

# Science des données 1

## M2 DS

Thierry Artières\*, Valentin Emiya†, Hachem Kadri†

\* Ecole Centrale de Marseille (ECM)

† Aix-Marseille University (AMU)

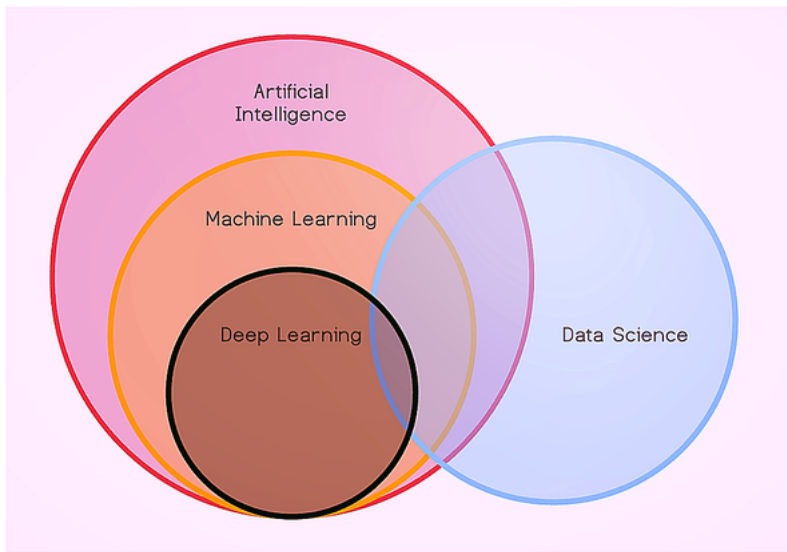
Laboratoire d'Informatique et des Systèmes (LIS), CNRS

QARMA team

<https://qarma.lis-lab.fr/>

6 octobre 2025

- Vous initier à l'apprentissage automatique
- Avoir une idée de :
  - Quand les machines peuvent-elles apprendre ?
  - Pourquoi les machines peuvent-elles apprendre ?
  - Comment les machines peuvent-elles apprendre ?



<https://www.quora.com/What-is-the-difference-between-data-science-machine-learning-and-artificial-intelligence>

"Field of study that gives computers the ability to learn without being explicitly programmed."

– Arthur Samuel, 1959

"Machine Learning is the study of computer algorithms that improve automatically through experience."

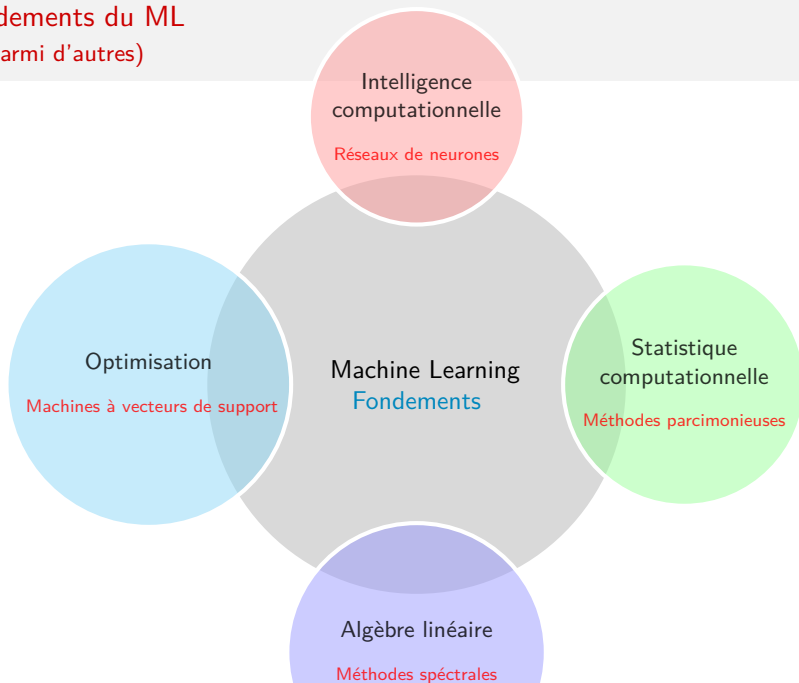
– Tom Mitchell, 1997

"For the last forty years we have programmed computers, for the next forty years we will train them."

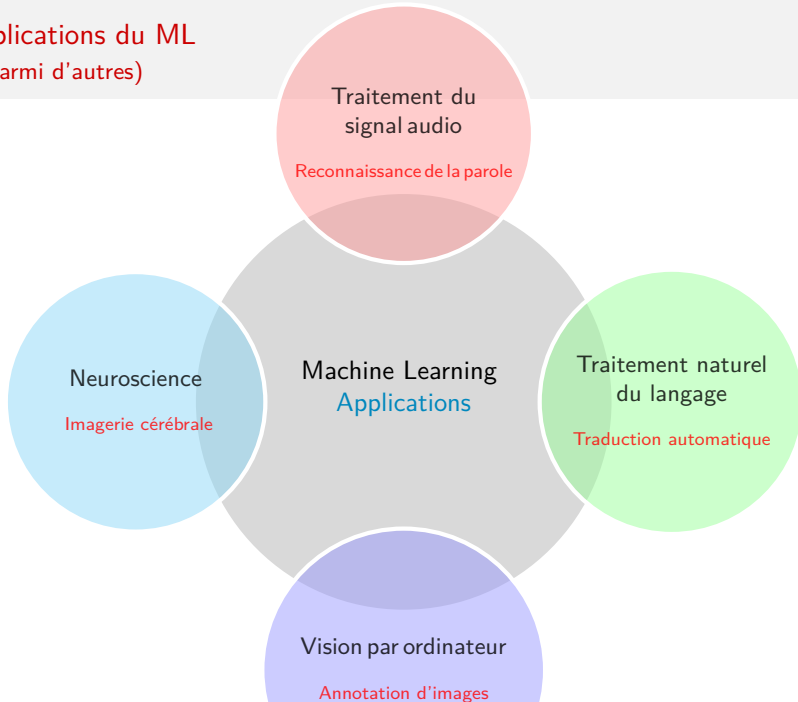
– Christopher Bishop, 2020

# Fondements du ML

(parmi d'autres)



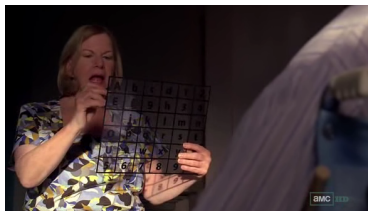
# Applications du ML (parmi d'autres)





# P300 Speller (from L. Ralaivola)

## Vintage P300 Speller



(from *Breaking bad*)

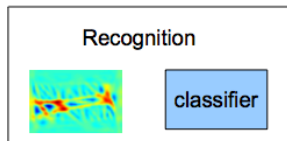
## Modern P300 Speller (from A. Rakotomamonjy)



EEG Signals

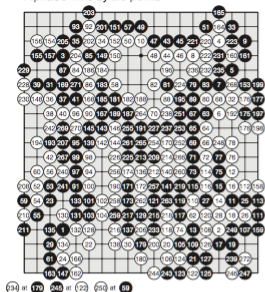


BCI

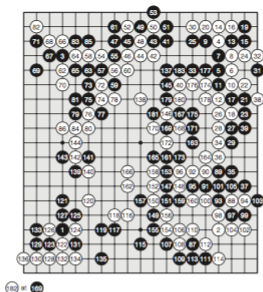


# AlphaGo (Silver et al. 2016)

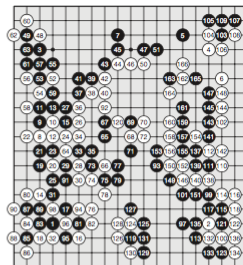
Game 1  
Fan Hui (Black), AlphaGo (White)  
AlphaGo wins by 2.5 points



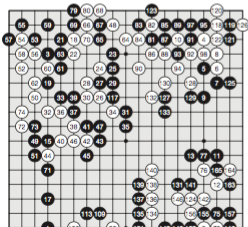
Game 2  
AlphaGo (Black), Fan Hui (White)  
AlphaGo wins by resignation



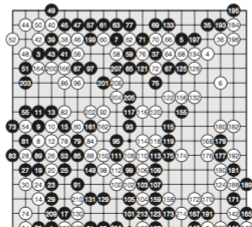
Game 3  
Fan Hui (Black), AlphaGo (White)  
AlphaGo wins by resignation



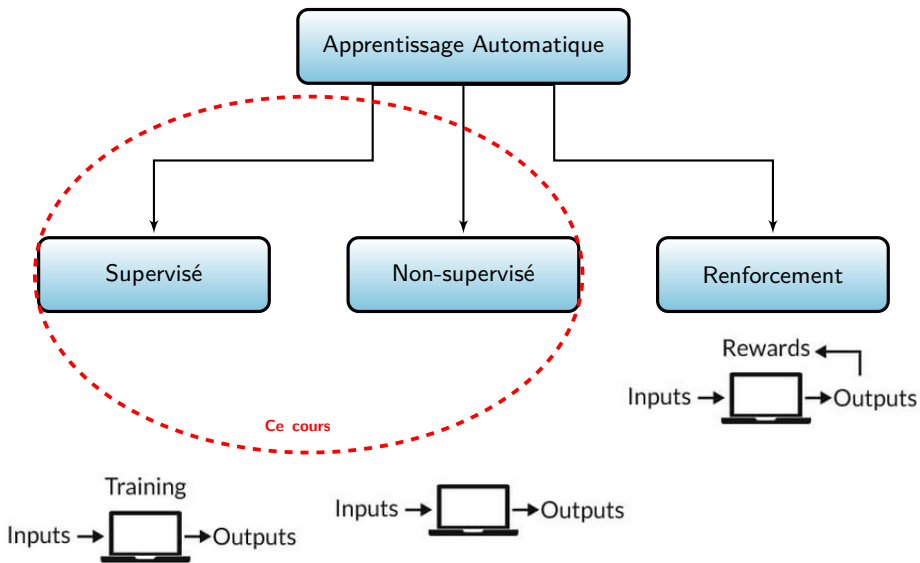
Game 4  
AlphaGo (Black), Fan Hui (White)  
AlphaGo wins by resignation



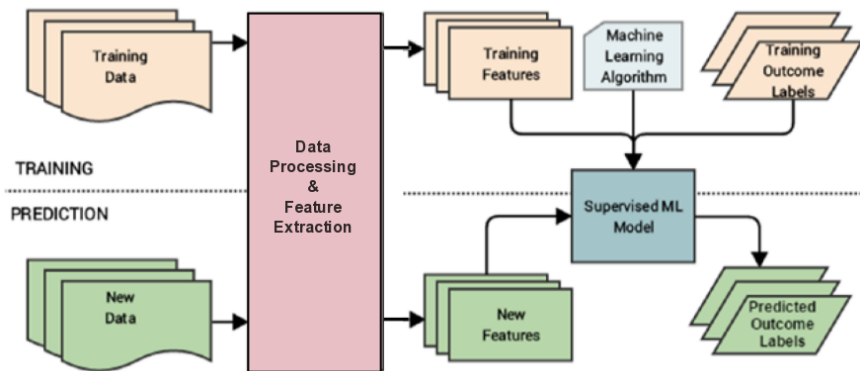
Game 5  
Fan Hui (Black), AlphaGo (White)  
AlphaGo wins by resignation



### 3 cadres d'algorithmes d'apprentissage principaux

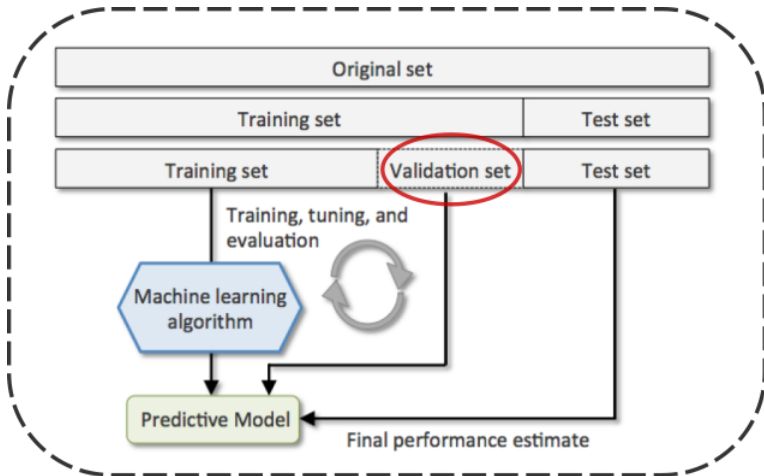


# Apprentissage supervisé : Pipeline



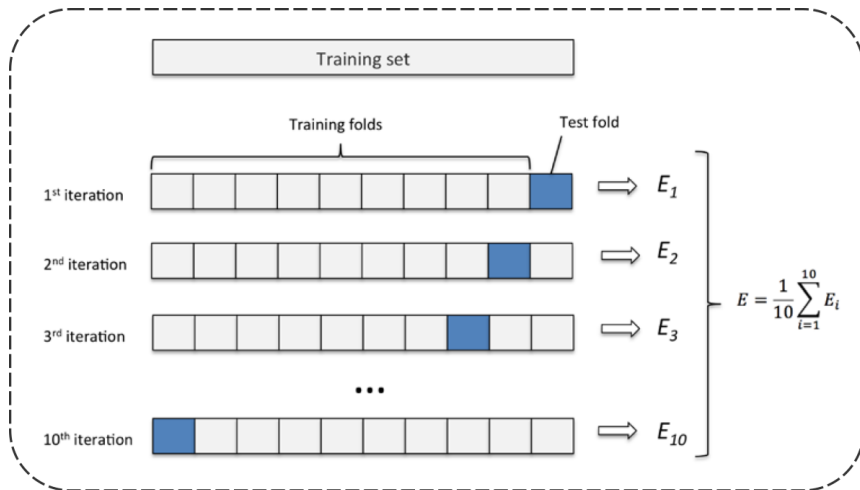
(from Sarkar et al., 2018)

## Ensemble de validation



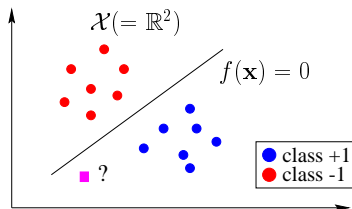
(from Raschka, 2017)

## k-fold Cross-validation (k=10)



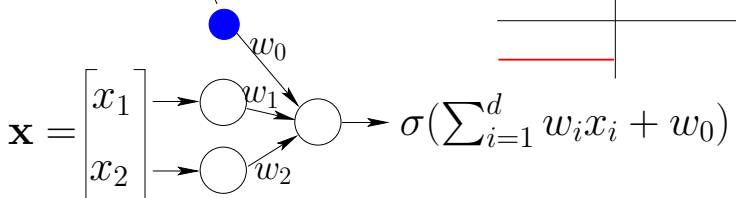
(from Raschka, 2017)

## Classification linéaire



Perceptron,  $\mathcal{X} = \mathbb{R}^d$ ,  $\mathcal{Y} = \{-1, +1\}$

biais : activation = 1



Un classifieur linéaire,  $\mathcal{X} = \mathbb{R}^d$ ,  $\mathcal{Y} = \{-1, +1\}$

- Poids du classifieur :  $w \in \mathbb{R}^d$
- Prédiction du classifieur :  $f(x) = \text{sign}\langle w, x \rangle \stackrel{\text{def}}{=} \sum_{j=1}^d w^{(j)} x^{(j)}$
- Question : comment **apprendre**  $w$  à partir des données ?

## Algorithm : Perceptron

**Input** :  $S = \{(x_i, y_i)\}_{i=1}^n$

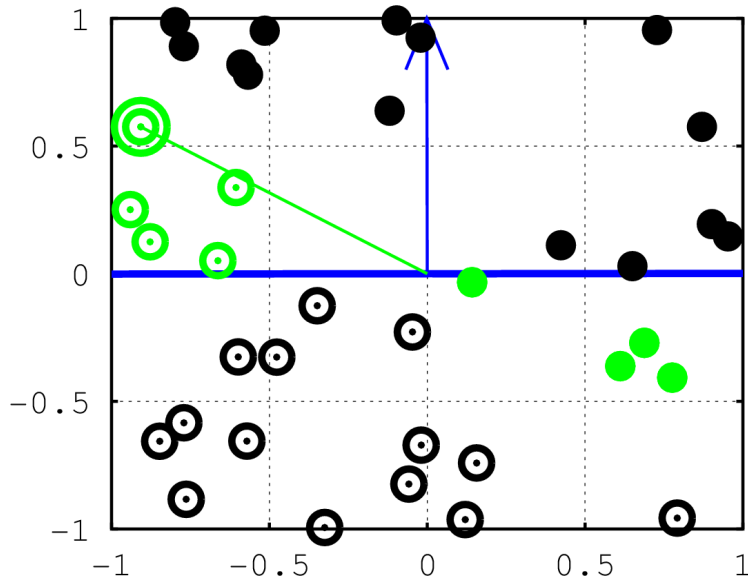
$w \leftarrow 0$

**while** it exists  $(x_i, y_i) : y_i \langle w, x_i \rangle \leq 0$  **do**

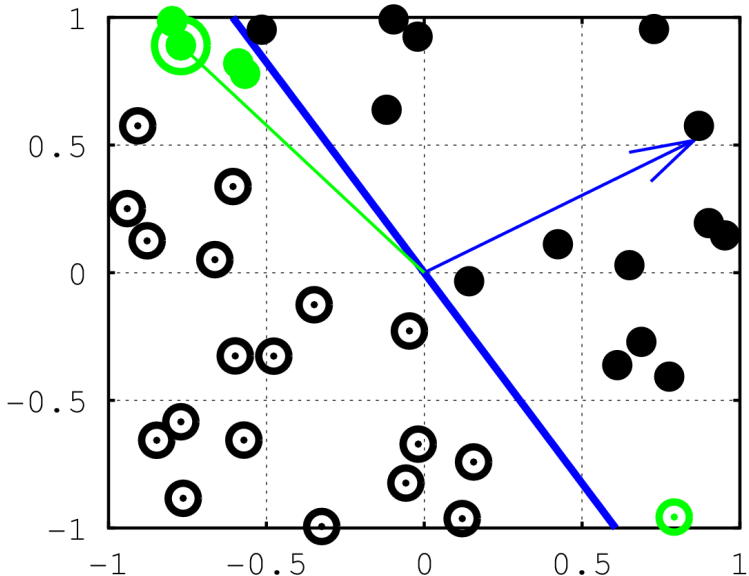
$w \leftarrow w + y_i x_i$

**end while**

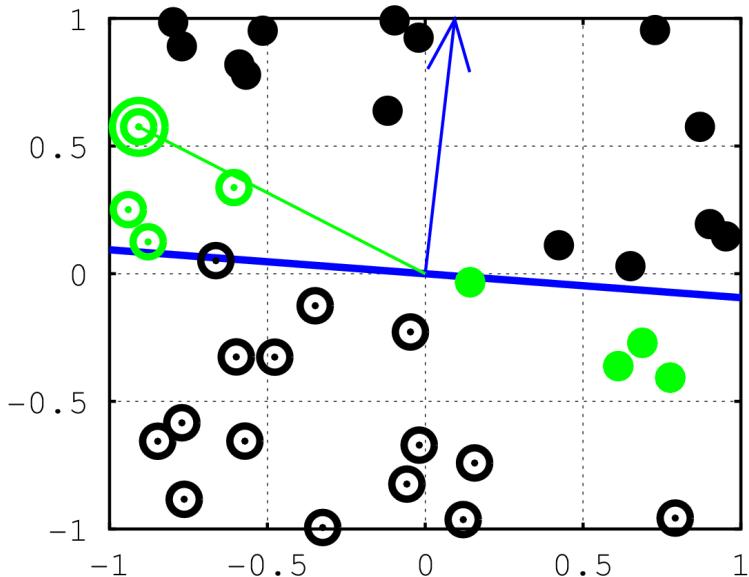
## Perceptron en action (phase d'apprentissage)



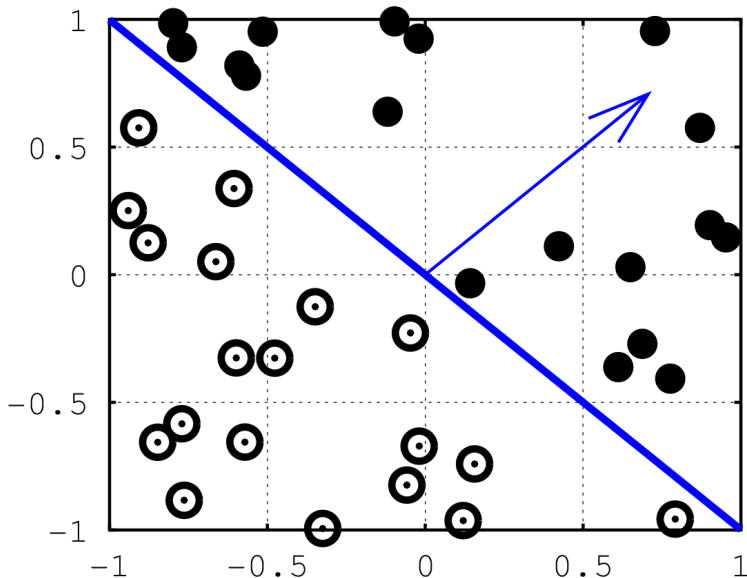
## Perceptron en action (phase d'apprentissage)



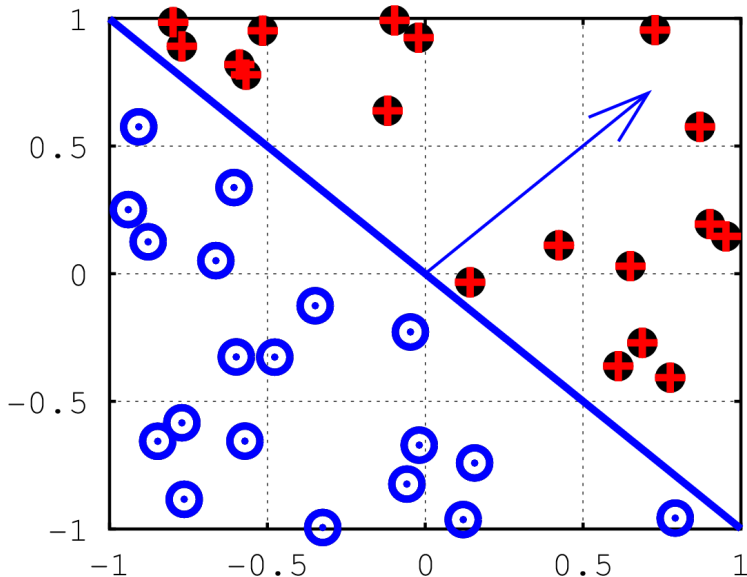
## Perceptron en action (phase d'apprentissage)



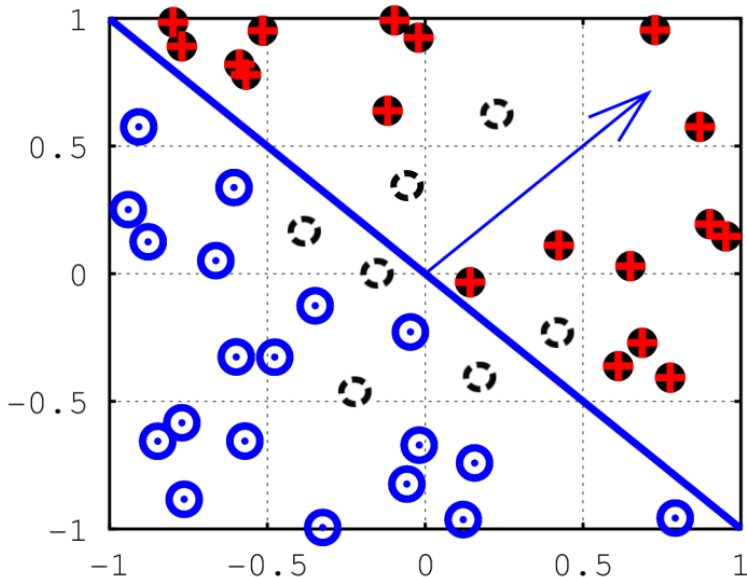
## Perceptron en action (phase d'apprentissage)



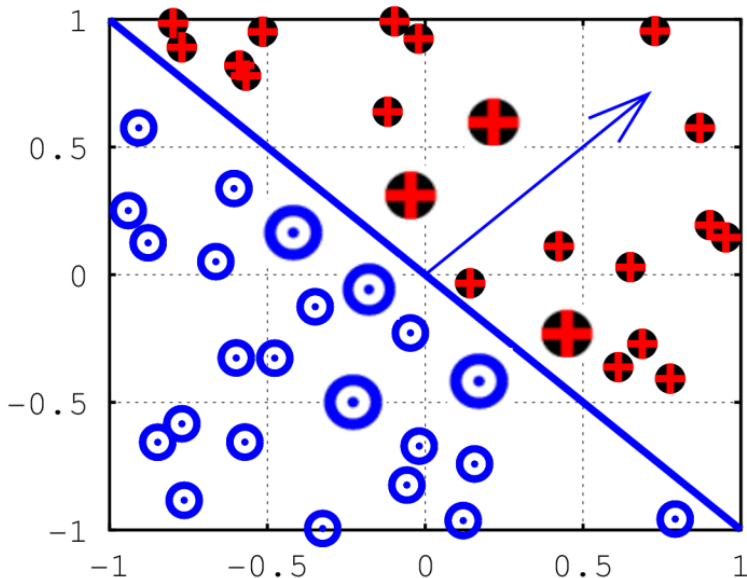
## Perceptron en action (phase d'apprentissage)



## Perceptron in action (phase de test)



## Perceptron in action (phase de test)



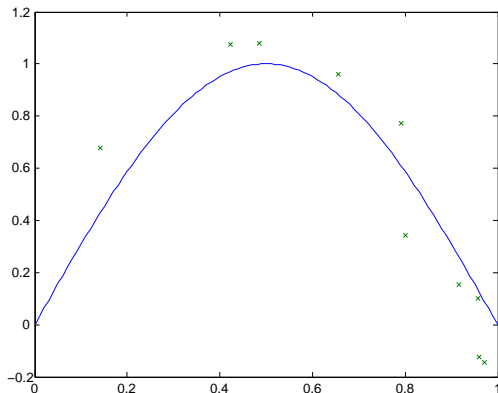
## Propriétés de l'algorithme

- Convergence garantie si les données sont linéairement séparables
- Convergence garantie en mops de  $\frac{R^2}{\gamma^2}$  itérations où
  - $R$  est tel que  $\forall i, \|x^i\| \leq R$
  - $\exists w^*, \gamma$ , tels que  $\|w^*\| = 1$  et  $\forall i, y^i \times w^{*T} \cdot x^i > \gamma$

# Régression linéaire

# Problème de régression

- On observe des données  $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$
- On cherche à modéliser la relation entre  $x$  (variable explicative) et  $y$  (réponse) par une fonction.



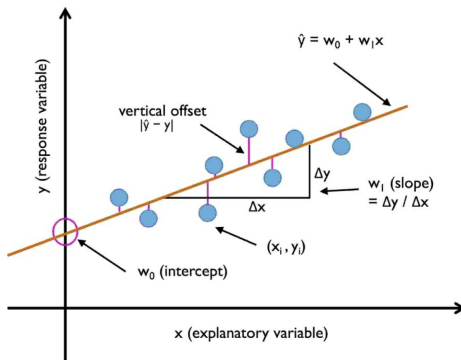
# Régression linéaire

- Modèle linéaire :  $y = \langle \alpha, x \rangle + \beta + \epsilon$

→  $x$  prend ses valeurs dans  $\mathbb{R}^d$

→  $\alpha \in \mathbb{R}^d$  et  $\beta \in \mathbb{R}$  : paramètres du modèle

→  $\epsilon$  est une variable aléatoire telle que  $E(\epsilon) = 0$  et  $V(\epsilon) = \sigma^2$



On suppose que

- $x$  prend des valeurs dans  $\mathbb{R}$ ,
- $y = \alpha x + \beta + \epsilon$  où  $E(\epsilon) = 0$  et  $V(\epsilon) = \sigma^2$  (variance indépendante de  $X$ ).

La fonction de régression est

$$f(x) = \alpha x + \beta.$$

- Soit  $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$  un échantillon. On cherche les valeurs de  $\hat{\alpha}$  et  $\hat{\beta}$  qui minimisent le critère des moindres carrés :

$$(\hat{\alpha}, \hat{\beta}) = \arg \min_{\alpha, \beta} \sum_{i=1}^n (y_i - (\alpha x_i + \beta))^2$$

- Solution :

$$\hat{\alpha} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \text{ et } \hat{\beta} = \bar{y} - \hat{\alpha} \bar{x}$$

où

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \text{ et } \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

La fonction de régression estimée est alors

$$f(x) = \hat{\alpha}x + \hat{\beta}.$$

Les erreurs estimées sont

$$\hat{\epsilon}_i = y_i - \hat{y}_i = y_i - (\hat{\alpha}x_i + \hat{\beta}).$$

La variance estimée est

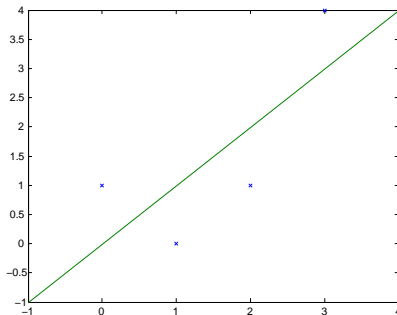
$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^n \hat{\epsilon}_i^2.$$

## Estimateurs des moindres carrés : Exemple

Soit  $S = \{(0, 1), (1, 0), (2, 1), (3, 4)\}$  un échantillon.

On trouve

$$\bar{x} = 3/2, \bar{y} = 3/2, \hat{\alpha} = 1 \text{ et } \hat{\beta} = 0.$$



On a  $\hat{\epsilon}_1 = 1, \hat{\epsilon}_2 = -1, \hat{\epsilon}_3 = -1, \hat{\epsilon}_4 = 1$  et  $\hat{\sigma}^2 = 2$ .

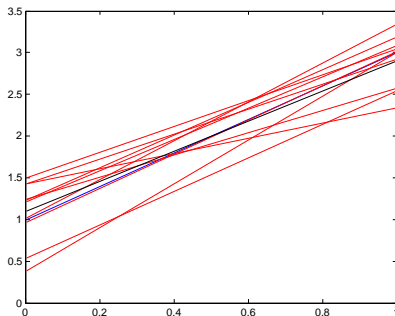
- $\hat{\alpha}$ ,  $\hat{\beta}$  et  $\hat{\sigma}^2$  sont des **estimateurs non biaisés** de  $\alpha$ ,  $\beta$  et  $\sigma^2$  : si on répète un grand nombre d'expériences avec le même modèle, les moyennes des estimations convergent vers les paramètres du modèle.
- $\hat{\alpha}$ ,  $\hat{\beta}$  et  $\hat{\sigma}^2$  sont des **estimateurs consistants** de  $\alpha$ ,  $\beta$  et  $\sigma^2$  : plus on dispose d'observations, plus les estimations se rapprochent des paramètres du modèle.
- si  $\epsilon$  suit une loi normale, l'estimateur des moindres carrés est aussi l'**estimateur du maximum de vraisemblance** : celui qui maximise la probabilité des observations.

## Estimateur non biaisé : Illustration

$x$  prend 11 valeurs équidistantes dans  $[0, 1]$  ;  $y = 2 * x + 1 + \text{Norm}(0, 1)$ .

On réalise 10 expériences.

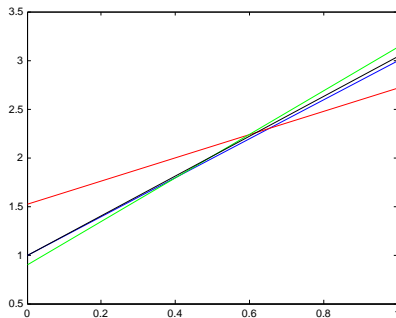
- en bleu : la droite de régression
- en rouge : chaque estimation
- en noir : la moyenne des estimations.



## Estimateur consistant : Illustration

$x$  prend  $n$  valeurs équidistantes dans  $[0, 1]$  ;  $y = 2 * x + 1 + \text{Norm}(0, 1)$ .

- en bleu : la droite de régression
- en rouge :  $n = 11$
- en vert :  $n = 101$
- en noir :  $n = 1001$ .



Soit  $\{(x_1, y_1), \dots, (x_n, y_n)\} \subset$  l'échantillon d'apprentissage.

Soit  $X$  la matrice  $n \times d$  dont la  $i$ -ème ligne est  $x_i$ .

Soit  $Y$  le vecteur colonne de taille  $n$  composé des étiquettes  $y_i$ .

$n$  est le nombre d'exemples et  $d$  est le nombre d'attributs.

- avec des notations matricielles :

$$\sum_{i=1}^n (y_i - \alpha^\top x_i)^2 = \|Y - X\alpha\|^2$$

- Gradient par rapport à  $\alpha$  :

$$\nabla_{\alpha} (\|Y - X\alpha\|^2) = -2X^\top (Y - X\alpha)$$

La solution du problème de régression par moindres carrés est obtenue en calculant le vecteur de paramètre  $\alpha$  qui annule la dérivée.

L'estimateur des moindres carrés est

$$\hat{\alpha} = (X^T X)^{-1} X^T Y$$

où  $X^T$  désigne la matrice transposée de  $X$ .

Si  $X^T X$  n'est pas inversible, ou si  $\det(X^T X) \simeq 0$ , ... il est nécessaire de transformer le problème.

Pour considérer le terme biais, c-a-d le modèle  $y = \alpha^\top x + \beta + \epsilon$ , on augmente la dimension de l'espace d'entrée et on ajoute une composante égale à 1 ( $\tilde{x} = (x, 1)$ ).

Soit  $\tilde{X}$  la matrice  $n \times (d + 1)$  dont la  $i$ -ème ligne est  $(x_i, 1)$ .

$$\begin{pmatrix} \hat{\alpha} \\ \hat{\beta} \end{pmatrix} = (\tilde{X}^\top \tilde{X})^{-1} \tilde{X}^\top Y$$

**Exemple :**

$$S = \{((0, 0), -1), ((0, 1), 1), ((1, 0), 1), ((1, 1), 1)\}.$$

On a

$$X = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, X^T = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \text{ et } Y = \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

On vérifie que

$$X^T X = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 4 \end{pmatrix}, (X^T X)^{-1} = \begin{pmatrix} 1 & 0 & -1/2 \\ 0 & 1 & -1/2 \\ -1/2 & -1/2 & 3/4 \end{pmatrix}$$

$$(X^T X)^{-1} X^T = \begin{pmatrix} -1/2 & -1/2 & 1/2 & 1/2 \\ -1/2 & 1/2 & -1/2 & 1/2 \\ 3/4 & 1/4 & 1/4 & -1/4 \end{pmatrix} \text{ et } (X^T X)^{-1} X^T Y = \begin{pmatrix} 1 \\ 1 \\ -1/2 \end{pmatrix}$$

soit

$$\hat{\alpha} = (1, 1) \text{ et } \hat{\beta} = -1/2.$$

- Peu vraisemblable, en général, que les données d'observations  $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$  puissent être précisément décrites par un modèle linéaire.
- Sous des conditions de régularités très générales, toute fonction  $f$  peut être approchée, au moins localement, par une combinaison **linéaire**
  - de **monômes** (développements limités)

$$\sin x = x - x^3/3! + x^5/5! - \dots$$

- ou de **fonctions trigonométriques** (développements de Fourier)

$$x = 2 \left( \sin x - \frac{\sin 2x}{2} + \frac{\sin 3x}{3} - \dots \right)$$

- Soient  $h_1, \dots, h_M : \mathbb{R}^d \mapsto \mathbb{R}$ .
- On transforme chaque  $x_i$  en un vecteur  $(h_1(x_i), \dots, h_M(x_i))$ .
- On considère le problème de régression linéaire multivarié suivant : trouver les coefficients  $\alpha_1, \dots, \alpha_M, \beta \in \mathbb{R}$  qui minimisent

$$\sum_{i=1}^n \left( y_i - \left( \sum_{j=1}^M \alpha_j h_j(x_i) + \beta \right) \right)^2.$$

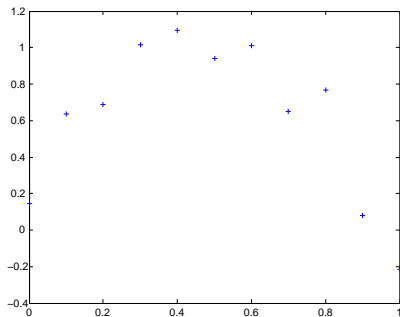
- On obtient ainsi une fonction de régression non linéaire

$$f(x) = \sum_{j=1}^M \hat{\alpha}_j h_j(x) + \hat{\beta}.$$

## Un exemple

On observe les données suivantes :

0	0.1000	0.2000	0.3000	0.4000	0.5000	0.6000	0.7000	0.8000	0.9000	1.0000
0.1434	0.6351	0.6856	1.0160	1.0964	0.9393	1.0098	0.6516	0.7655	0.0796	-0.2138



Les données ne semblent pas alignées : on les transforme par les fonctions

$$h_1(x) = x, h_2(x) = x^2 \text{ et } h_3(x) = x^3.$$

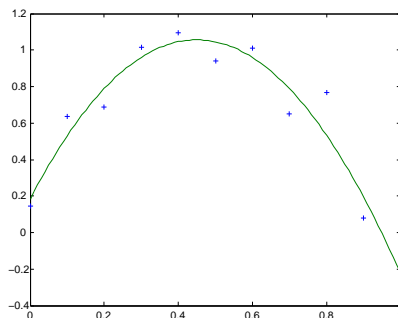
## Un exemple (suite)

Une régression linéaire multivariée permet de trouver

$$\beta = 0.1848, \alpha_1 = 3.8960, \alpha_2 = -4.3942 \text{ et } \alpha_3 = 0.0878$$

soit le polynôme

$$p(x) = 0.1848 + 3.8960x - 4.3942x^2 + 0.0878x^3.$$



Les observations  $x$  peuvent dépendre d'un très grand nombre de variables.

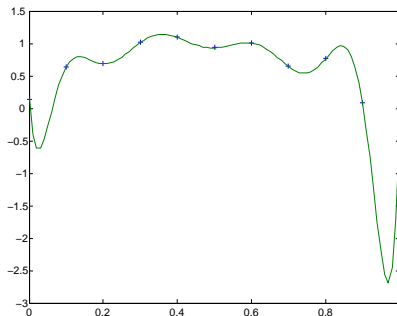
- Constat : trop de variables nuisent à la qualité de la prédiction.
- Le modèle induit peut être difficile à interpréter s'il fait intervenir toutes les variables au même niveau.
- Question : comment trouver un bon compromis entre biais et variance ? **underfitting** et **overfitting** ?

En trouvant un compromis entre l'adéquation aux données et la complexité du modèle.

## Un exemple (suite)

Si on prend comme base les monômes  $h_i(x) = x^i$  pour  $1 \leq i \leq 10$ , on trouve le polynôme

$$p(x) \simeq 0.1463 - 76.0471x + 2394.5864x^2 - 28139.8810x^3 + 173816.7283x^4 - 634281.2333x^5 + \\ 1437424.4162x^6 - 2044405.0970x^7 + 1774215.9551x^8 - 858102.0064x^9 + 177152.2278x^{10}.$$



Quels sont les monômes les plus significatifs ?

Pénaliser la taille du modèle. On cherche à minimiser

$$\sum_{i=1}^n (y_i - (\alpha x_i + \beta))^2 + C(\beta^2 + \|\alpha\|_2^2),$$

avec  $\|\alpha\|_2 = \sqrt{\sum_j \alpha_j^2}$ .

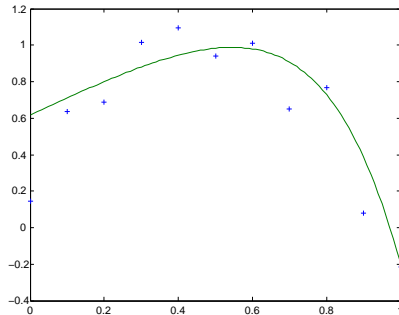
- se résoud comme la régression multivariée sans pénalisation.
- mais comment trouver la valeur de  $C$  ? Par exemple, par validation croisée.

## Un exemple (suite)

Avec les monômes  $h_i(x) = x^i$  pour  $1 \leq i \leq 10$  et  $C = 0.1$ , on trouve les coefficients

0.6189 ; 0.9368 ; -0.0655 ; -0.3620 ; -0.4026

-0.3529 ; -0.2745 ; -0.1906 ; -0.1095 ; -0.0341 ; 0.0350.



On cherche à minimiser

$$\sum_{i=1}^n (y_i - (\alpha x_i + \beta))^2 + C(|\beta| + \|\alpha\|_1),$$

avec  $\|\alpha\|_1 = \sum_j |\alpha_j|$ .

- Un grand nombre de coefficients s'annule.
- Il faut toujours déterminer une bonne valeur pour  $C$ .

En pratique ...

- NumPy : tableaux multidimensionnels
- SciPy : calcul scientifique
- Matplotlib : visualisation
- pandas : analyse de données
- scikit-learn : apprentissage automatique

## Ecosystème Python

# Installation de Python and des packages

## Anaconda Python distribution

est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique.

## Anaconda installer :

<https://www.anaconda.com/download/>

## Anaconda quick-start guide :

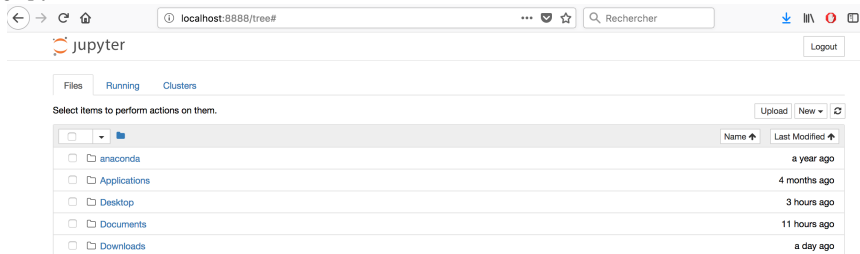
<https://conda.io/docs/user-guide/getting-started.html>

# Jupyter Notebook

- évolution du projet IPython.
- environnement computationnel interactif
- peut contenir du code, du texte, des images, ...
- installé par défaut avec la distribution Anaconda

Pour lancer Jupyter notebook, exécuter dans le terminal la commande :

```
[fontsize=\scriptsize]  
$ jupyter notebook
```



- Outil interne google pour la data science et le ML
- Découvrir Colab : <https://colab.research.google.com/notebooks/welcome.ipynb>

```
1 | print('Hello, Colaboratory!')
2 | Hello, Colaboratory!

Colaboratory allows you to execute TensorFlow code in your browser with a single click. The example below adds two matrices.


$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \end{bmatrix}$$


3 | import tensorflow as tf
   import numpy as np
   with tf.Session() as sess:
       input1 = tf.constant([1,0, 3], dtype=tf.int32)
       input2 = tf.constant(np.reshape(np.arange(1,5, 7,0, dtype=np.float32), (2, 3)))
       output = tf.add(input1, input2)
       result = output.eval()
       print(result)
4 | [[ 2.  3.  4.]
   [ 5.  6.  7.]]

Colaboratory includes widely used libraries like pandas, matplotlib, simplifying visualization.

5 | import matplotlib.pyplot as plt
   import numpy as np
   # = np.arange(20)
   y = np.linspace(0, 1, np.random.randn(1), 1)
   # x = np.linspace(0, 1, np.random.randn(1), 1)
   plt.plot(x, y, 'o', np.arange(20)+1, '-');
6 |
```

## Notations ...

$n$	nombre d'exemples
$d$	nombre d'attributs
$x$	donnée d'entrée dans $\mathbb{R}^d$
$x^{(j)}$	$j$ -ème composante du vecteur $x$
$y$	étiquette
$X$	matrice des données d'entrée dans $\mathbb{R}^{n \times d}$
$Y$	vecteur des étiquettes dans $\mathbb{R}^n$
$\langle u, v \rangle$	produit scalaire entre deux vecteurs $u$ et $v$
$\ u\ , \ u\ _2$	norme euclidienne d'un vecteur $u$
$\ u\ _1$	norme 1 d'un vecteur $u$
$M^\top, u^\top$	transposée d'une matrice $M$ ou d'un vecteur $u$
$M^{-1}$	inverse d'une matrice $M$
$\nabla f(\alpha)$	le gradient d'une fonction $f : \mathbb{R}^d \rightarrow \mathbb{R}$ au point $\alpha \in \mathbb{R}^d$