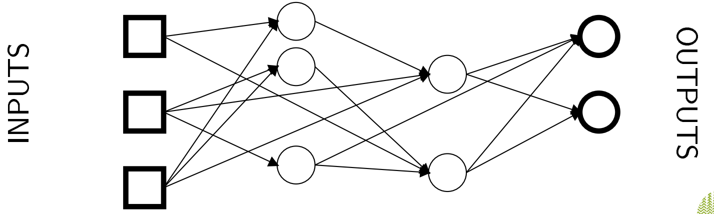
- (artificial) **"Neural networks"**: general name for supervised and unsupervised methods developed in analogy to the brain;
- **combination** (network) of **simple elements** (neurons).

Basics [Bishop, 1995]

Common properties

- (artificial) **"Neural networks"**: general name for supervised and unsupervised methods developed in analogy to the brain;
- **combination** (network) of **simple elements** (neurons).

Example of graphical representation:



Main features

A neural network is defined by:

1. the network structure;
2. the neuron type.

Main features

A neural network is defined by:

1. the network structure;
2. the neuron type.

Standard examples

- **Multilayer perceptrons** (MLP) *Perceptron multi-couches*: dedicated to supervised problems (classification and regression);

Main features

A neural network is defined by:

1. the network structure;
2. the neuron type.

Standard examples

- **Multilayer perceptrons** (MLP) *Perceptron multi-couches*: dedicated to supervised problems (classification and regression);
- **Radial basis function networks** (RBF): same purpose but based on local smoothing;

Main features

A neural network is defined by:

1. the network structure;
2. the neuron type.

Standard examples

- **Multilayer perceptrons** (MLP) *Perceptron multi-couches*: dedicated to supervised problems (classification and regression);
- **Radial basis function networks** (RBF): same purpose but based on local smoothing;
- **Self-organizing maps** (SOM): dedicated to unsupervised problems (clustering), self-organized;
- ...

MLP: Advantages/Drawbacks

Advantages

- **classification OR regression** (i.e., Y can be a numeric variable or a factor);
- **non parametric** method: no prior assumption needed;
- **accurate** (universal approximation).

MLP: Advantages/Drawbacks

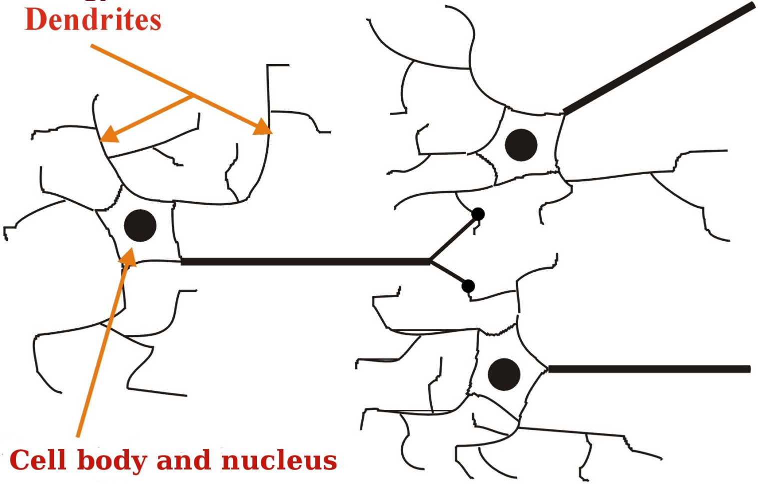
Advantages

- **classification OR regression** (i.e., Y can be a numeric variable or a factor);
- **non parametric** method: no prior assumption needed;
- **accurate** (universal approximation).

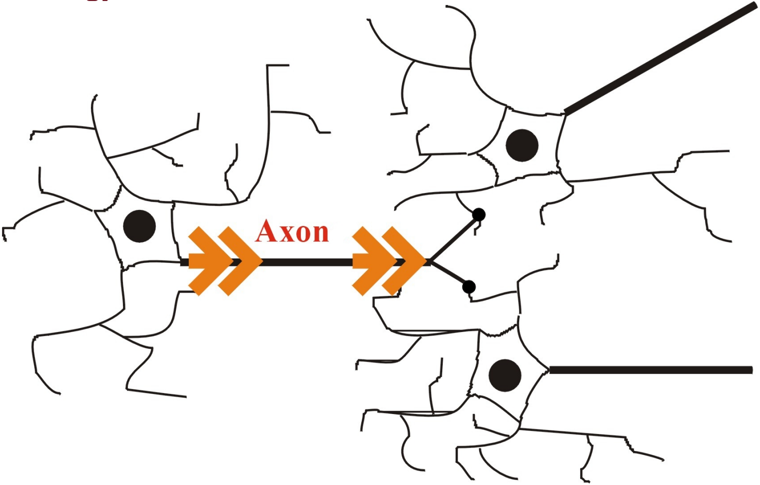
Drawbacks

- **hard to train** (high computational cost, especially when d is large);
- **overfit easily**;
- **"black box"** models (hard to interpret)

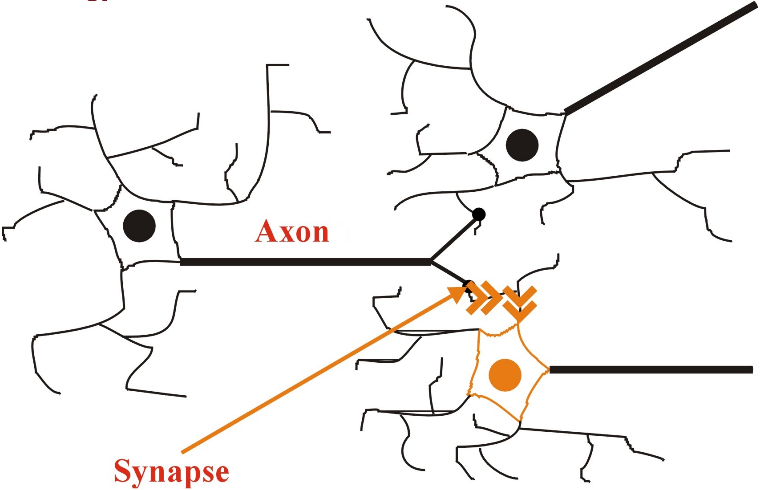
Analogy to the brain



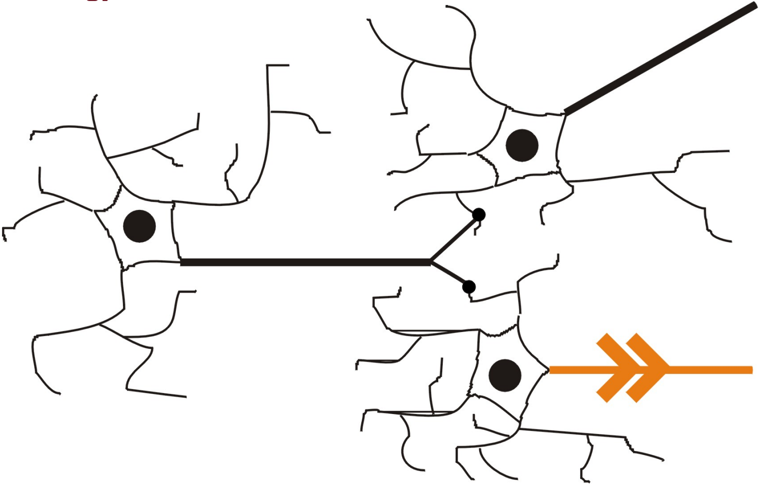
Analogy to the brain



Analogy to the brain

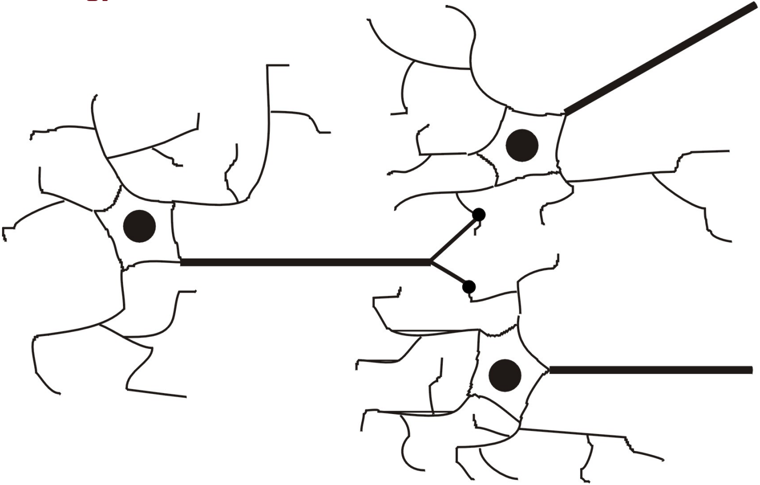


Analogy to the brain



If $\sum >$ **activation threshold** then

Analogy to the brain



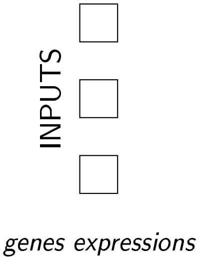
If $\sum <$ **activation threshold** then

(artificial) Perceptron

Layers

- MLP have one input layer (X values), one output layer (Y values) and several **hidden layers** (only 1 is necessary);
- no connections within a layer;
- connections between two consecutive layers (feedforward).

Example:

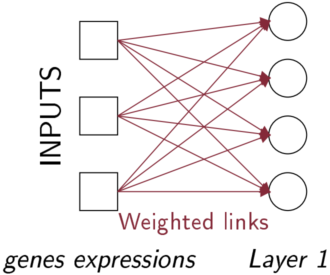


(artificial) Perceptron

Layers

- MLP have one input layer (X values), one output layer (Y values) and several **hidden layers** (only 1 is necessary);
- no connections within a layer;
- connections between two consecutive layers (feedforward).

Example:

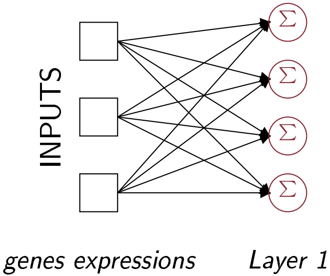


(artificial) Perceptron

Layers

- MLP have one input layer (X values), one output layer (Y values) and several **hidden layers** (only 1 is necessary);
- no connections within a layer;
- connections between two consecutive layers (feedforward).

Example:

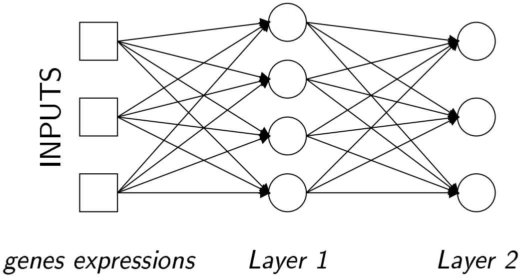


(artificial) Perceptron

Layers

- MLP have one input layer (X values), one output layer (Y values) and several **hidden layers** (only 1 is necessary);
- no connections within a layer;
- connections between two consecutive layers (feedforward).

Example:



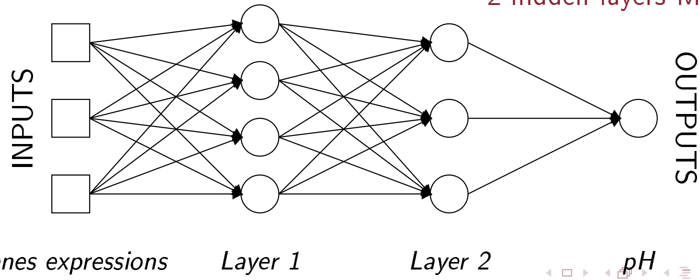
(artificial) Perceptron

Layers

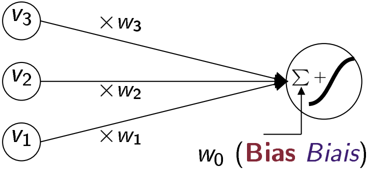
- MLP have one input layer (X values), one output layer (Y values) and several **hidden layers** (only 1 is necessary);
- no connections within a layer;
- connections between two consecutive layers (feedforward).

Example:

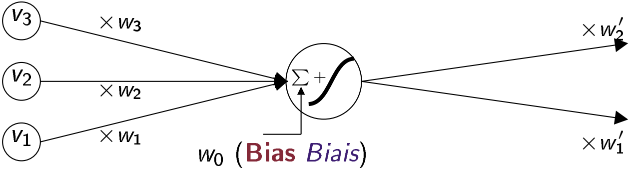
2 hidden layers MLP



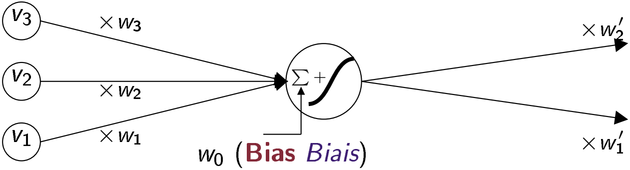
A neuron



A neuron

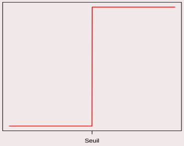

v_3
v_2
v_1
$\times w_3$
$\times w_2$
$\times w_1$
$\Sigma +$
w_0 (**Bias** *Biais*)
$\times w'_2$
$\times w'_1$

A neuron



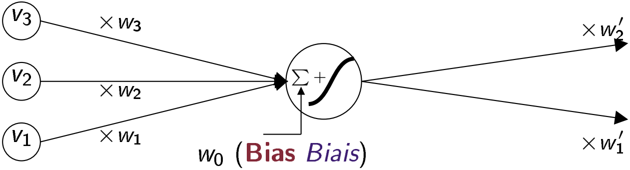
Standard activation functions *fonctions de lien / d'activation*

Biologically inspired: **Heaviside function**



$$S(t) = \begin{cases} 0 & \text{if } t < \text{threshold;} \\ 1 & \text{if not.} \end{cases}$$

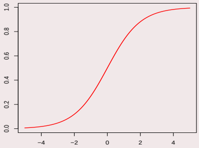
A neuron



Standard activation functions

Main issue with the Heaviside function: not continuous!

Logistic function

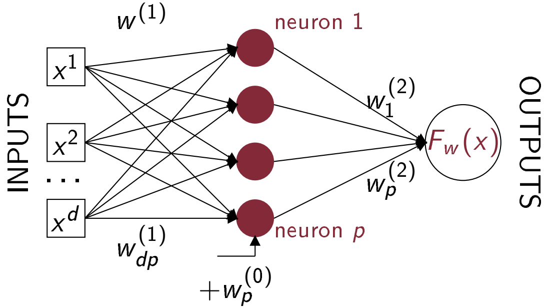


$$S(t) = \frac{1}{1+e^{-t}}$$

Summary

If Y is numeric, **linear output**:

$$\forall x \in \mathbb{R}^d,\ F_w(x) = \sum_{j=1}^{p} w_j^{(2)} \mathcal{S}\left(\sum_{k=1}^{d} w_{kj}^{(1)} x^k + w_j^{(0)}\right).$$



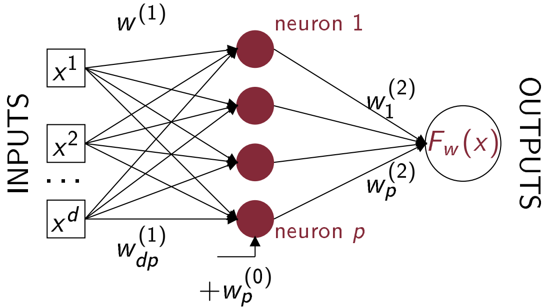
No analytical expression!!

Summary

If Y is a factor, **logistic output**:

$$\forall\, x \in \mathbb{R}^d,\ \mathbb{P}(X = C|x = x) \simeq F_w(x) = \mathcal{S}\left[\sum_{j=1}^{p} w_j^{(2)} \mathcal{S}\left(\sum_{k=1}^{d} w_{kj}^{(1)} x^k + w_j^{(0)}\right)\right]$$

(with a maximum probability rule for the final classification)



No analytical expression!!

Universal approximation

[Hornik et al., 1989] (among others)

For any given Φ, smooth enough and any precision ϵ, there exists a **one-hidden layer** perceptron (with sigmoïd activation functions) that approximates Φ with a precision at most ϵ.

Learning weights

p is given

Chose w s.t.:

$$w^n = \arg\min_w \frac{1}{n}\sum_{i=1}^n (y_i - F_w(x_i))^2 .$$

(MSE minimization)

Learning weights

p is given

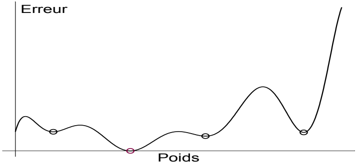
Chose w s.t.:

$$w^n = \arg\min_w \frac{1}{n}\sum_{i=1}^{n}(y_i - F_w(x_i))^2 .$$

(MSE minimization)

Main issues:

1. no exact solution ($\Rightarrow$ approximation algorithms, e.g., Newton's method + backpropagation principle): **local minima**;



Learning weights

p is given

Chose w s.t.:

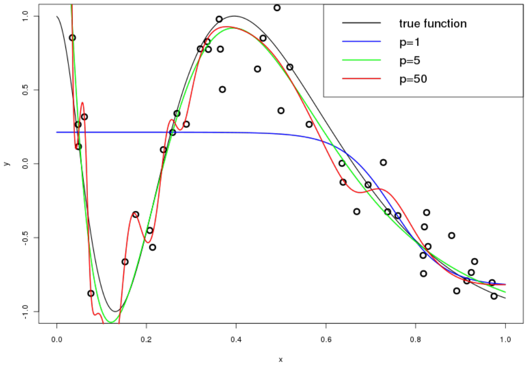
$$w^n = \arg\min_w \frac{1}{n}\sum_{i=1}^{n}(y_i - F_w(x_i))^2 .$$

(MSE minimization)

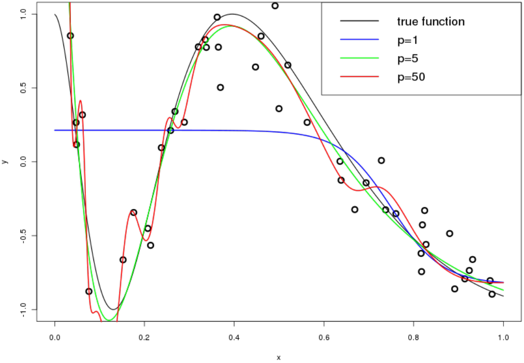
Main issues:

1. no exact solution ($\Rightarrow$ approximation algorithms, e.g., Newton's method + backpropagation principle): **local minima**;
2. **overfitting**: the larger p is, the more flexible the perceptron is and the more it can overfit the data.

Overfitting



Overfitting



Weight decay can help improve the generalization ability:

$$w^n = \arg\min_w \frac{1}{n}\sum_{i=1}^{n}(y_i - F_w(x_i))^2 + \lambda\|w\|^2$$

Tuning p and λ

p and λ are called **hyperparameters** *hyper-paramètres* (not learned from the data but **tuned**).

Tuning p and λ

p and λ are called **hyperparameters** *hyper-paramètres* (not learned from the data but **tuned**).

- **grid search** using a **simple validation**;

Tuning p and λ

p and λ are called **hyperparameters** *hyper-paramètres* (not learned from the data but **tuned**).

- **grid search** using a **simple validation**;
- **grid search** using a (K-fold) **cross validation** (better but computationally expensive when K is large).

Cross validation

Algorithm

1: Set the grid search for p, $\mathcal{G}_p$, and λ, $\mathcal{G}_\lambda$
2: Split the data into K groups
3: **for** $p \in \mathcal{G}_p$ and $\lambda \in \mathcal{G}_\lambda$ **do**
4:     **for** group $= 1..K$ **do**
5:         Train model$_{p,\lambda,\mathrm{group}}$ without observations in "group"
6:         Test error, $\mathrm{MSE}_{p,\lambda,\mathrm{group}}$ for observations in "group"
7:     **end for**
8:     Average $\mathrm{MSE}_{p,\lambda,\mathrm{group}}$ over "group" $\Rightarrow \mathrm{MSE}_{p,\lambda}$
9: **end for**
10: Select p and λ with minimum $\mathrm{MSE}_{p,\lambda}$

Cross validation

Algorithm

Set the grid search for p, $\mathcal{G}_p$, and λ, $\mathcal{G}_\lambda$
Split the data into K groups
for $p \in \mathcal{G}_p$ and $\lambda \in \mathcal{G}_\lambda$ **do**
    for group $= 1..K$ **do**
        Train model$_{p,\lambda,\mathrm{group}}$ without observations in "group"
        Test error, $\mathrm{MSE}_{p,\lambda,\mathrm{group}}$ for observations in "group"
    end for
    Average $\mathrm{MSE}_{p,\lambda,\mathrm{group}}$ over "group" $\Rightarrow \mathrm{MSE}_{p,\lambda}$
end for
Select p and λ with minimum $\mathrm{MSE}_{p,\lambda}$

n-fold cross validation is called **Leave-One-Out** (LOO).

Cross validation

Algorithm

Set the grid search for p, $\mathcal{G}_p$, and λ, $\mathcal{G}_\lambda$
Split the data into K groups
for $p \in \mathcal{G}_p$ and $\lambda \in \mathcal{G}_\lambda$ **do**
    for group $= 1..K$ **do**
        Train model$_{p,\lambda,\mathrm{group}}$ without observations in "group"
        Test error, $\mathrm{MSE}_{p,\lambda,\mathrm{group}}$ for observations in "group"
    end for
    Average $\mathrm{MSE}_{p,\lambda,\mathrm{group}}$ over "group" $\Rightarrow \mathrm{MSE}_{p,\lambda}$
end for
Select p and λ with minimum $\mathrm{MSE}_{p,\lambda}$

n-fold cross validation is called **Leave-One-Out** (LOO).
Standard choice for K: LOO or 10-fold CV.

Outline

Overview

CART: **C**lassification **A**nd **R**egression **T**rees introduced by [Breiman et al., 1984].

Overview

CART: **C**lassification **A**nd **R**egression **T**rees introduced by [Breiman et al., 1984].

Advantages

- **classification OR regression** (i.e., Y can be a numeric variable or a factor);
- **non parametric** method: no prior assumption needed;
- can deal with a **large number of input variables**, either numeric variables or factors (a variable selection is included in the method);
- provide an **intuitive interpretation**.

Overview

CART: **C**lassification **A**nd **R**egression **T**rees introduced by [Breiman et al., 1984].

Advantages

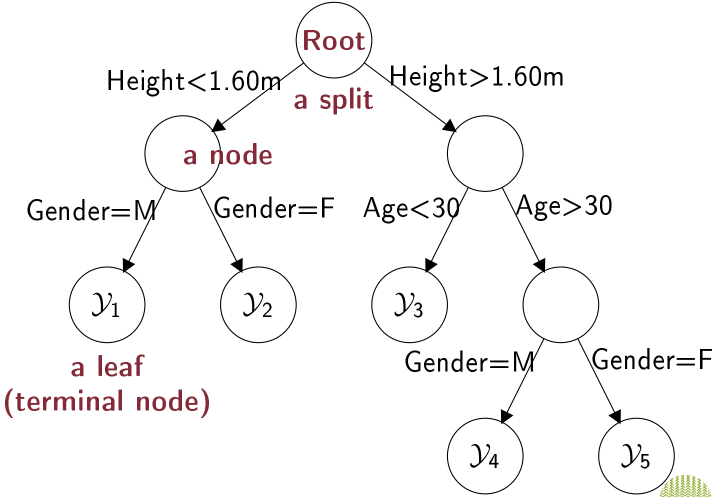
- **classification OR regression** (i.e., Y can be a numeric variable or a factor);
- **non parametric** method: no prior assumption needed;
- can deal with a **large number of input variables**, either numeric variables or factors (a variable selection is included in the method);
- provide an **intuitive interpretation**.

Drawbacks

- require a **large training dataset** to be efficient;
- as a consequence, are often **too simple** to provide accurate predictions.

Example

$X = (\text{Gender}, \text{Age}, \text{Height})$ and $Y = \text{Weight}$



CART learning process

Algorithm

1: Start from root
2: **repeat**
3: move to a "new" node
4: **if** the node is **homogeneous** or **small** enough **then**
5: STOP
6: **else**
7: split the node into two child nodes with **maximal "homogeneity"**
8: **end if**
9: **until** all nodes are processed

Further details

Homogeneity?

- if Y is a numeric variable, **variance** of $(y_i)_i$ for the observations assigned to the node (Gini index is also sometimes used);

Further details

Homogeneity?

- if Y is a numeric variable, **variance** of $(y_i)_i$ for the observations assigned to the node (Gini index is also sometimes used);
- if Y is a factor, **node purity**: % of observations assigned to the node whose Y values are not the node majority class.

Further details

Homogeneity?

- if Y is a numeric variable, **variance** of $(y_i)_i$ for the observations assigned to the node (Gini index is also sometimes used);
- if Y is a factor, **node purity**: % of observations assigned to the node whose Y values are not the node majority class.

Stopping criteria?

- Minimum size node (generally 1 or 5)
- Minimum node purity or variance
- Maximum tree depth

Further details

Homogeneity?

- if Y is a numeric variable, **variance** of $(y_i)_i$ for the observations assigned to the node (Gini index is also sometimes used);
- if Y is a factor, **node purity**: % of observations assigned to the node whose Y values are not the node majority class.

Stopping criteria?

- Minimum size node (generally 1 or 5)
- Minimum node purity or variance
- Maximum tree depth

Hyperparameters can be **tuned by cross-validation** using a grid search.

Further details

Homogeneity?

- if Y is a numeric variable, **variance** of $(y_i)_i$ for the observations assigned to the node (Gini index is also sometimes used);
- if Y is a factor, **node purity**: % of observations assigned to the node whose Y values are not the node majority class.

Stopping criteria?

- Minimum size node (generally 1 or 5)
- Minimum node purity or variance
- Maximum tree depth

Hyperparameters can be **tuned by cross-validation** using a grid search.
An alternative approach is **pruning**...

Choosing an optimal subtree

Algorithm

1: Train the maximal tree, $\mathcal{T}$

2: *Pruning:* Find an "optimal" subtrees sequence $(\mathcal{T}_k)_{k=1,\ldots,K}$

3: **By cross validation, find the errors** $L(\mathcal{T}_k) + \lambda\mathcal{C}(\mathcal{T}_k)$ for $k = 1, \ldots, K$ where L is the error and $\mathcal{C}$ is a complexity measure (number of leafs)

4: Select the subtree s.t. $L(\mathcal{T}_k) + \lambda\mathcal{C}(\mathcal{T}_k)$ is minimum

Making new predictions

A new observation, x_{new}

- is **assigned to a leaf** (straightforward);

Making new predictions

A new observation, x_{new}

- is **assigned to a leaf** (straightforward);
- the corresponding predicted $\hat{y}_{\mathrm{new}}$ is
  - if Y is numeric, the **mean value** of the observations (training set) assigned to the same leaf;

Making new predictions

A new observation, x_{new}

- is **assigned to a leaf** (straightforward);
- the corresponding predicted $\hat{y}_{\text{new}}$ is
  - if Y is numeric, the **mean value** of the observations (training set) assigned to the same leaf;
  - if Y is a factor, the **majority class** of the observations (training set) assigned to the same leaf.

Outline

Advantages/Drawbacks

Random Forest: introduced by [Breiman, 2001].

Advantages/Drawbacks

Random Forest: introduced by [Breiman, 2001].

Advantages

- **classification OR regression** (i.e., Y can be a numeric variable or a factor);
- **non parametric** method (no prior assumption needed) and **accurate**;
- can deal with a **large number of input variables**, either numeric variables or factors;
- can deal with small samples.

Advantages/Drawbacks

Random Forest: introduced by [Breiman, 2001].

Advantages

- **classification OR regression** (i.e., Y can be a numeric variable or a factor);
- **non parametric** method (no prior assumption needed) and **accurate**;
- can deal with a **large number of input variables**, either numeric variables or factors;
- can deal with small samples.

Drawbacks

- **black box** model;
- is **not supported by strong mathematical results** (consistency...) until now.

Basic description

A fact: When the **sample size is small**, you might be unable to estimate properly.

Basic description

A fact: When the **sample size is small**, you might be unable to estimate properly.
This issue is commonly tackled by **bootstrapping** and, more specifically, **bagging** (**B**oostrap **Agg**regat**ing**).

Basic description

A fact: When the **sample size is small**, you might be unable to estimate properly.
This issue is commonly tackled by **bootstrapping** and, more specifically, **bagging** (**B**oostrap **Agg**regat**ing**).

- **Bagging**: combination of simple (and underefficient) regression (or classification) functions;

Basic description

A fact: When the **sample size is small**, you might be unable to estimate properly.
This issue is commonly tackled by **bootstrapping** and, more specifically, **bagging** (**B**oostrap **Agg**regat**ing**).

- **Bagging**: combination of simple (and underefficient) regression (or classification) functions;
- **Random forest** $\simeq$ CART bagging.

Bootstrap

Bootstrap sample: random sampling (with replacement) of the training dataset.

Bootstrap

Bootstrap sample: random sampling (with replacement) of the training dataset.

Standard bootstrap sample size: $2/3n$.

Bootstrap

Bootstrap sample: random sampling (with replacement) of the training dataset.

Standard bootstrap sample size: $2/3n$.

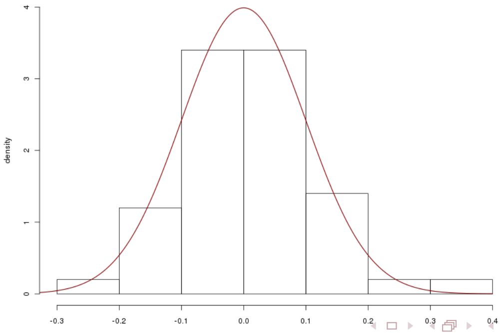
General (and robust) approach to solve several problems:

- Estimating **confidence intervals** (of $\overline{X}$ with no prior assumption on the distribution of X)
  1. Build P bootstrap samples from $(x_i)_i$
  2. Use them to estimate $\overline{X}$ P times
  3. The confidence interval is based on the percentiles of the empirical distribution of $\overline{X}$

Bootstrap

Bootstrap sample: random sampling (with replacement) of the training dataset.
Standard bootstrap sample size: $2/3n$.
General (and robust) approach to solve several problems:

- Estimating **confidence intervals** (of $\overline{X}$ with no prior assumption on the distribution of X)



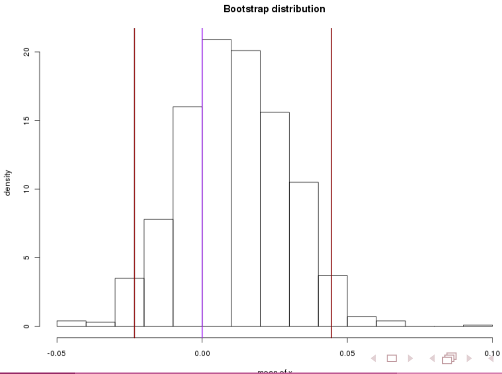
Bootstrap

Bootstrap sample: random sampling (with replacement) of the training dataset.

Standard bootstrap sample size: $2/3n$.

General (and robust) approach to solve several problems:

- Estimating **confidence intervals** (of $\overline{X}$ with no prior assumption on the distribution of X)



Bootstrap

Bootstrap sample: random sampling (with replacement) of the training dataset.

Standard bootstrap sample size: $2/3n$.

General (and robust) approach to solve several problems:

- Estimating **confidence intervals** (of $\overline{X}$ with no prior assumption on the distribution of X)
- Also useful to estimate p-values, residuals, ...

Bagging

Average the estimates of the regression (or the classification) function obtained from B bootstrap samples.

Bagging

Average the estimates of the regression (or the classification) function obtained from B bootstrap samples.

Bagging with regression trees

1: **for** $b = 1, \dots, B$ **do**
2: Construct a bootstrap sample ξ_b
3: Train a regression tree from ξ_b, $\hat{\phi}_b$
4: **end for**
5: Estimate the regression function by

$$\hat{\Phi}^n(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{\phi}_b(x).$$

Bagging

Average the estimates of the regression (or the classification) function obtained from B bootstrap samples.

Bagging with regression trees

1: **for** $b = 1, \ldots, B$ **do**
2: Construct a bootstrap sample ξ_b
3: Train a regression tree from ξ_b, $\hat{\phi}_b$
4: **end for**
5: Estimate the regression function by

$$\hat{\Phi}^n(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{\phi}_b(x).$$

For **classification**, the predicted class is the majority vote class.

Random forests

CART bagging with **additional disturbances**

1. each node is **based on a random (and different) subset of q variables** (an advisable choice for q is $\sqrt{p}$ for classification and $p/3$ for regression).

Random forests

CART bagging with **additional disturbances**

1. each node is **based on a random (and different) subset of q variables** (an advisable choice for q is $\sqrt{p}$ for classification and $p/3$ for regression).
2. the tree is **restricted to the first l nodes** with l small.

Hyperparameters

- those of the CART algorithm;
- those that are specific to the random forest: q, bootstrap sample size, number of trees.

Random forests

CART bagging with **additional disturbances**

1. each node is **based on a random (and different) subset of q variables** (an advisable choice for q is $\sqrt{p}$ for classification and $p/3$ for regression).
2. the tree is **restricted to the first l nodes** with l small.

Hyperparameters

- those of the CART algorithm;
- those that are specific to the random forest: q, bootstrap sample size, number of trees.

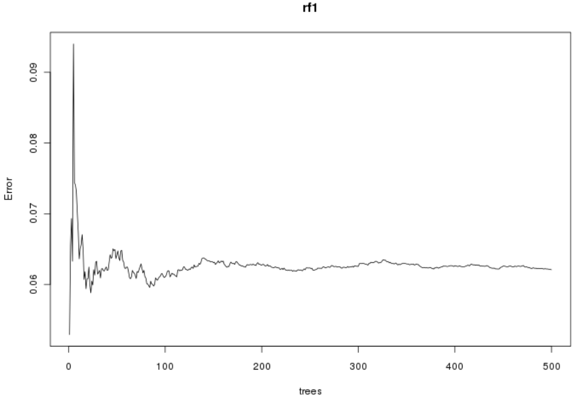
Random forest **are not very sensitive** to hyper-parameters setting: default values for q and bootstrap sample size ($2n/3$) should work in most cases.

Additional tools

- **OOB (Out-Of Bags) error**: error based on the observations not included in the "bag"

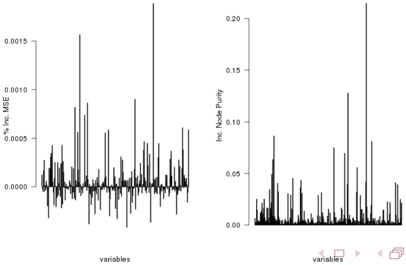
Additional tools

- **OOB (Out-Of Bags) error**: error based on the observations not included in the "bag"
  Stabilization of OOB error is a good indication that there is enough trees in the forest



Additional tools

- **OOB (Out-Of Bags) error**: error based on the observations not included in the "bag"
- **Importance of a variable** to help interpretation: for a given variable X^j
  1. randomize the values of the variable
  2. make predictions from this new dataset
  3. the importance is the mean decrease in accuracy (MSE or misclassification rate)



Bishop, C. (1995).
Neural Networks for Pattern Recognition.
Oxford University Press, New York, USA.

Breiman, L. (2001).
Random forests.
Machine Learning, 45(1):5–32.

Breiman, L., Friedman, J., Olsen, R., and Stone, C. (1984).
Classification and Regression Trees.
Chapman and Hall, New York, USA.

Hornik, K., Stinchcombe, M., and White, H. (1989).
Multilayer feed-forward networks are universal approximators.
Neural Networks, 2:359–366.

Liaubet, L., Lobjois, V., Faraut, T., Tircazes, A., Benne, F., Iannuccelli, N., Pires, J., Glénisson, J., Robic, A., Le Roy, P., SanCristobal, M., and Cherel, P. (2011).
Genetic variability or transcript abundance in pig peri-mortem skeletal muscle: eQTL localized genes involved in stress response, cell death, muscle disorders and metabolism.
BMC Genomics, 12(548).