Apprentissage automatique Avancé

Lesson 1 : linear models again

Sébastien Verel

Laboratoire d'Informatique, Signal et Image de la Côte d'opale (LISIC)
Université du Littoral Côte d'Opale, Calais, France
http://www-lisic.univ-littoral.fr/~verel/

17 September, 2024

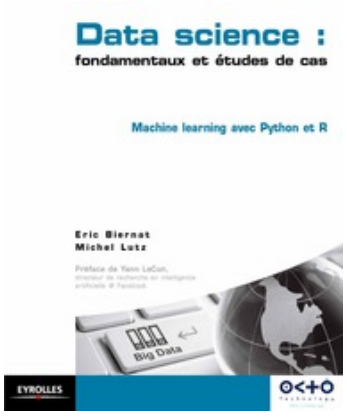
General outline

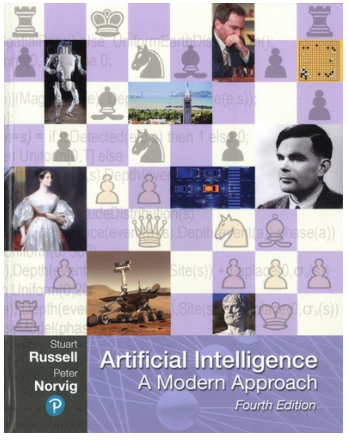
- Partie n°1 [6h] : Bonnes pratiques en IA
- Partie n°2 [3h] : Méthodes Ensemblistes
- Partie n°3 [3h] : Autoencodeurs
- Partie n°4 [3h] : Réseaux convolutionnels
- Partie n°5 [6h] : Réseaux antagonistes génératifs
- Partie n°6 [6h] : Traitement Naturel du Langage
- Partie n°7 [7h] : Apprentissage par renforcement

Outline of the day

- Reminders, context of machine learning
- Linear regression methods
- Overfiiting

Bibliography





• Data Science : fondamentaux et études de cas, Machine Learning avec Python et R , Eric Biernat, Michel Lutz, Eyrolles, 2015.

• Artificial Intelligence : A Modern Approach, Fourth edition, 2020, Stuart Russell and Peter Norvig.

• Vincent Barra, Antoine Cornuéjols, Laurent Miclet, "Apprentissage Artificiel. Concepts et algorithmes. De Bayes et Hume au Deep learning" Eyrolles. Mars 2021. 990 pages.

Example

Problem

Predict the water in the ground

Problem

How to proceed ?

Machine Learning

"Slopy" definition

Study, and design of systems able to learn from data.
(system : computational methods on a computer)

Example

A system able to distinguish spam, and non-spam emails.

Machine Learning

E : set of all possible tasks.
S : a system (a machine)

A more formal definition [T.M. Mitchell, 1997]

$T \subset E$: set of tasks called *training set*
$P : \mathcal{S} \times E \rightarrow \mathbb{R}$: performance metric of a system on tasks.

A system S **learns** from an experience Exp if
the performance of S on tasks T, measured by P, is improving.
$$P(S_{\text{before Exp}}, T) \leq P(S_{\text{after Exp}}, T)$$

Example

Task T : Classifier of emails during one day
Performance P : rejection rate of spams by S
Experience Exp : 1 weak of emails from users

Learning from L. Valliant, 1984 [Turing award, 2010]

PAC ("Probably Approximately Correct")

In model of PAC Learning under the uniform distribution on X, a learning problem is defined with a concept class $\mathcal{C}$, which is just a collection of functions $f : X \to \mathbb{R}$; *"We learn a class $\mathcal{C}$ of functions"*.

A learning algorithm A for $\mathcal{C}$ is a randomized algorithm which has limited access to an unknown target function $f \in \mathcal{C}$.
The two access models are :

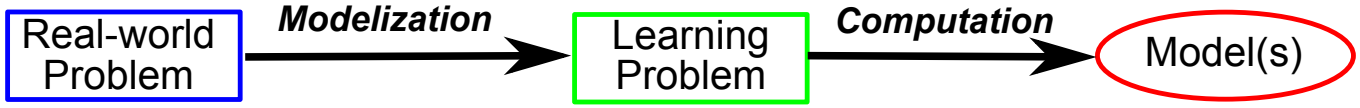
- random : A can draw pairs $(x, f(x))$ where $x \in X$ is uniformly random
- queries : A can request the value $f(x)$ for any $x \in X$ of its choice.

A is given as input an accuracy parameter $\epsilon \in [0, 1/2]$.
Output of A : a hypothesis function $h : X \to \mathbb{R}$.

PAC learning

A learns $\mathcal{C}$ with error ϵ if for any $f \in \mathcal{C}$, with high probability,
A outputs an h which is ϵ-close to f : $\text{dist}(f, h) \leq \epsilon$.

Modelization



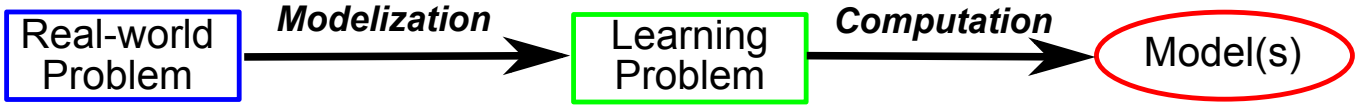
Definition : modelization

Transform a real-world problem
into an abstract learning problem

Modelization

- Abstraction of the reality
- Simplification of the reality :
  number of parameters, noice, defaults, etc.
- Keep relevant elements with respect to problem to learn

Modelization



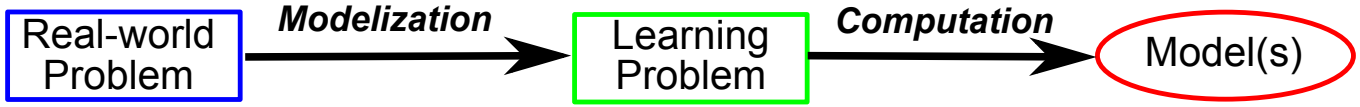
Design of (good) model

Difficult step, but with a good team of :

- Expert of the domain
- Expert in algorithms, abstract representation

it is a very powerful experience

Modelization



Design of (good) model

Difficult step, but with a good team of :

- Expert of the domain
- Expert in algorithms, abstract representation

it is a very powerful experience

Tools for designing a model (representation)

- Binary numbers, integer numbers, floating point numbers
- Combinatoric structure (vector, permutation, list, graph,...)
- Automata, abstract computing machines, etc.

Learn from Data

To learn with a computer,
we need some information, in particular **data**

Definition

Data : "The result of an observation on a population, or a sample"

Statistic, dictionnaire encyclopédique, Springer [Dodge, 2007]

A data is **number**, or a **feature** which gives an **information** on an individual, an object, or an observation.

Example

Sébastien : "I am 10 year old."

Variable

Link between one variable et data :

The features fluctuate according to the individual/object.

Notations :

- Variable X_j
- For the individual/object/observation i : X_{ij}.

Variable X_{age} for the individuals $1, 2, \ldots$: $X_{1age}, X_{2age}, \ldots$

Data type

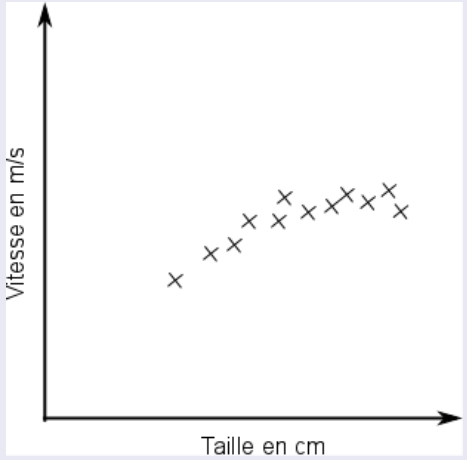
- **Quantitative** data
  mesurable quantity, answer to "how much ?"
  allow computation (mean, etc.),
  comparaisons (equality, difference, inferior/superior)
  - Numerical : $\in \mathbb{R}$
  - Discrete : number of values are limited
- **Qualitative** data
  quality or features
  answer to the "category"
  - Nominale (categorial), ex : eyes color
    comparison (equality / difference)
  - Ordinal
    Order between elements (degree to test, etc.)
    comparison : superior / inférior
- **Structured** data
  relations, etc.
  - Tree, graph, complex data, etc.

Typology according to available information

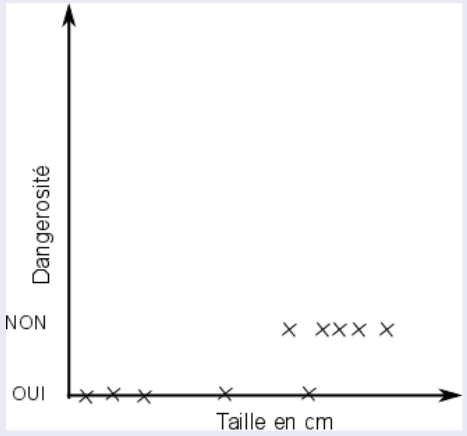
- Supervised learning :
  Learn from a set of examples :
  $\{(x_i, y_i) \mid i \in 1..n\}$
- Non-supervised learning :
  Learn from a set of example without labels (cf. clustering)
  $\{x_i \mid i \in 1..n\}$
- Semi-supervised learning :
  Learn from a set of examples with, and without labels
- Reinforcement learning :
  Learn when the actions on environment
  are rewarded by a score
- ...

Typology according to data

- Regression : (x_i, y_i) with $y_i \in \mathbb{R}$



- Classification : (x_i, y_i) with y_i discrete



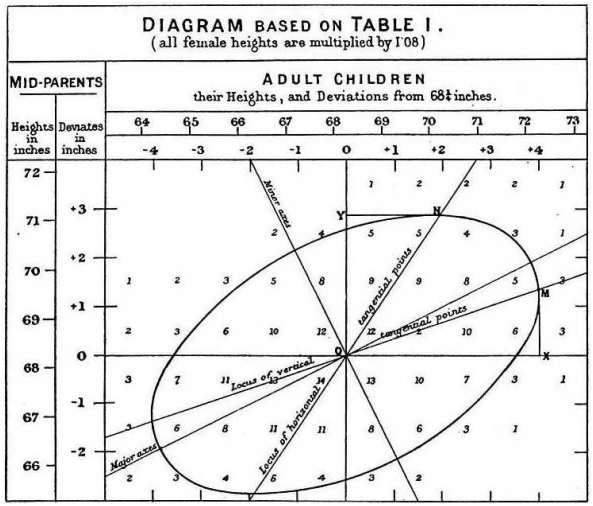
Univariate linear regression

Definition of the model [F. Galton, 1886]

$$\text{input value } x \xrightarrow{\text{model } h} \text{output value } y$$

With univariate linear regression :

$$h_\beta(X) = \beta_0 + \beta_1 X$$



Define : Variance, co-variance, correlation.

Univariate linear regression

Definition of the model [F. Galton, 1886]

$$\text{input value } x \xrightarrow{\text{model } h} \text{output value } y$$

With univariate linear regression :

$$h_\beta(X) = \beta_0 + \beta_1 X$$

Find parameter β such that $h_\beta(X)$ is the closest to Y

Mean square error (MSE) function

$$J_{x,y}(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (h_\beta(x_i) - y_i)^2$$

Multivariate linear regression

Definition of the model

$$\text{input value } x \xrightarrow{\text{model } h} \text{output value } y$$

With multivariate linear regression :

$$h_\beta(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

Mean square error function

$$J_{x,y}(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (h_\beta(x_i) - y_i)^2$$

Define : covariance/correlation matrix.

Computing parameters : time complexity

Ordinary Least Square (OLS)

After algebraic computation,

$$\beta = (X^\intercal X)^{-1} X^\intercal y$$

where X is the matrix of predictor values with n lines of p predictors, y the vector of predicted values

Complexity

$\mathcal{O}(p^2 n)$ for multiplication of $X^\intercal X$
$\mathcal{O}(pn)$ for multiplication of $X^\intercal y$
$\mathcal{O}(p^3)$ to compute the LU factorization of $X^\intercal X$ and compute the product $(X^\intercal X)^{-1} X^\intercal y$

if $n \geq p$ then complexity is $\mathcal{O}(p^2 n)$,
if $n < p$ then complexity is $\mathcal{O}(p^3)$.

Computing parameters : time complexity

Iterative method : gradient descent

Minimize $J_{x,y}(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\beta(x_i) - y_i)^2$ with gradient descent

Gradient of $J_{x,y}(\beta)$ is a close formula (time complexity $\mathcal{O}(np)$)

Gradient step for each variable j :

$$\beta_j := \beta_j - \alpha\frac{\partial}{\partial\beta_j}J(\beta)$$

The time complexity for each "classical" gradient step $\mathcal{O}(pn)$, and we can expect around p steps...

Any gradient variant can be used :
stochastic gradient, nesterov, adam, nadam, etc.
Conjugate gradient ($\mathcal{O}(p^2n)$)

Practice with scikit-learn

Scikit-learn is a library in python with MLmodels, and related tools.
Open source, BSD license, `https://scikit-learn.org/`

From example Ordinary Least Square

```
from sklearn import linear_model

reg = linear_model.LinearRegression()
reg.fit ([[0, 0], [1, 1], [2, 2]], [0, 1, 2])

print(reg.coef_)
```

see code : `linear_reg.ipynb` and url :
`https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-download-auto-examples-linear-model-plot-ols-py`

Practice alone (1)

Use the data set `cars` from `data01/cars.csv`

1. Read the data with pandas :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data01/cars.csv')
```

2. Describe the data with head(), info(), describe()
3. Compute a linear model, and plot it !

Practice alone (2)

Use the data set `basketData09.csv` from `data01`

What is the most important variable to predict the "average points per game" height, or weight ?

Use the data set `advertising.csv` from `data01`

What is the most important variable to predict the "sales" ?
Is there a possible combined effect of variables ?

Interpretation of a linear model

Linear models are simple to interpret :

- **Effect** of a single predictor :
  when the predictor x_j increases by a factor 2,
  then the response is increased by the term $2\beta_j$
- **Importance of predictors** can be compared :
  when the model is $y = 0.001x_1 + 10x_2 + \epsilon$
  then variable x_2 is more important than x_1
  (if the scale of predictors are similar ! ! !)

Sometime, it is more useful :
to understand the relation between variables, and response
than having an accurate model of prediction of response

$\Rightarrow$ Tradeoff between Explanation *vs.* Prediction, toward XAI...

Scaling the data

Don't forget to scale your data :

- Between minimum and maximum :
  $z_i = \frac{x_i - \min[x]}{\max[x] - \min[x]}$, then $z_i \in [0, 1]$
- Using mean, and standard deviation :
  $z_i = \frac{x_i - \bar{x}}{\sigma}$ where $\bar{x}$, and σ are mean, and std dev. of x_i for all i, then, $E[z] = 0$, and $\mathrm{Var}[z] = 1$
- Robust scaling :
  $z_i = \frac{x_i - \mathrm{med}[x]}{q_3[x] - q_1[x]}$ where $\mathrm{med}[x]$ is the median, and $q_1[x]$, $q_3[x]$ first, and third quartile of x_i for all i,
  then, $\mathrm{med}[z] = 0$

See `StandardScaler`, and `Pipeline` in scikit learn

See also the `exo01.py`

Binary classifier

Goal

Find a function f such that $\forall x,\ f(x) \in \{0, 1\}$

Approach

Transform the model h into a binary response :

$$r(x) = \begin{cases} 0 & \text{si } h(x) < 0.5 \\ 1 & \text{si } h(x) \geqslant 0.5 \end{cases}$$

The model h is interpreted as probability function :
$h(x) \in [0, 1]$, and $h(x) = Pr(y = 1|x)$.

Sigmoid fonctions

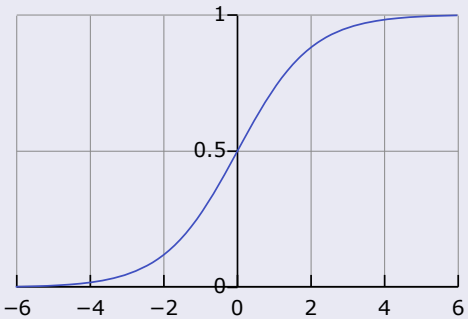
Transform a real number from $\mathbb{R}$ into $[0, 1]$ using ;

- hyperbolic tangent, inverse of normal density, logistic function

Logistic regression

Model : linear model composed with logistic function

$h_\beta(X) = g(\beta_0 + \beta_1 X_1 + \beta_2 X_2 \ \ldots + \beta_n X_n)$ with g logistic function



Loss function : cross-entropy

$j_{x,y}(h) = -\Pr(y = 1) \log \Pr(h(x) = 1) - \Pr(y = 0) \log \Pr(h(x) = 0)$
which gives :
$j_{x,y}(h) = -y \log(h(x)) - (1 - y) \log(1 - h(x))$

$$J_{x,y}(\beta) = \frac{1}{m} \sum_{i=1}^{m} j_{x_i,y_i}(h_\beta)$$

Multiclass case

Goal

Find a function : $f(x) \in \{0, 1, \ldots, k\}$

- For each class c, we build a binary classifier $h^{(c)}$ which measures the probability $y \in c$ (and $y \notin c$)
- The predicted class is the class with the highest probability :

$$y^{\text{pred}} = \text{argmax}_{c \in C} \; h^{(c)}(x)$$