Intelligence artificielle

Frédéric Camps
fcamps@laas.fr

Machine learning, deep learning

Agenda

Approche du machine learning

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Equation normale
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone
- Machine à vecteurs de support linéaire et non linéaire (SVM)

Apprentissage non supervisé:

- K-NN, K-MEANS
- Arbre de décision

Analyse des séries chronologiques:

- Modèle ARIMA

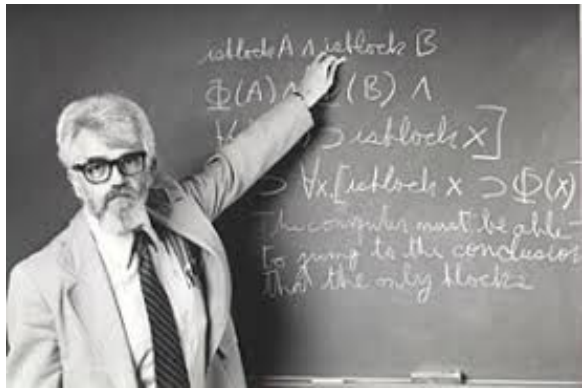
Outils

TP avec données des IoT

http://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257
https://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)

Qu'est ce que l'intelligence artificielle ?

Intelligence Artificielle (IA) : Concevoir des systèmes capables de reproduire le comportement de l'humain dans ses activités de raisonnement et comportement.



Selon John MacCarthy, l'un des créateurs de ce concept, « toute activité intellectuelle peut être décrite avec suffisamment de précision pour être simulée par une machine ».

http://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257
https://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)
Intelligence Artificielle - Stuart Russel - Peter Norvig

Les approches scientifiques de l'IA

→Le processus de la **pensée** et du **raisonnement**

→Le **comportement** d'un système

http://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257
https://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)
Intelligence Artificielle - Stuart Russel - Peter Norvig

→ L'IA selon le processus de la **pensée** et du **raisonnement**:

- Des systèmes qui pensent comme les humains (*Haugeland, 1985*)
- Des systèmes qui pensent rationnellement (*Charniak et McDermott, 1985*)

http://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257
https://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)

→ L'intelligence artificielle (IA) selon le **comportement** d'un système:

- Des systèmes qui agissent comme les humains (*Kurzweil, 1990*),
- Des systèmes qui agissent rationnellement (*Pool et al., Nilsson, 1998*)

Intelligence Artificielle

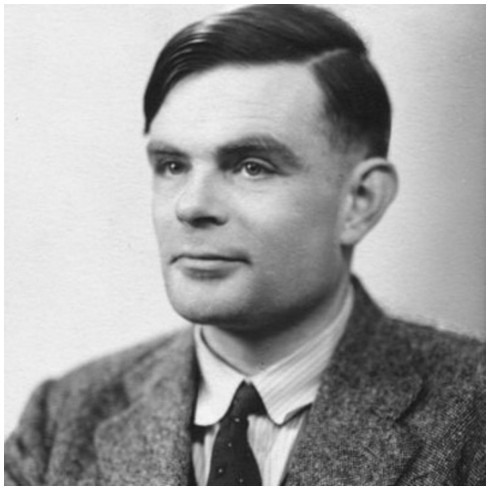
Notion de base / Approche scientifique

http://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257
https://fr.wikipedia.org/wiki/Rationalit%C3%A9
https://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)
Intelligence Artificielle - Stuart Russel - Peter Norvig

Réussite par rapport aux **performances humaines**	Concept idéal de l'intelligence appelé **"rationalité"**
Penser comme les humains	**Penser** rationnellement
Agir comme les humains	**Agir** rationnellement

https://en.wikipedia.org/wiki/Alan_Turing
https://en.wikipedia.org/wiki/Turing_test

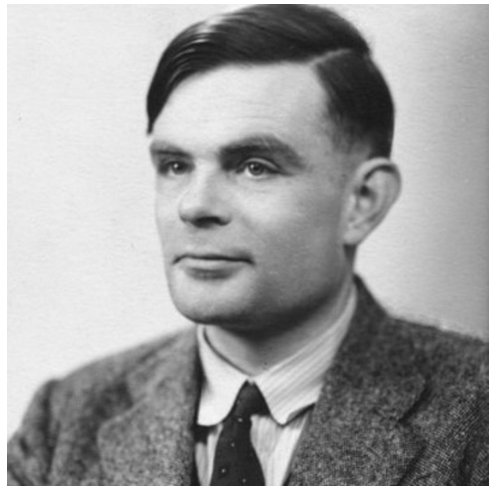
Agir comme les humains, le test de Turing (1950):



"Un ordinateur réussit le test si, après avoir posé un certain nombre de questions écrites, un humain est dans l'incapacité de dire si les réponses proviennent d'une personne ou d'un ordinateur."

https://en.wikipedia.org/wiki/Alan_Turing
https://en.wikipedia.org/wiki/Turing_test

Agir comme les humains, le test de Turing (1950):



Un ordinateur doit posséder les fonctionnalités suivantes:

- le traitement du langage naturel,
- la représentation des connaissances,
- le raisonnement automatisé,
- l'apprentissage,
- vision artificielle,
- capacité robotique.

https://fr.wikipedia.org/wiki/Intelligence_artificielle

Penser comme les humains, l'approche cognitive:

Il faut déterminer comment pensent les humains (comprendre les rouages de l'esprit):

- saisir les pensées,
- les expériences psychologiques,
- observer les comportements d'une personne,
- observer le cerveau ...

→ Définir une théorie de l'esprit puis la traduire informatiquement.

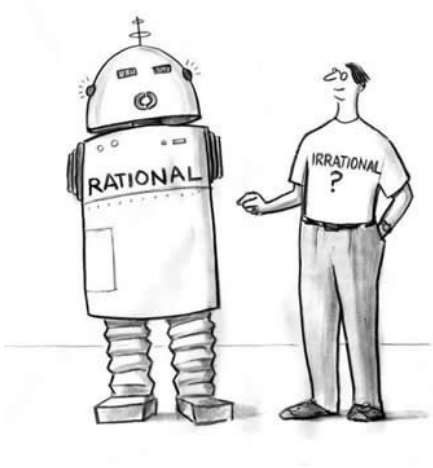
https://fr.wikipedia.org/wiki/Intelligence_artificielle

Penser rationnellement: les “lois de la pensée“

- Aristote: procédés des raisonnements irréfutables
- Suite de syllogismes, exemple: “*Socrate est un homme, tous les hommes sont mortels, donc Socrate est mortel*”
- Notation précise des assertions (Proposition, de forme affirmative ou négative, qu'on avance et qu'on donne comme vraie)

→ Ces études ont défini le domaine de la logique, traduit aujourd’hui en approche **logiciste**.

https://fr.wikipedia.org/wiki/Inf%C3%A9rence



Agir rationnellement: l'approche de l'agent rationnel

- L'agent fonctionne par inférence (ajout d'information),
- La validité des inférences donne accès à la rationalité,
- L'agent fonctionne en autonomie et s'adapte au contexte,
- Un agent rationnel est une entité qui propose la meilleure solution selon un contexte.

https://fr.wikipedia.org/wiki/Intelligence_artificielle
https://fr.wikipedia.org/wiki/Pr%C3%A9dicat_(logique_math%C3%A9matique)

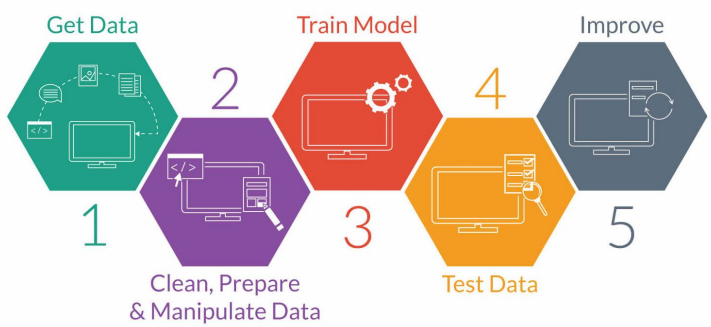
Les différentes approches:

Gestion de l'incertitude	- Approche Bayésienne - Filtrage numérique
Résolution par exploration	- La solution est une séquence d'action (environnement observable, déterministe) - La solution dans un espace (sous contrainte)
Résolution basée sur la connaissance	- Logique propositionnelle: comprendre l'environnement (simple) et déduire les actions à entreprendre - Prédicat
Apprentissage par l'exemple	- Réseau de neurones - Arbre de décision - Modèle linéaire ...

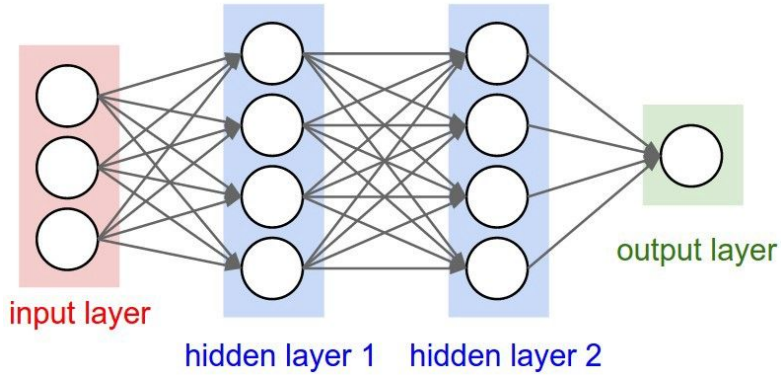
https://fr.wikipedia.org/wiki/Intelligence_artificielle

Les différentes approches:

Machine learning



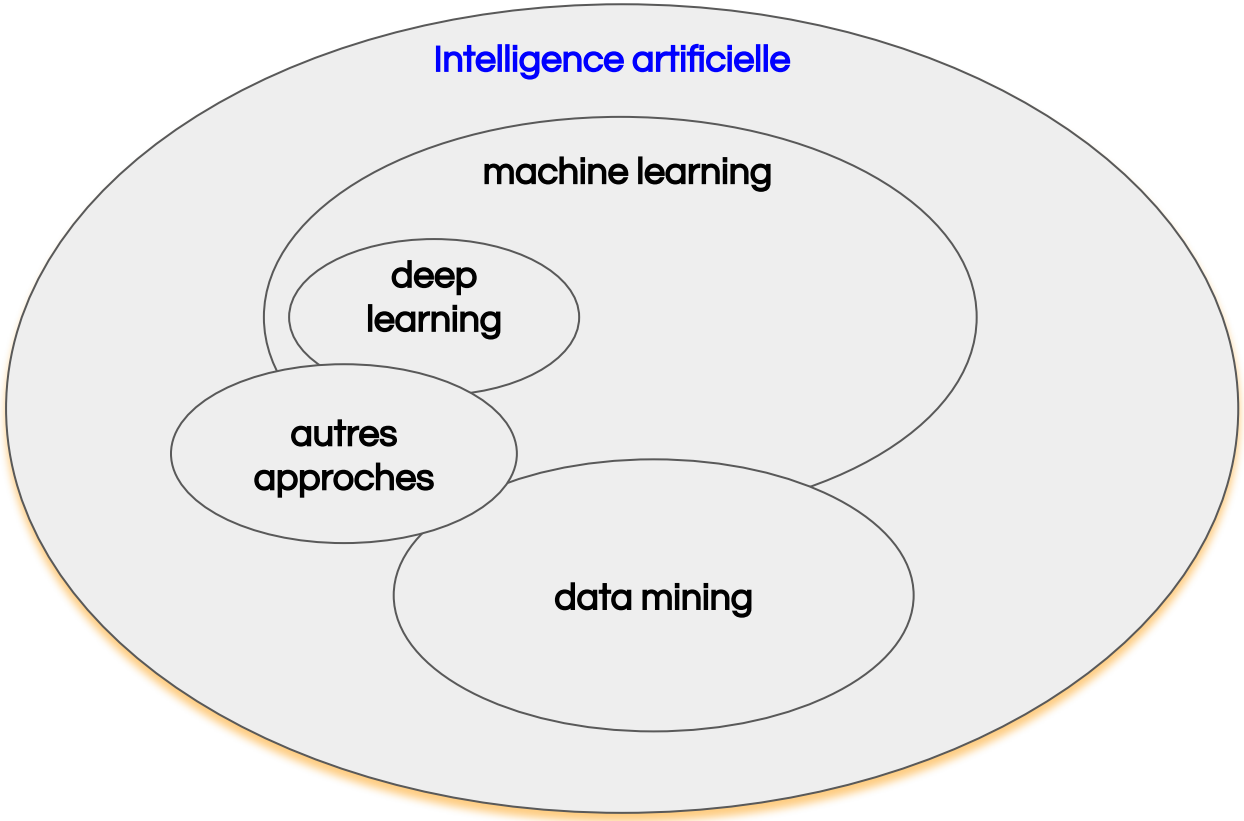
Deep learning



Autres approches …

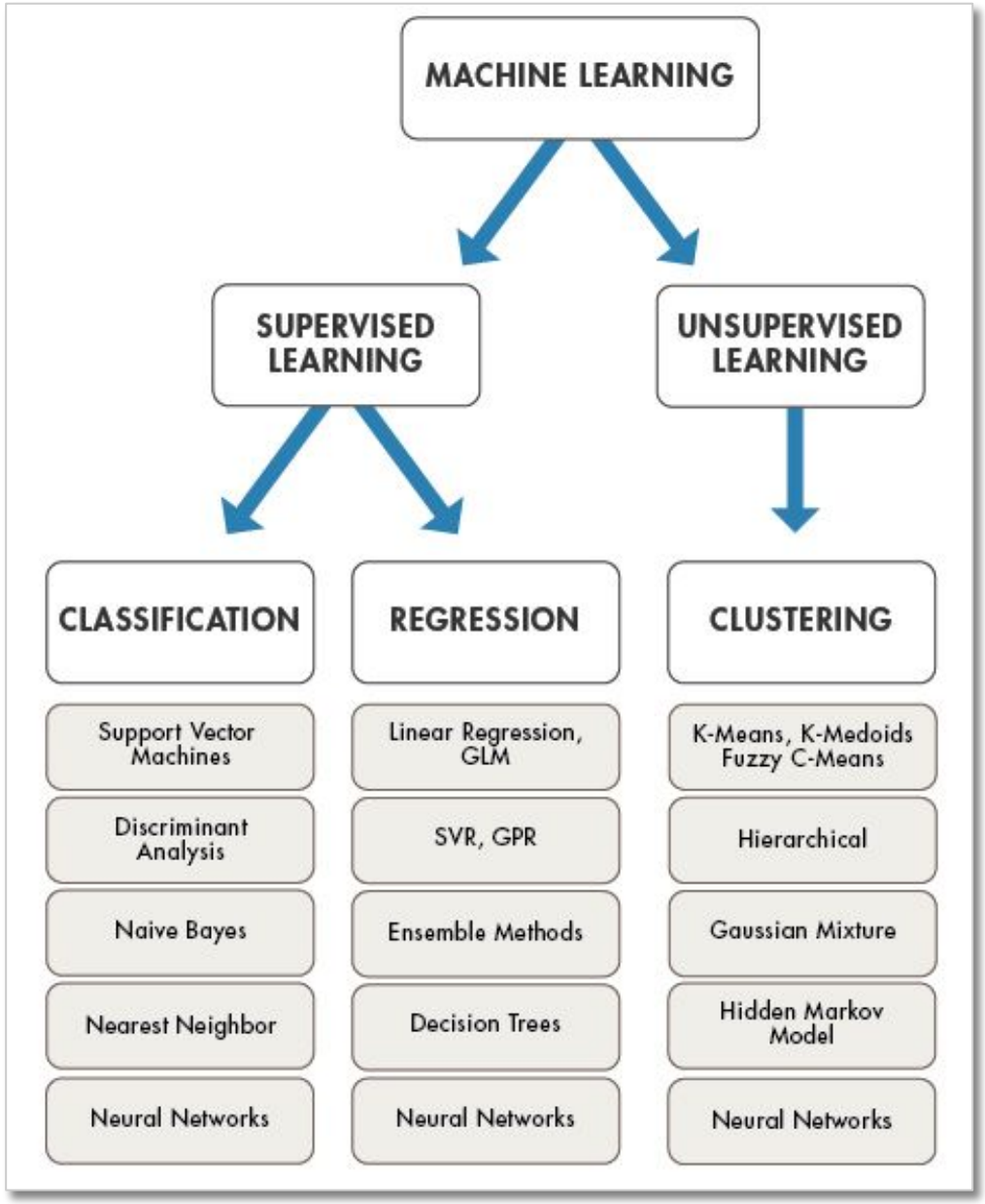
https://fr.wikipedia.org/wiki/Intelligence_artificielle

Structuration



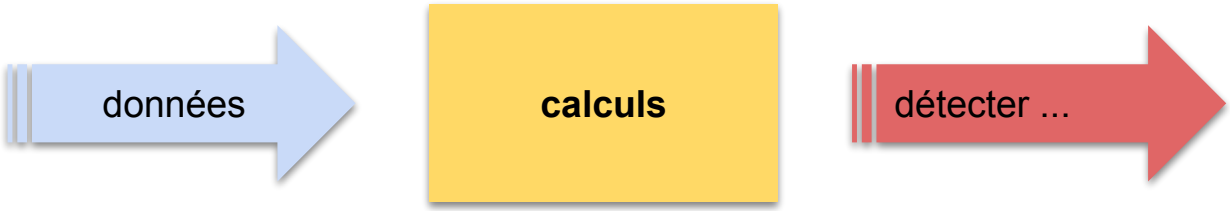
https://fr.mathworks.com/help/stats/machine-learning-in-matlab.html

Structuration



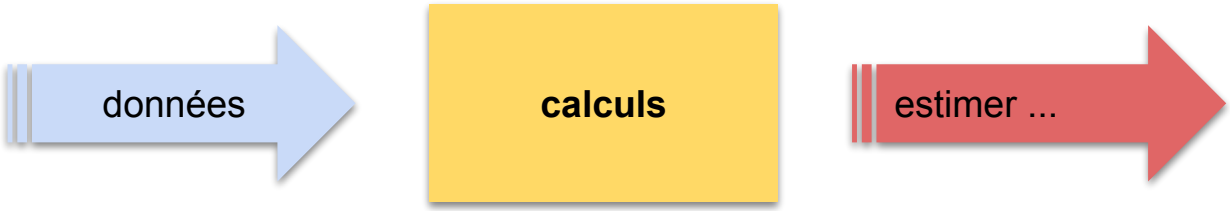
https://fr.wikipedia.org/wiki/Intelligence_artificielle

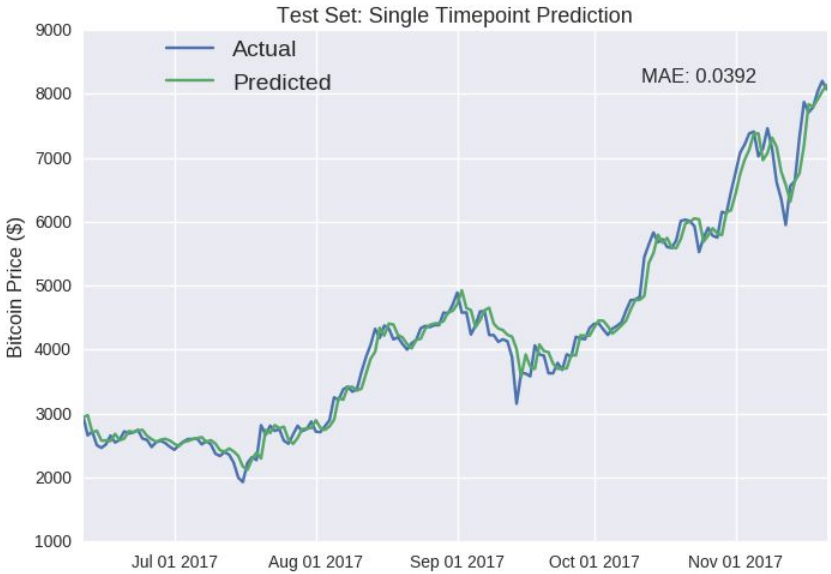
Prédiction dans le présent ⇒ analyser, classifier, identifier, traduire ...

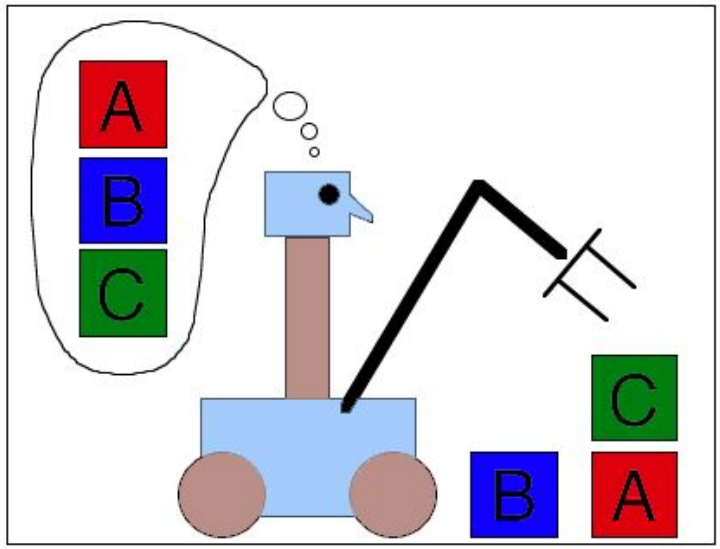


https://fr.wikipedia.org/wiki/Intelligence_artificielle

Prédiction dans le futur ⇒ estimer, planifier ...







https://fr.wikipedia.org/wiki/Intelligence_artificielle

L'IA s'appui sur des modèles:

- Le modèle est une proposition mathématique qui permet de résoudre un problème que l'on observe,

- Le modèle ne donne pas toujours une solution exacte, il existe une erreur plus ou moins importante qu'il faut minimiser,

- Le modèle ne reflète pas toujours la réalité.

https://en.wikipedia.org/wiki/Machine_learning

Quelle est l'approche machine learning ?

1- Analyser les données,

2- Choisir un modèle,

3- Les modèles sont entraînés avec des données (data mining),

4- Estimer l'erreur du modèle,

5- Mettre à jour le modèle.

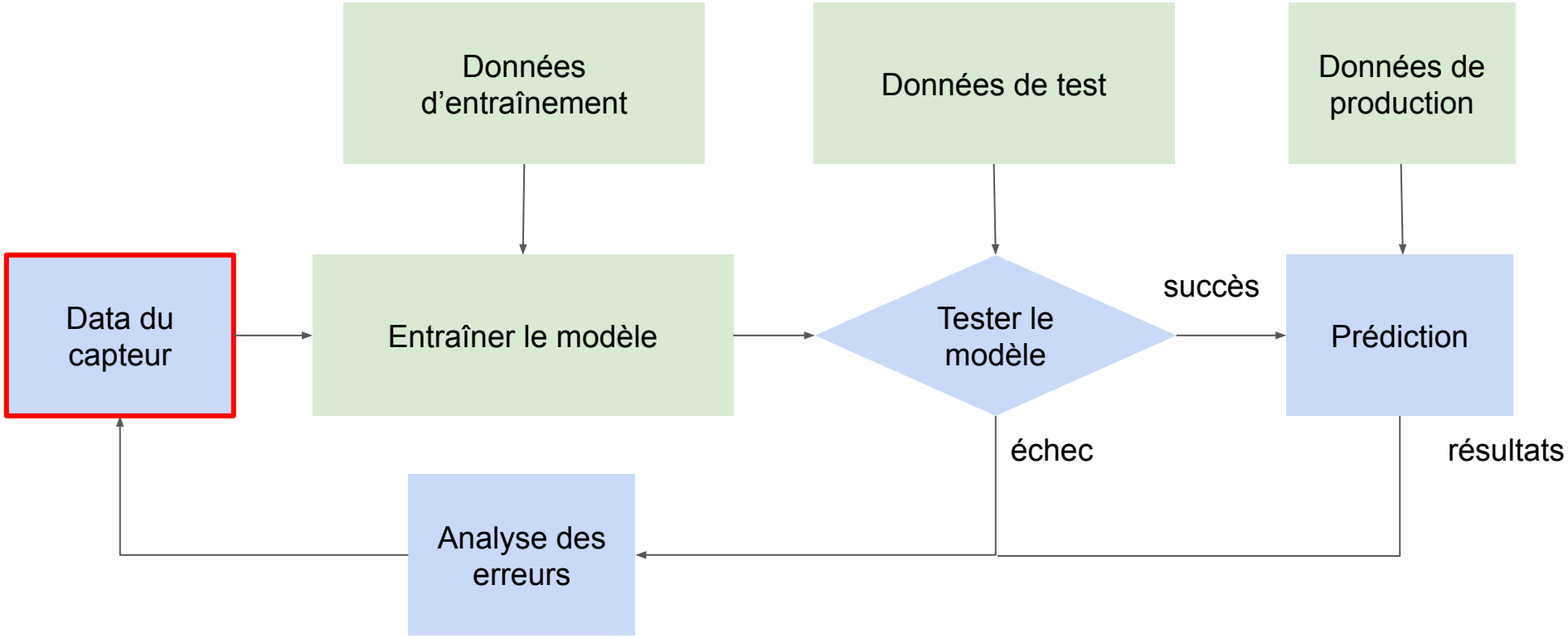
Contraintes:

→ Les données doivent être de très bonne qualité

→ Le volume des données est important pour entraîner le modèle

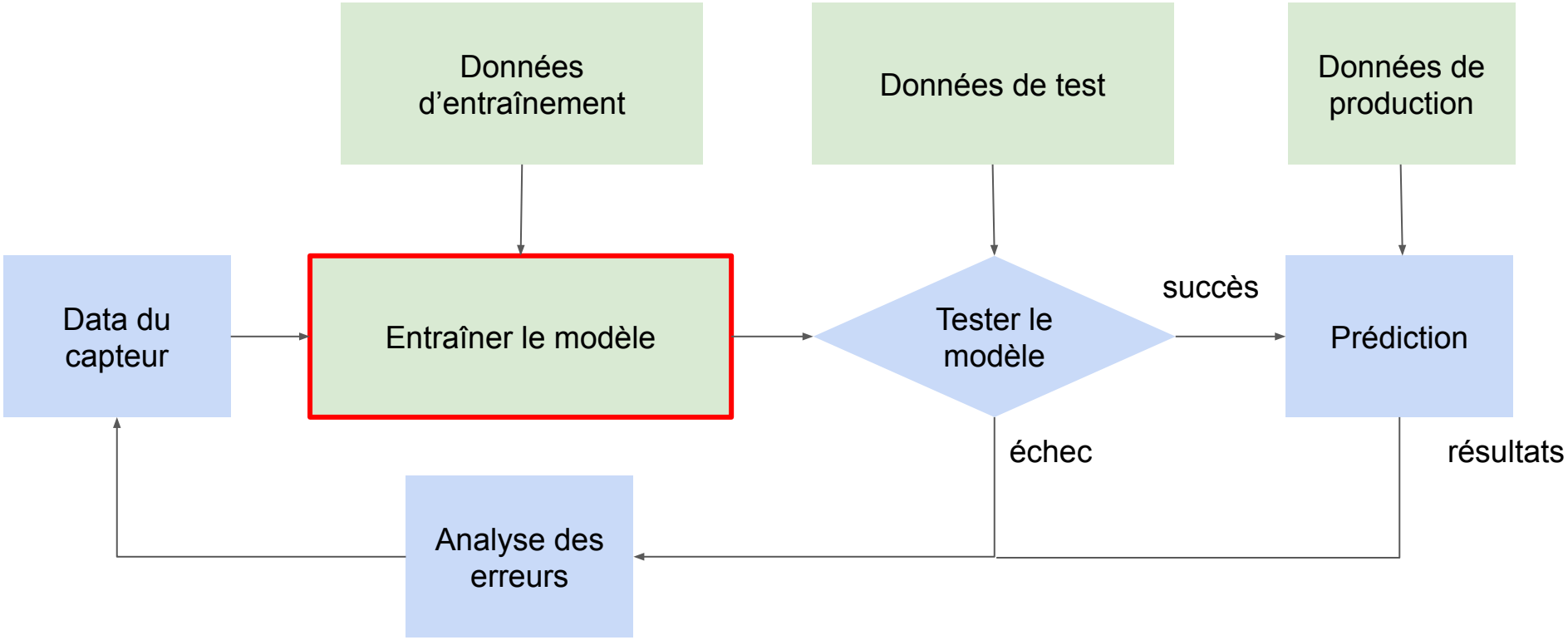
https://en.wikipedia.org/wiki/Machine_learning

Approche Machine Learning et IoT ?



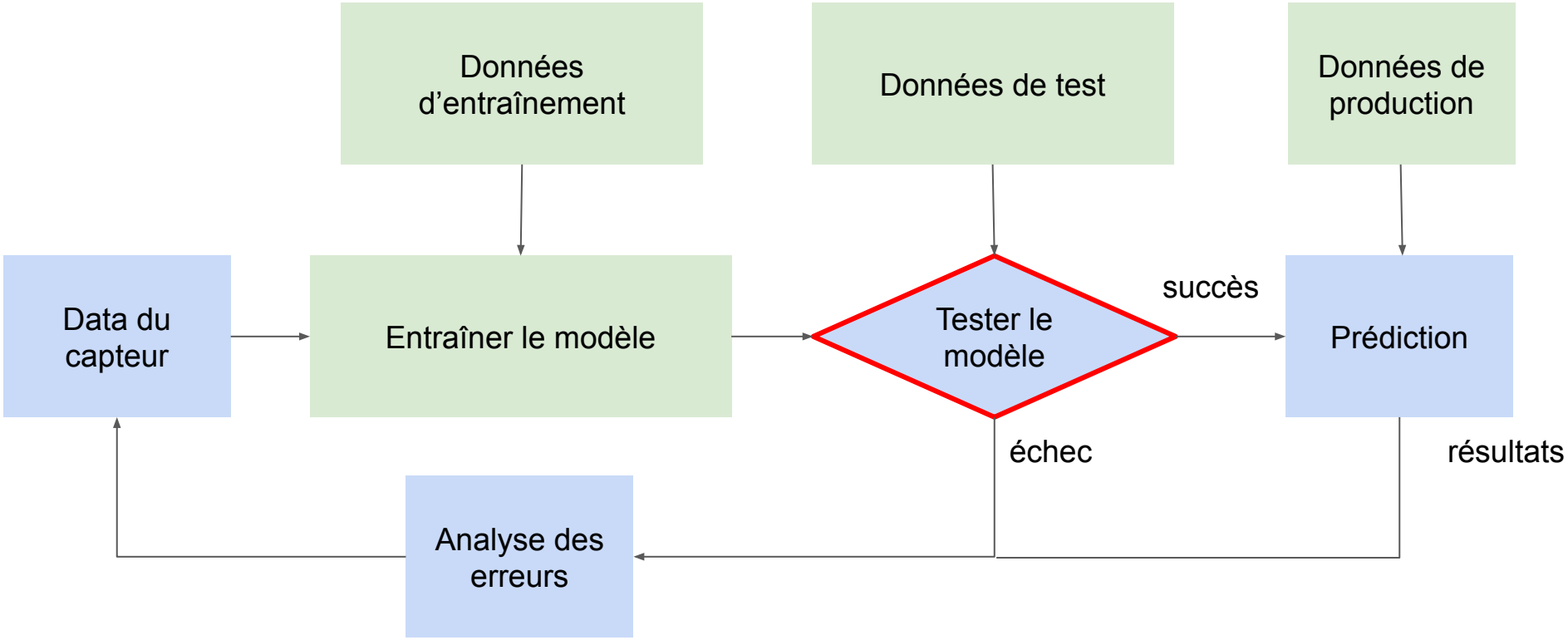
https://en.wikipedia.org/wiki/Machine_learning

Approche Machine Learning et IoT ?



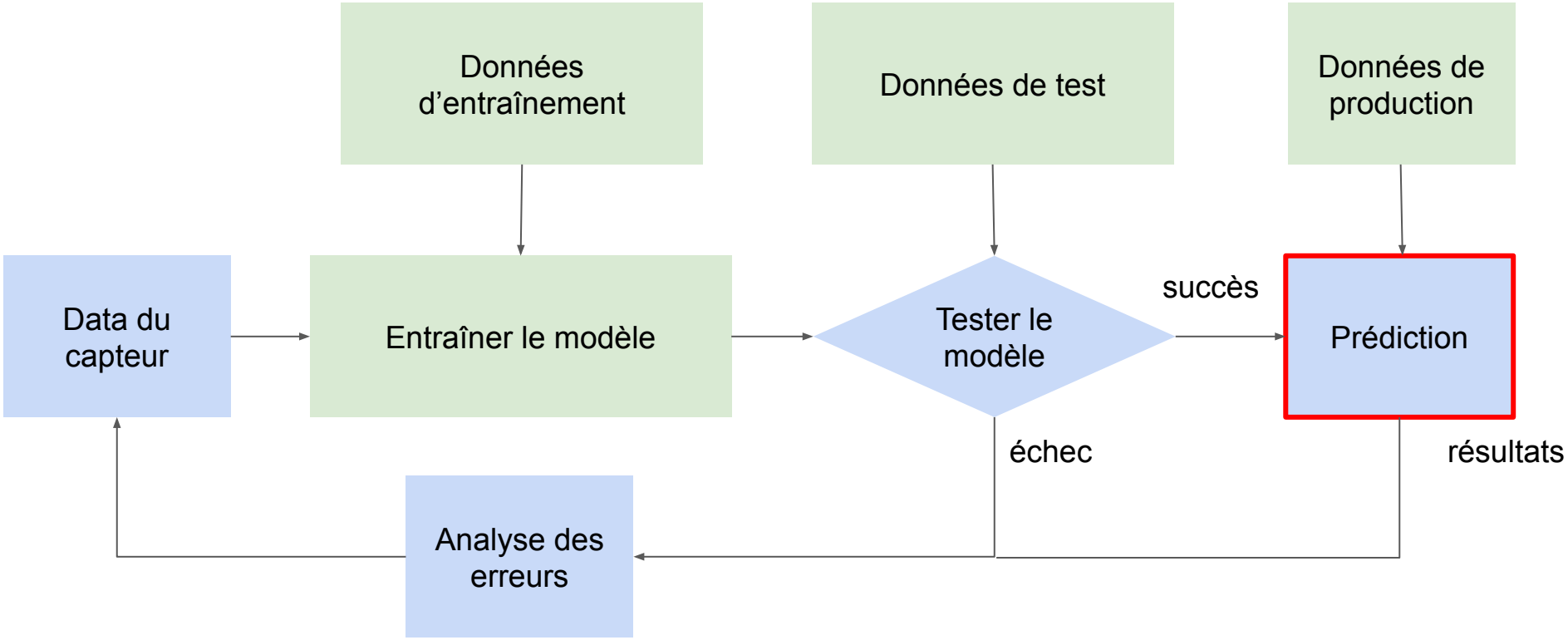
https://en.wikipedia.org/wiki/Machine_learning

Approche Machine Learning et IoT ?



https://en.wikipedia.org/wiki/Machine_learning

Approche Machine Learning et IoT ?



Agenda

Apprentissage supervisé:

- **Régression linéaire simple**
- Régression linéaire multiple
- Equation normale
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

Analyse des séries chronologiques:

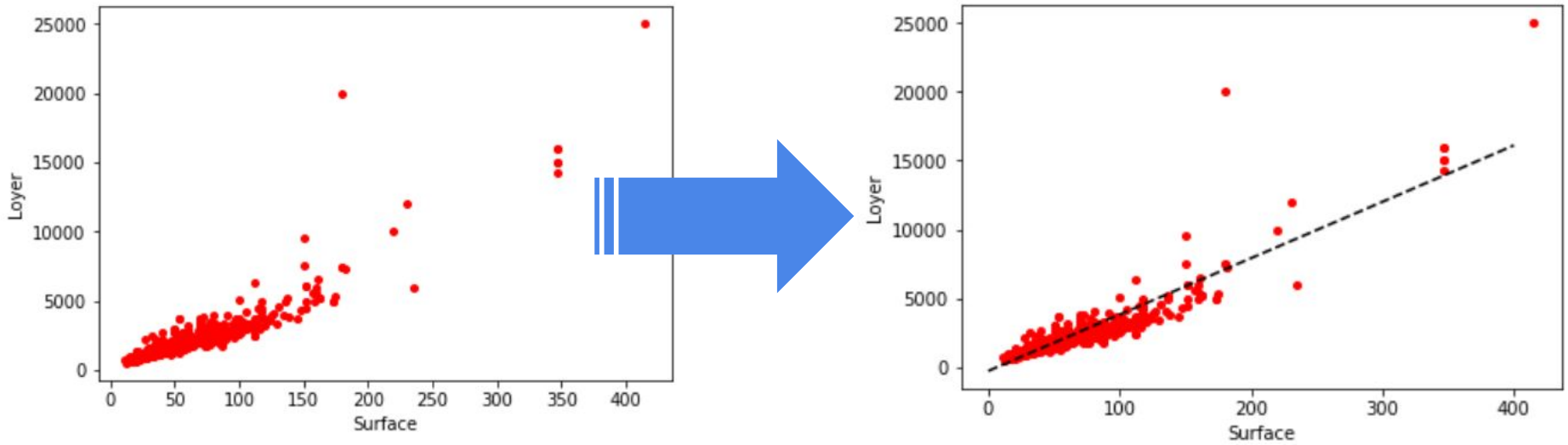
- Modèle ARIMA

Régression linéaire

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

Définition: "Une régression linéaire est un modèle qui cherche à établir une relation linéaire entre une variable, dite expliquée, et une ou plusieurs variables, dites explicatives."

Le modèle de régression linéaire le plus simple est l'ajustement affine.



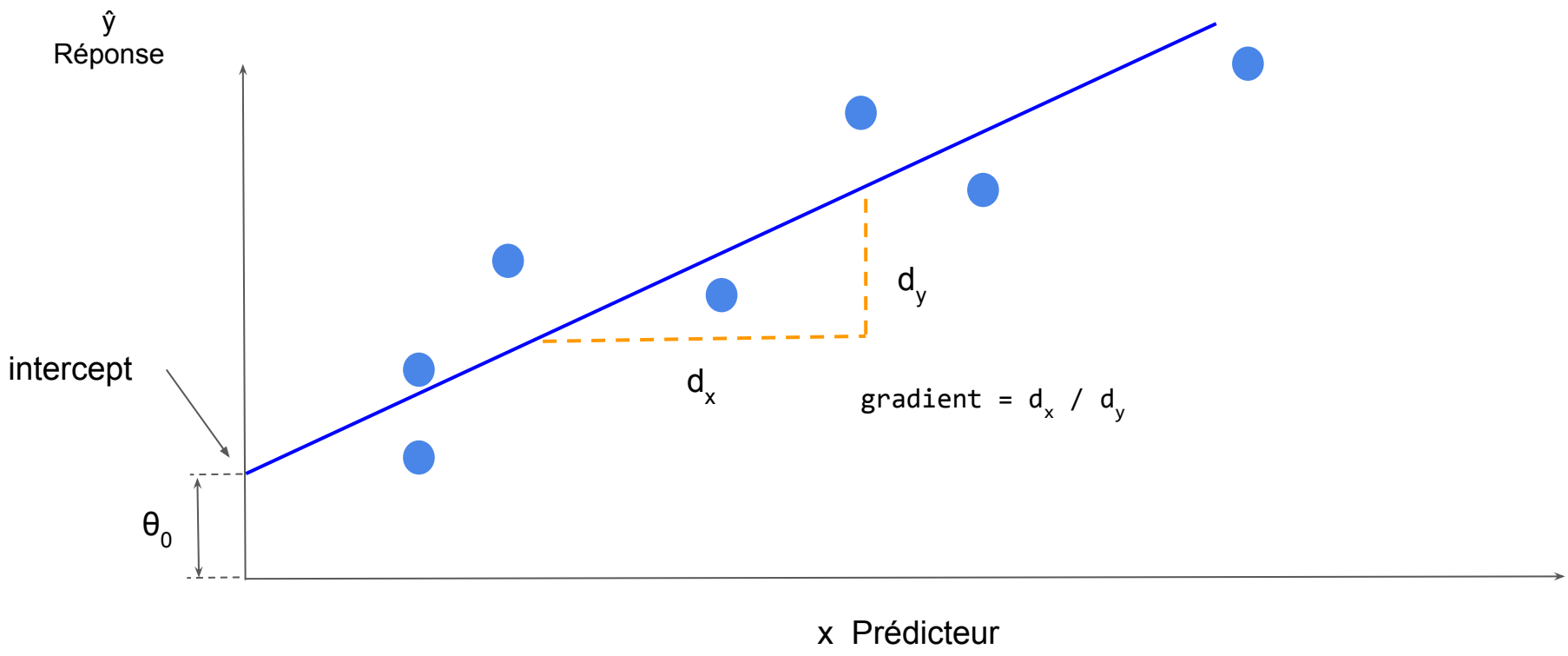
$y = w0 + w1\ x$

Régression linéaire simple

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

Notation affine:

$$\hat{y} = \theta_0 + \theta_1 x_1$$



Régression linéaire simple

https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression?hl=fr

Notation affine:

$$\hat{y} = \theta_0 + \theta_1 x_1$$

$\hat{y}$ est l'étiquette prédite (la sortie désirée) ;

θ_0 est le biais (l'ordonnée à l'origine). Dans certains documents sur le Machine Learning, il est appelé ω_0

θ_1 est la pondération de la caractéristique 1.

x_1 est une caractéristique (une entrée connue).

https://fr.wikipedia.org/wiki/Coefficient_de_d%C3%A9termination
https://en.wikipedia.org/wiki/Coefficient_of_determination

Le coefficient de détermination, noté R^2 ou r^2, est une mesure de la qualité de la prédiction d'une régression linéaire. Dans le cas d'une régression linéaire simple ce coefficient tend vers 1 si le modèle est proche des données.

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

http://larmarange.github.io/analyse-R/multicolinearite.html
https://www.xlstat.com/fr/solutions/fonctionnalites/statistiques-de-multicolinearite

Hypothèses et préparation des données

Comportement linéaire: La régression linéaire suppose que la relation entre une entrée et une sortie est linéaire donc avec une représentation affine $y = ax+b$.

Suppression du bruit: La régression linéaire suppose que les variables d'entrée et de sortie ne sont pas bruitées. Il est nécessaire de filtrer les données. Il est important aussi de supprimer les valeurs aberrantes.

Supprimer la colinéarité: La régression linéaire sur-ajuste les données lorsque les variables sont fortement corrélées. La multicolinéarité n'a aucune incidence sur l'adéquation de l'ajustement, ni sur la qualité de la prévision. Cependant, les coefficients individuels associés à chaque variable explicative ne peuvent pas être interprétés de façon fiable.

http://larmarange.github.io/analyse-R/multicolinearite.html
https://www.xlstat.com/fr/solutions/fonctionnalites/statistiques-de-multicolinearite

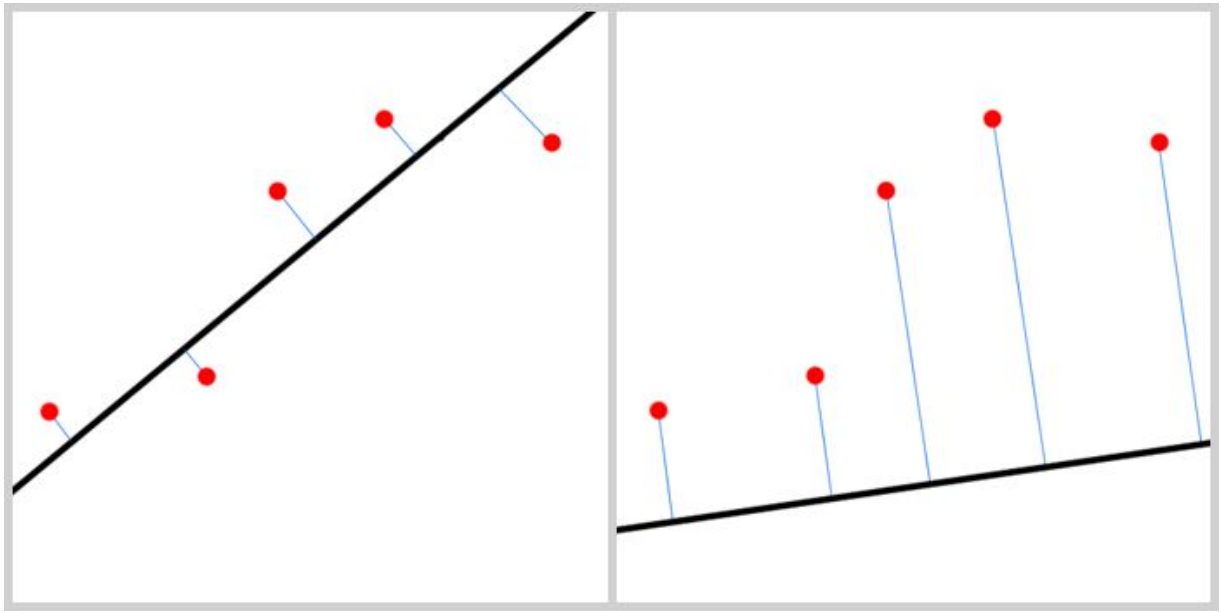
Hypothèses et préparation des données

Distributions Gaussiennes: La régression linéaire réalisera des prédictions plus fiables si les variables d'entrée et de sortie ont une distribution gaussienne.

Redimensionnement des entrées: La régression linéaire rend souvent les prévisions plus fiables en redimensionnant les variables d'entrée avec la normalisation.

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

Minimiser l'erreur : Si on doit approximer un nuage de point avec une droite, alors il faut mesurer l'éloignement des points avec celle-ci. On mesure en principe l'erreur quadratique (somme des distances entre la droite et les points), noté aussi MSE (Mean Square Error) .



https://en.wikipedia.org/wiki/Root-mean-square_deviation

- La RMSE est en générale la fonction de coût utilisée pour les tâches de régression:
  - m le nombre d'observation du jeux de données,
  - $\mathbf{x}^{i}$ le vecteur des valeurs des variables et y^{i} la valeur souhaitée,
  - $\mathbf{X}$ est une matrice contenant toutes la valeurs des variables de toutes les observations du jeux de données,
  - h est la fonction de prédiction du système, pour une observation X^{i},

  alors $\hat{y}^{(i)} = h(x^{(i)})$

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}^{(i)}) - y^{(i)})^2}$$

https://en.wikipedia.org/wiki/Root-mean-square_deviation

Autre facteur mesure de performance:

- Mean Absolute Error (MAE), la moyenne de la valeur absolue des erreurs:

$$\frac{1}{n}\sum_{i=1}^{n}|Actual - Predicted|$$

- Mean Squared Error (MSE), la moyenne des erreurs quadratiques:

$$\frac{1}{n}\sum_{i=1}^{n}|Actual - Predicted|^2$$

Régression linéaire simple

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

Notation affine:

$$\hat{y} = \theta_0 + \theta_1 x$$

Pente de la droite (slope)

$$\theta_1 = \frac{\sum_{i=1}^{n} (x_i - mean(x))\ (y_i - mean(y))}{\sum_{i=1}^{n} (x_i - mean(x))^2}$$

Ordonnée à l'origine (intercept)

$$\theta_0 = mean(y) - \theta_1\ mean(x)$$

Estimation de l'erreur :

$$\mathrm{RMSE}\,(\mathbf{X},\ h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}^{(i)}) - y^{(i)})^2}$$

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- **Régression linéaire multiple**
- Equation normale
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

Analyse des séries chronologiques:

- Modèle ARIMA

Régression linéaire multiple

https://fr.wikipedia.org/wiki/Apprentissage_supervis%C3%A9

Supervisé :

- Prédire une valeur cible à partir d'autres données (feature) y=f(x1, x2, x3)

Entraînement:

- Trouver les paramètres du modèle y=f(x1, x2, x3) pour minimiser la fonction coût

Dataset:

- y: target ; x: feature

Exemple:

- Classification de mail comme spam ou non spam …

Régression linéaire multiple

https://fr.wikipedia.org/wiki/Apprentissage_supervis%C3%A9

Exemple :

Identifier le prix en fonction de la marque, moteur, modèle: $y = f(X1, X2, X3)$

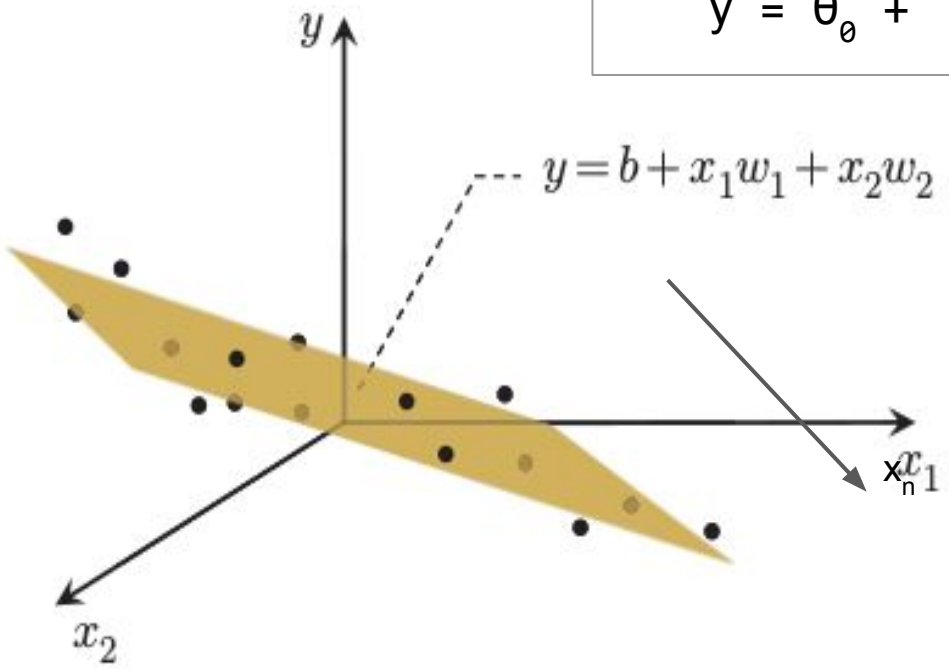
Prix	Marque	Puissance Moteur	Modèle
100000	Ferrari	500	F1
3000	Citroen	2	2CV
400000	Porsche	460	C1
TARGET	**FEATURES**		
y	X1	X2	X3

Régression linéaire multiple

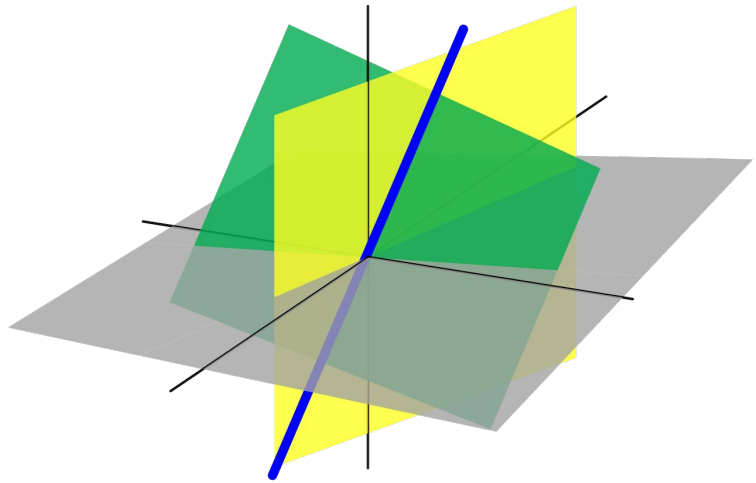
https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire_multiple

Notation scalaire:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$



Un modèle de régression linéaire en deux dimensions est une ligne droite; en trois dimensions c'est un plan, et en plus de trois dimensions, un hyperplan.



$\hat{y}$: Estimateur, c'est la prédiction

n : Le nombre de variables,

x_i : La valeur de $i^{ième}$ variable

θ_n : le $n^{ième}$ paramètre du modèle (terme constant θ_0 et coefficients de pondération des variables θ_1 θ_2 θ_3... θ_n)

https://en.wikipedia.org/wiki/Root-mean-square_deviation

Rappel: Notation de transposée:

X^T = Transposée du vecteur X

La transposée d'un vecteur colonne est un vecteur ligne.

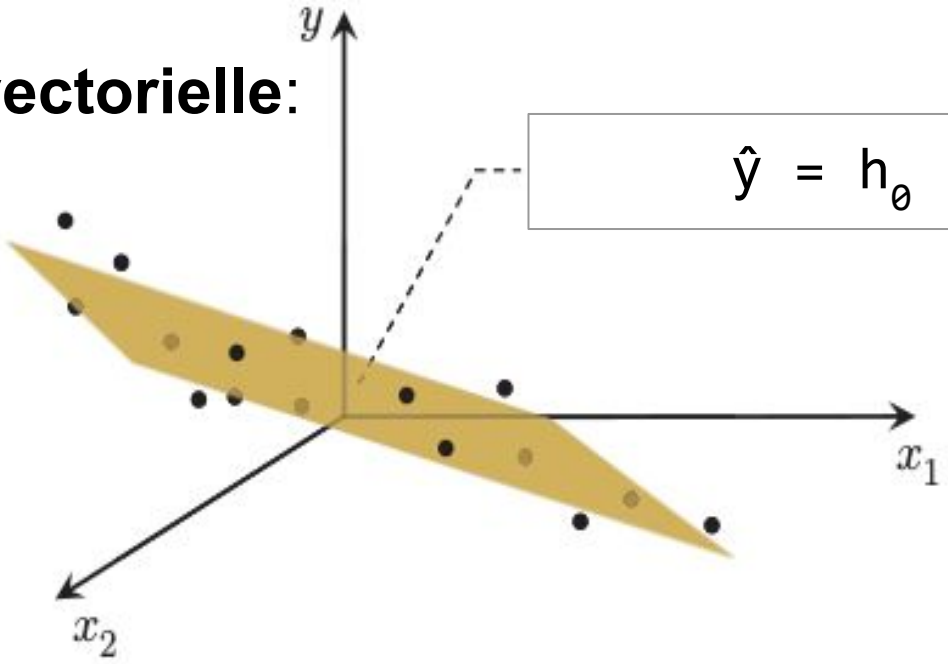
$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \Leftrightarrow \mathbf{X}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$$

$$\mathbf{x}^{(1)} = \begin{pmatrix} -118.29 \\ 33.91 \\ 1,416 \\ 38,372 \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(1999)})^T \\ (\mathbf{x}^{(2000)})^T \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1,416 & 38,372 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Régression linéaire multiple

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

Notation vectorielle:



$$\hat{y} = h_0(X) = \theta^T . x$$

$\hat{y}$: Estimateur, c'est la prédiction

θ^T : transposée des paramètres de pondération (vecteur ligne au lieu d'un vecteur colonne), avec un terme constant θ_0 et les coefficients de pondération variables θ_1 θ_2 θ_3... θ_n)

X : vecteur des valeurs d'une observation avec x_0=1 puis jusqu'à Xn

θ^T . x : Produit scalaire

h0 : la fonction hypothèse

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- **Régression linéaire équation normale**
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

Analyse des séries chronologiques:

- Modèle ARIMA

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

Il existe une solution analytique exacte qui permet de trouver le vecteur θ. Il s'agit de l'équation normale :

$$y = \beta_0 + \beta_1 x_1 + \beta_n x_n + \varepsilon$$

$$Y = XB \Rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \quad \text{avec} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, B = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Solution: ⇨ $\beta = Y/X$

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

Equation normale :

$$\beta = Y/X$$

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

X : Jeux d'entraînement

y : Vecteur des valeurs cibles d'entraînement,

β^T : Vecteur de β qui minimise la fonction de coût,

Conséquences :

⇒ La solution est exacte.

⇒ Il faut inverser une matrice (pas toujours possible)

⇒ L'inversion d'une matrice est très coûteux en temps et mémoire

⇒ Le temps de calcul peut être très long

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Régression linéaire équation normale
- **Régression linéaire descente de gradient**
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

Analyse des séries chronologiques:

- Modèle ARIMA

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

Principe:

- Algorithme d'optimisation très général,
- Capacité de trouver des solutions optimales,
- Le but est de modifier les paramètres de l' équation pour minimiser sa fonction de coût par exemple la fonction RMSE

Conséquences :

⇒ Stockage des données = utilisation mémoire,

⇒ Exécution en une passe,

⇒ Plus rapide que l'équation normale,

⇒ Garantie d'avoir une solution car l'équation normale n'est pas "garantie",

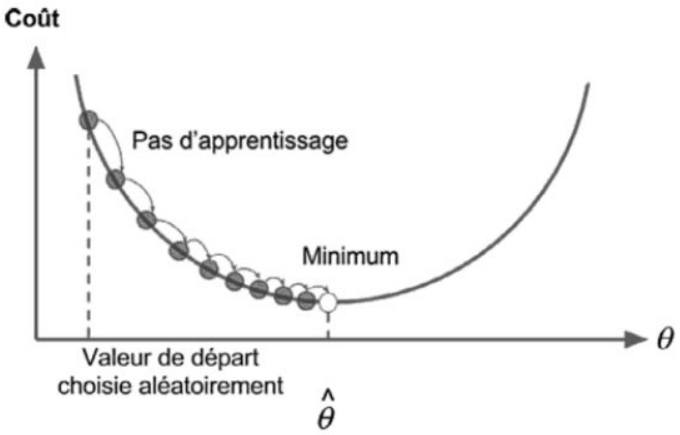
⇒ Estimateur moins précis que l'équation normale.

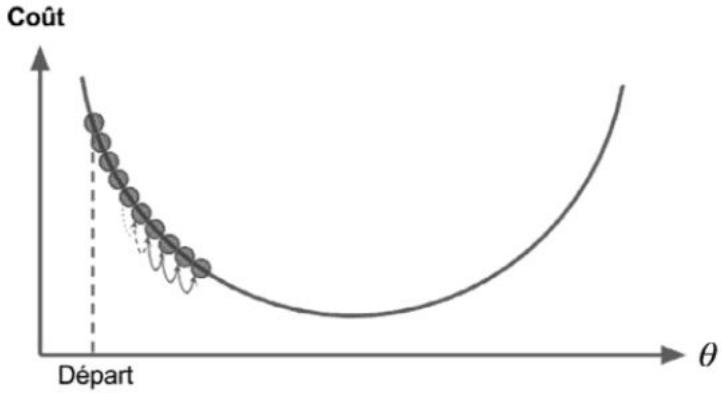
Régression linéaire

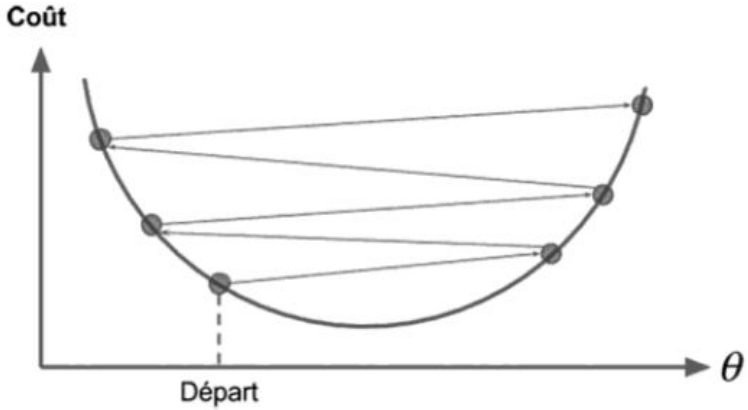
Descente de gradient ordinaire

https://fr.wikipedia.org/wiki/Algorithme_du_gradient

- L'algorithme du gradient désigne un algorithme d'optimisation différentiable.
- L'idée générale de Gradient Descent est de modifier les paramètres de manière itérative afin de minimiser une fonction de coût MSE.
- L'algorithme est itératif et procède donc par améliorations successives.

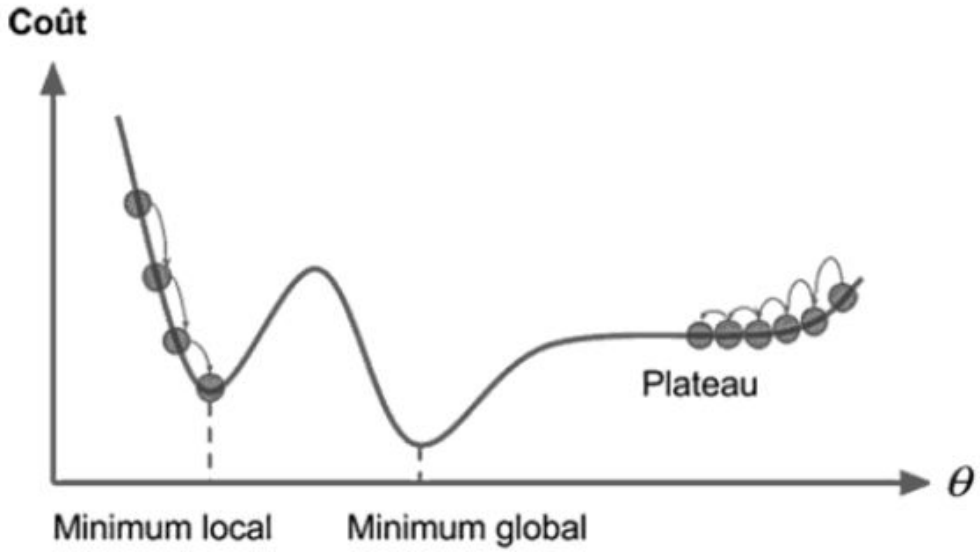






https://fr.wikipedia.org/wiki/Algorithme_du_gradient

- La fonction de coût MSE du modèle de régression linéaire est une fonction convexe:
  - Il n'y a pas de minima locaux,
  - La fonction est continue,
- Les propriétés de cette fonction de coût permet d'avoir une descente de gradient qui se rapproche du minimum global.



https://fr.wikipedia.org/wiki/Algorithme_du_gradient

La fonction de coût MSE du modèle de régression linéaire

$$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\boldsymbol{\theta}) = \frac{2}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

$$\nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \mathrm{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \mathrm{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \mathrm{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

$$\boldsymbol{\theta}^{(\text{étape suivante})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta})$$

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Régression linéaire équation normale
- Régression linéaire descente de gradient
- **Régression linéaire descente de gradient stochastique (SGD)**
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

Analyse des séries chronologiques:

- Modèle ARIMA

Régression linéaire

Descente de gradient stochastique (SGD)

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

- La descente de gradient stochastique sélectionne simplement une instance aléatoire dans l'ensemble d'apprentissage,
- Calcule les gradients en fonction uniquement d'une seule instance,
- Algorithme beaucoup plus rapide car il a très peu de données à manipuler à chaque itération,
- Il permet également de s'entraîner sur d'énormes ensembles d'entraînement, car une seule instance doit être en mémoire à chaque itération,

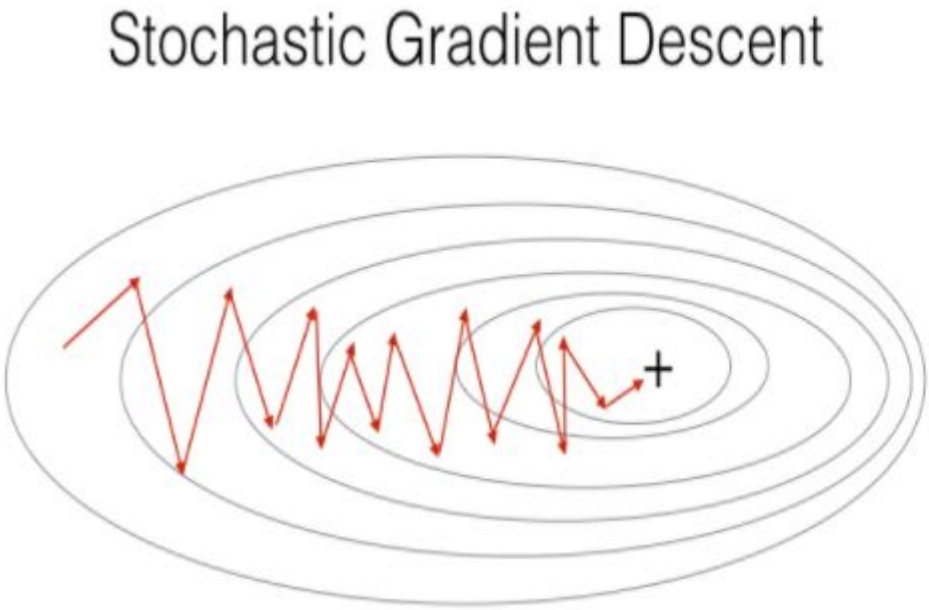
https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

- La fonction de coût rebondira de haut en bas, ne diminuant qu'en moyenne,
- Une fois l'algorithme arrêté, les valeurs finales des paramètres sont bonnes, mais pas optimales,

Régression linéaire

Descente de gradient stochastique (SGD)

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s



Exemple

```
...

def learning_schedule(t):
    return t0 / (t + t1)

m = len(X_b)

for epoch in range(n_iterations):
     for i in range(m):
        #print(i)
        random_index = rnd.randint(m)

        xi = X_b[random_index:random_index+1]
        yi = y[random_index]

        gradients = 2 * xi.T.dot(xi.dot(theta) - yi)
        eta = learning_schedule(epoch * m + i)
        theta = theta - eta * gradients
theta

...
```

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Régression linéaire équation normale
- Régression linéaire descente de gradient
- Régression linéaire descente de gradient stochastique (SGD)
- **Descente de gradient mini-lot (mini-batch)**
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

Analyse des séries chronologiques:

- Modèle ARIMA

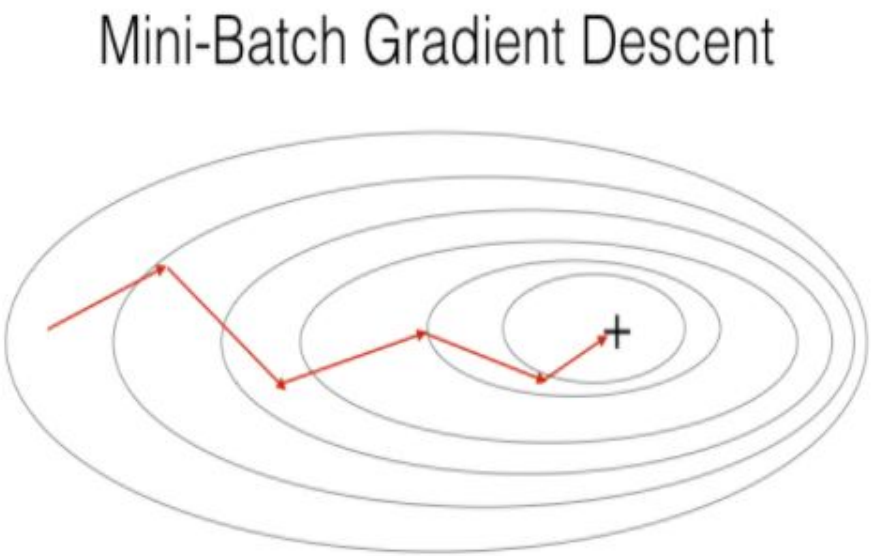
https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s

- Mini-batch calcule les gradients sur de petits ensembles aléatoires d'instances appelés mini-lots,
- Progression moins désordonnée que la version stochastique,
- Plus rapide que la descente de gradient ordinaire.

Régression linéaire

Descente de gradient mini-lot (mini-batch)

https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire
https://fr.wikipedia.org/wiki/M%C3%A9thode_des_moindres_carr%C3%A9s



Exemple

```
theta_path_mgd = []
n_iterations = 50
minibatch_size = 20
np.random.seed(42)
theta = np.random.randn(2,1)  # random initialization

t0, t1 = 200, 1000

def learning_schedule(t):
    return t0 / (t + t1)

t = 0
for epoch in range(n_iterations):
    shuffled_indices = np.random.permutation(m)
    X_b_shuffled = X_b[shuffled_indices]
    y_shuffled = y[shuffled_indices]
    for i in range(0, m, minibatch_size):
        t += 1
        xi = X_b_shuffled[i:i+minibatch_size]
        yi = y_shuffled[i:i+minibatch_size]
        gradients = 2/minibatch_size * xi.T.dot(xi.dot(theta) - yi)
        eta = learning_schedule(t)
        theta = theta - eta * gradients
        theta_path_mgd.append(theta)
print(theta)
```

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Régression linéaire équation normale
- Régression linéaire descente de gradient
- Régression linéaire descente de gradient stochastique (SGD)
- Descente de gradient mini-lot (mini-batch)
- **Régression polynomiale**
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

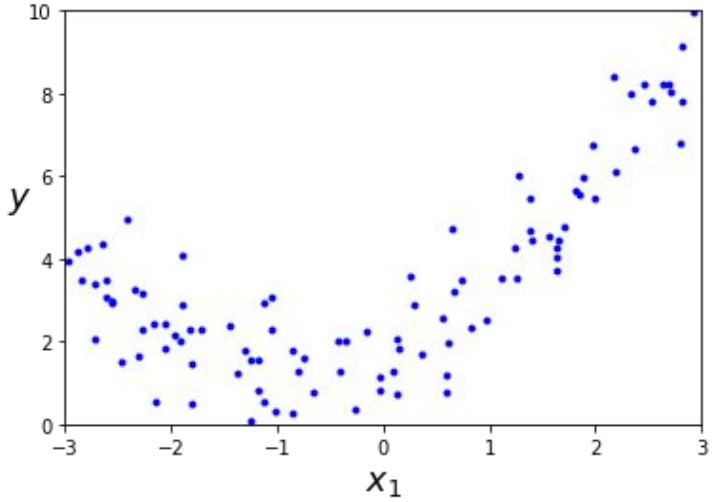
Analyse des séries chronologiques:

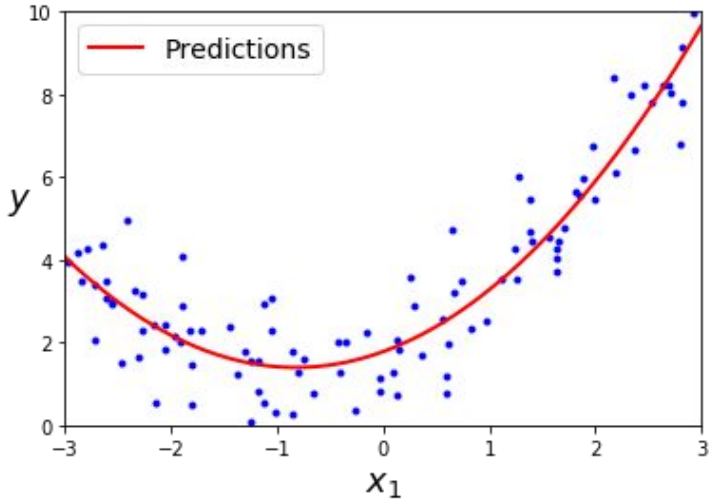
- Modèle ARIMA

Régression polynomiale

https://fr.wikipedia.org/wiki/R%C3%A9gression_polynomiale

Une droite ne peut pas tout ajuster:






```
import numpy as np
import numpy.random as rnd
import matplotlib as mpl
import matplotlib.pyplot as plt

np.random.seed(42)

m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)

plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([-3, 3, 0, 10])
plt.show()
```

Régression polynomiale

https://fr.wikipedia.org/wiki/R%C3%A9gression_polynomiale

Exemple

Utiliser matlab pour résoudre la série suivante:

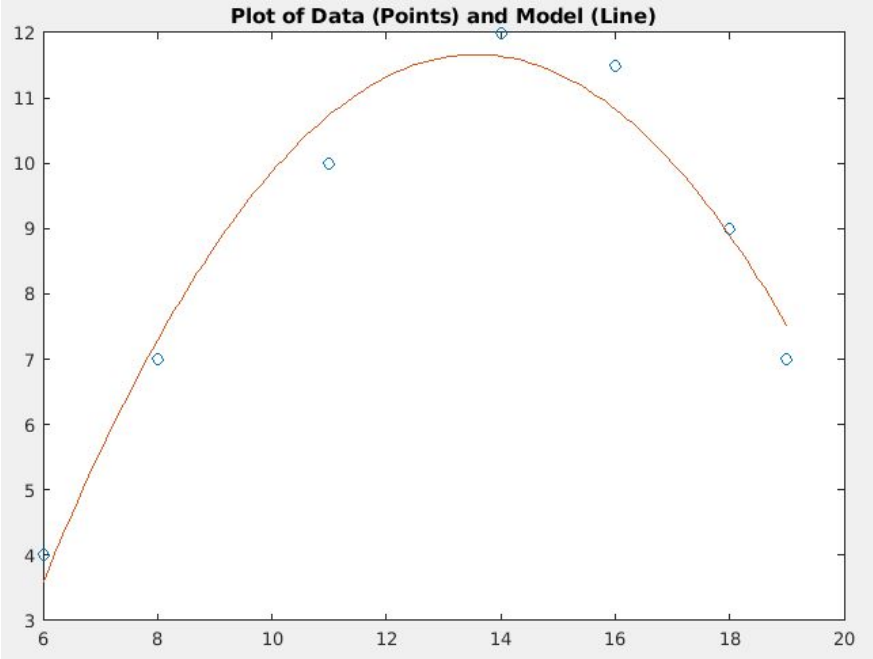
Time = [6 8 11 14 16 18 19];
Temp = [4 7 10 12 11.5 9 7];

Solution :

coeff = -0.1408 X^2 + 3.8207 X -14.2562

Ressources:

https://fr.mathworks.com/help/matlab/ref/polyfit.html
https://fr.mathworks.com/help/symbolic/coeffs.html



Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Régression linéaire équation normale
- Régression linéaire descente de gradient
- Régression linéaire descente de gradient stochastique (SGD)
- Descente de gradient mini-lot (mini-batch)
- Régression polynomiale
- **Machine à vecteurs de support linéaire et non linéaire (SVM)**
- Réseau de neurone,

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

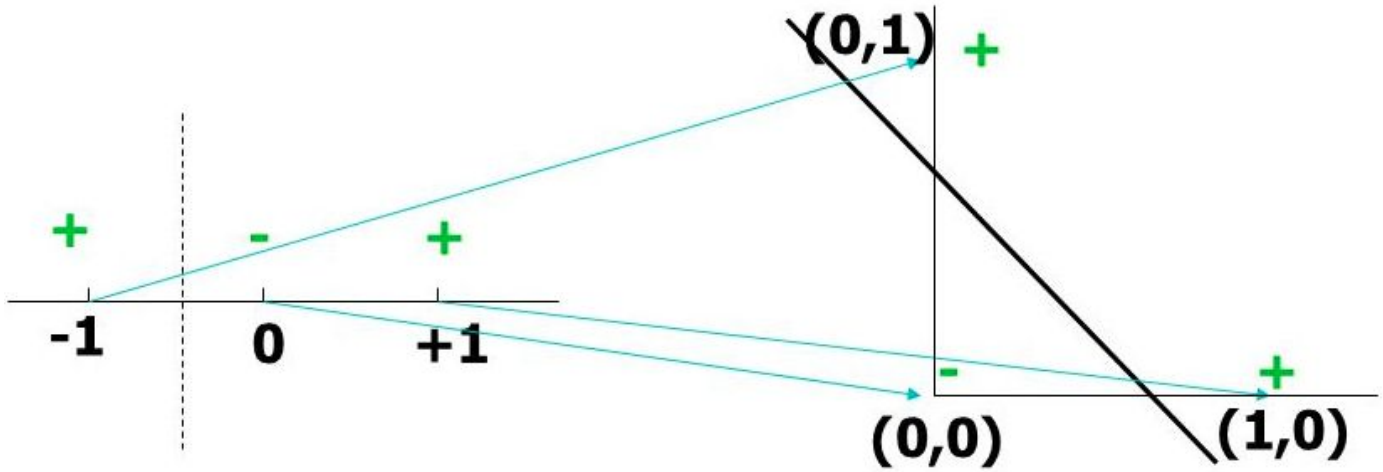
Analyse des séries chronologiques:

- Modèle ARIMA

https://scikit-learn.org/stable/modules/svm.html#svm-kernels
https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_de_Mercer

SVM non linéaire

- Transposer les données dans un espace de plus grande dimension pour trouver un hyperplan séparateur.
- Utiliser la notion de “fonction noyau “ pour définir un espace de projection
  - Linéaire
  - Polynomial
  - Radial gaussien (RBF: Radial Basis Function)
  - Sigmoïde ...



Classification
Machine à vecteurs de support linéaire (SVM*)

https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

- Une machine à vecteur de support permet de délimiter des données, il s'agit d'un apprentissage automatique utilisé pour des classifications linéaires multi-classes, non linéaires, régression.
- Très efficace quand on ne dispose que de peu de données d'entraînement, SVM s'adresse à des données de taille réduite.
- Le but d'un SVM est de trouver une frontière optimale, en maximisant la distance entre les points d'entraînement et la frontière,
- Les points d'entraînement les plus proches de la frontière sont appelés vecteurs support.

*SVM: Support Vector Machine

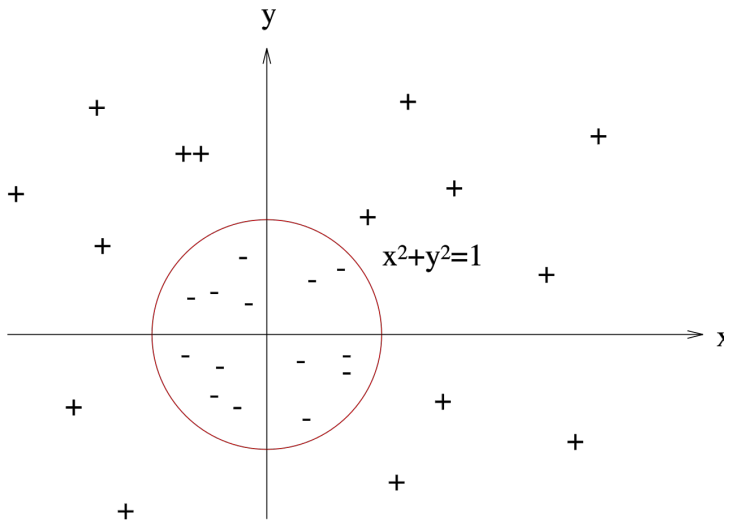
Classification

Machine à vecteurs de support linéaire (SVM*)

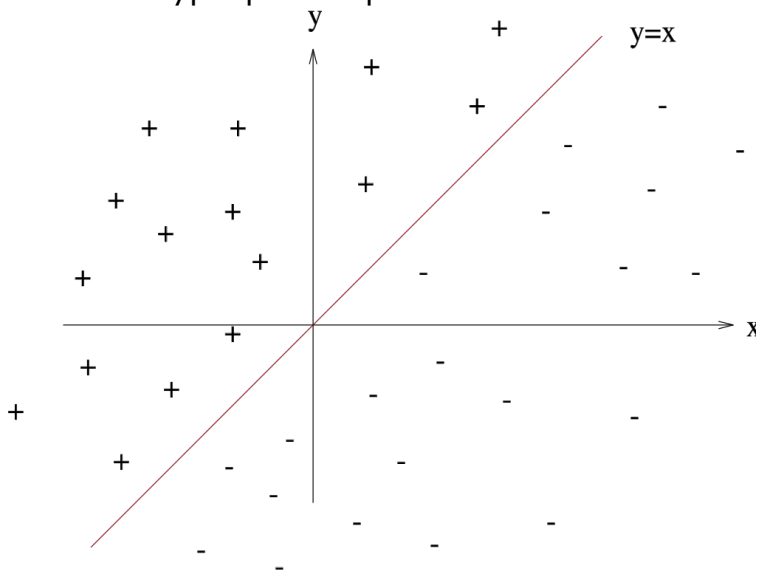
https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

Une machine à vecteur de support linéaire permet de délimiter des données linéairement séparable:

Le problème n'est pas linéairement séparable. Il n'existe pas d'hyperplan séparateur.



Le problème est linéairement séparable. Il existe un hyperplan séparateur



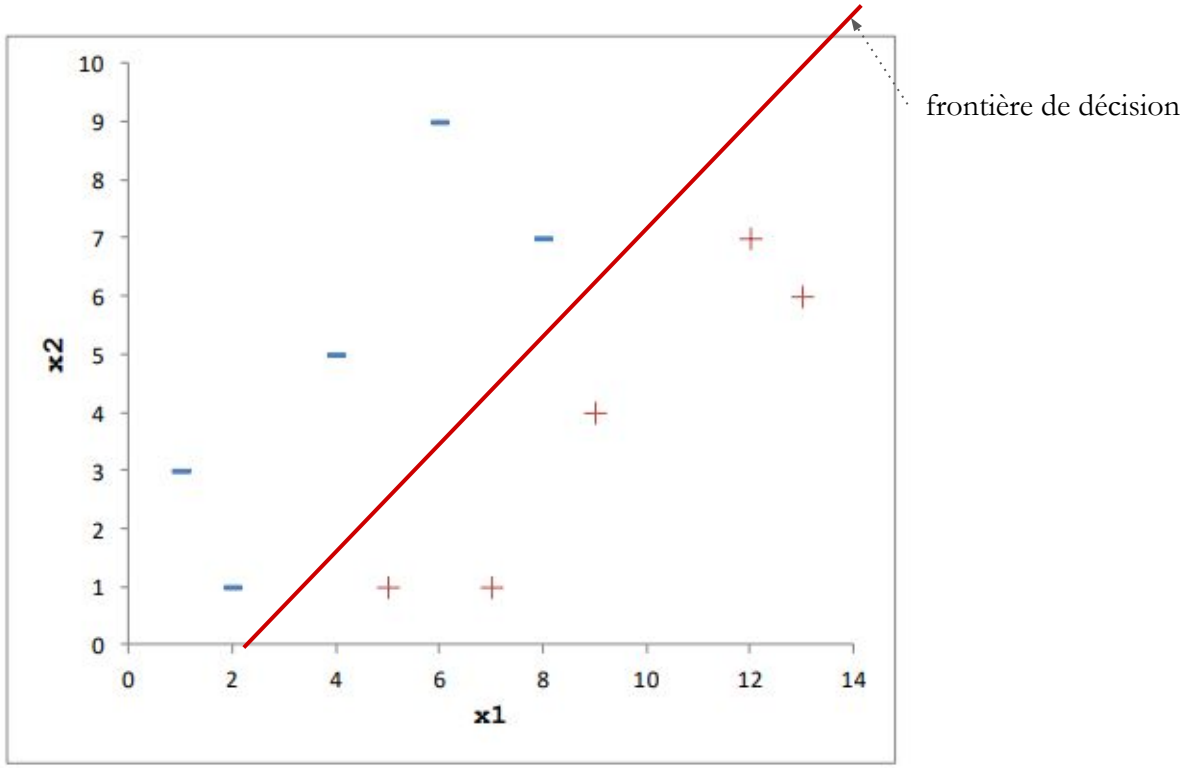
*SVM: Support Vector Machine

Classification
Machine à vecteurs de support linéaire (SVM*)

https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

Une machine à vecteur de support = apprentissage automatique supervisé, ici on définit X1 , X2 et **Y** (Y représente une classification supervisée car on définit sa classe soit "-" soit "+"):

x1	x2	y
1	3	-1
2	1	-1
4	5	-1
6	9	-1
8	7	-1
5	1	1
7	1	1
9	4	1
12	7	1
13	6	1



⇒ Le but de SVM est de trouver le chemin le plus large possible et une frontière de décision.

*SVM: Support Vector Machine

Classification

Machine à vecteurs de support linéaire (SVM)

https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

La frontière optimale consiste à maximiser la distance entre les points d'entraînement et une frontière: on maximise la distance **d** entre la frontière et les points d'entraînement (fig.1). Il existe une infinité de frontière non-optimales (fig.2).

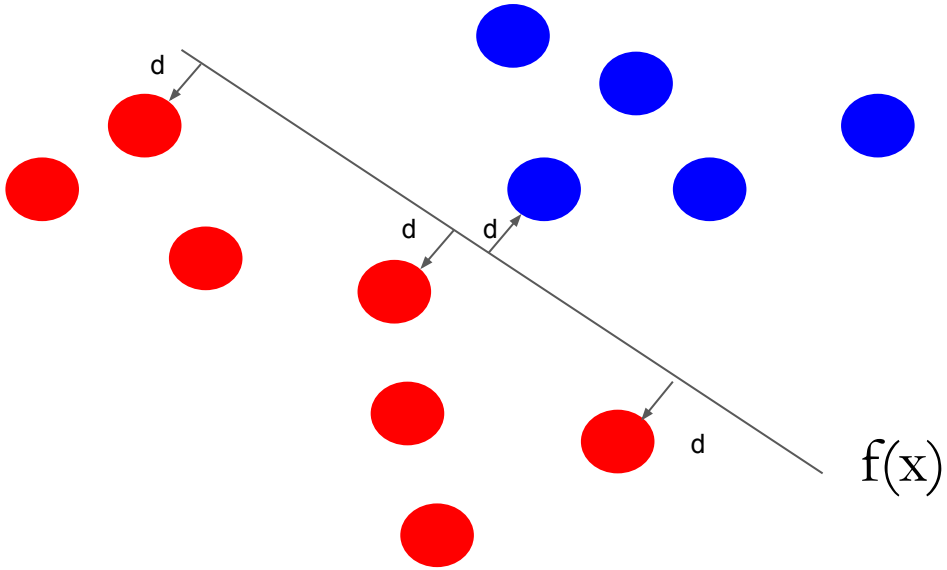


figure 1: il existe une unique frontière optimale qui maximise **d**

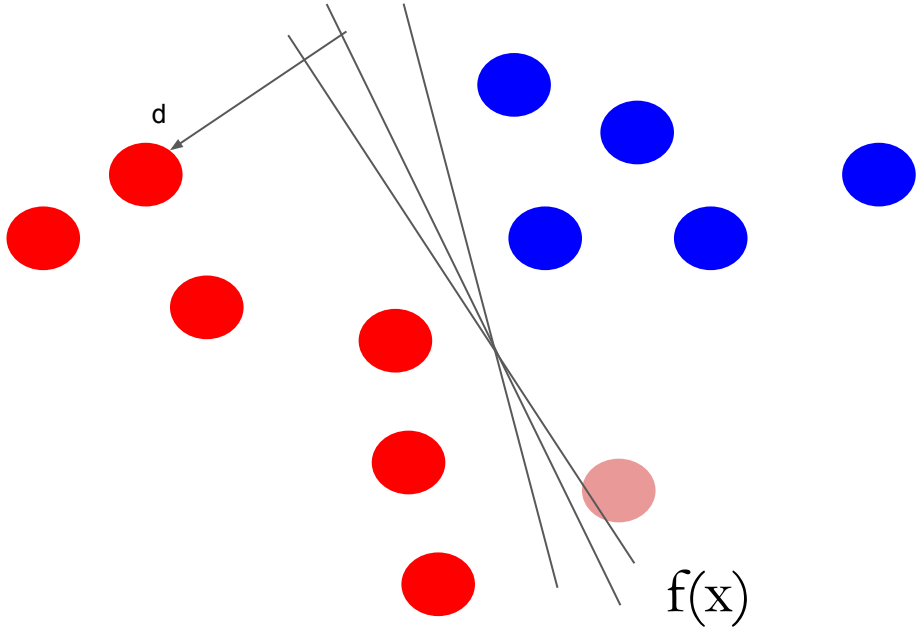


figure 2: les frontières non optimales passent très près des points d'entraînement, il existe une infinité de plan.

$$f(x) = X^T\beta+\beta_0 = x_1\beta_1 + x_2\beta_2 + \beta_0$$

Classification
Machine à vecteurs de support linéaire (SVM*)

https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

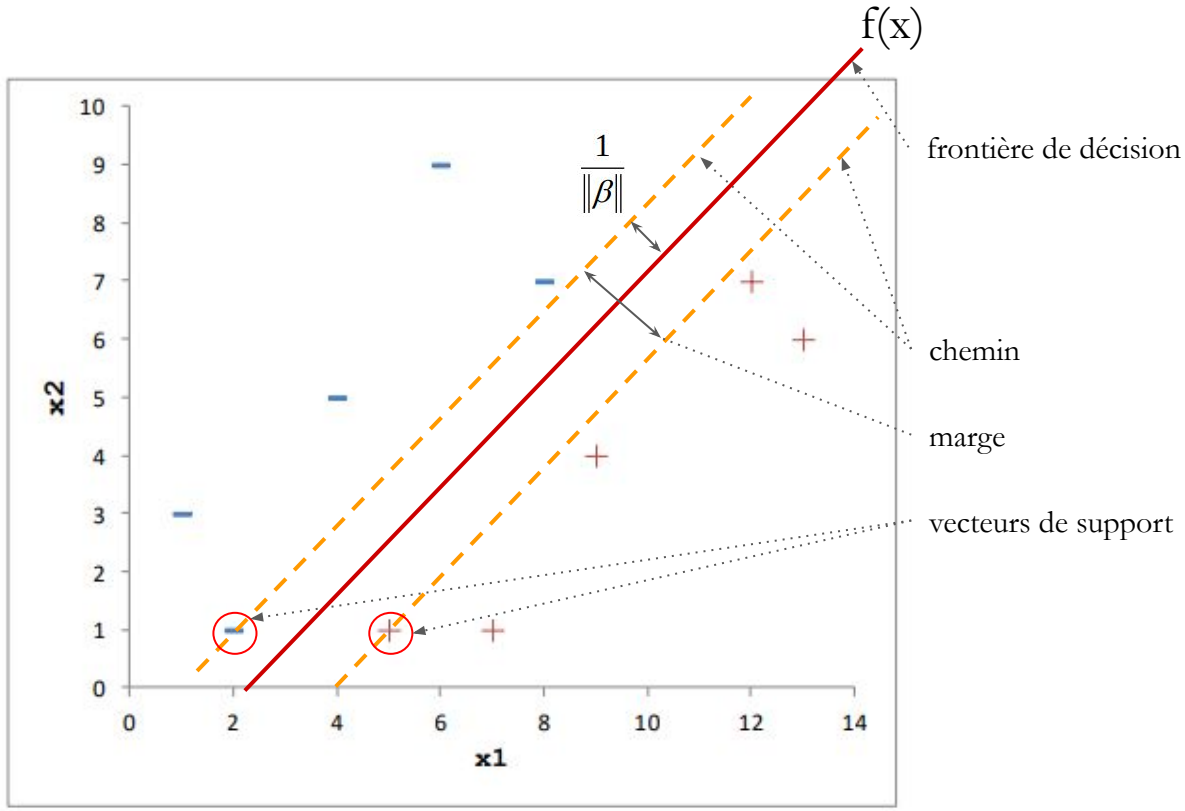
Une machine à vecteur de support = problème d'optimisation convexe (fonction objectif quadratique, contraintes linéaires). Un optimum global existe.

f(x) = 0, max de la marge
f(x) > 0, zone des « + »
f(x) < 0, zone des « - »
f(x) = +1 chemin à droite
f(x)= -1 chemin à gauche

Marge maximum: $\delta = \frac{2}{\|\beta\|}$

Maximiser la marge revient à minimiser le vecteur β:

$$\max \frac{2}{\|\beta\|} \Leftrightarrow \min \|\beta\|$$



$$f(x) = X^T\beta + \beta_0 = x_1\beta_1 + x_2\beta_2 + \beta_0$$

*SVM: Support Vector Machine

Classification
Machine à vecteurs de support linéaire (SVM*)

https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

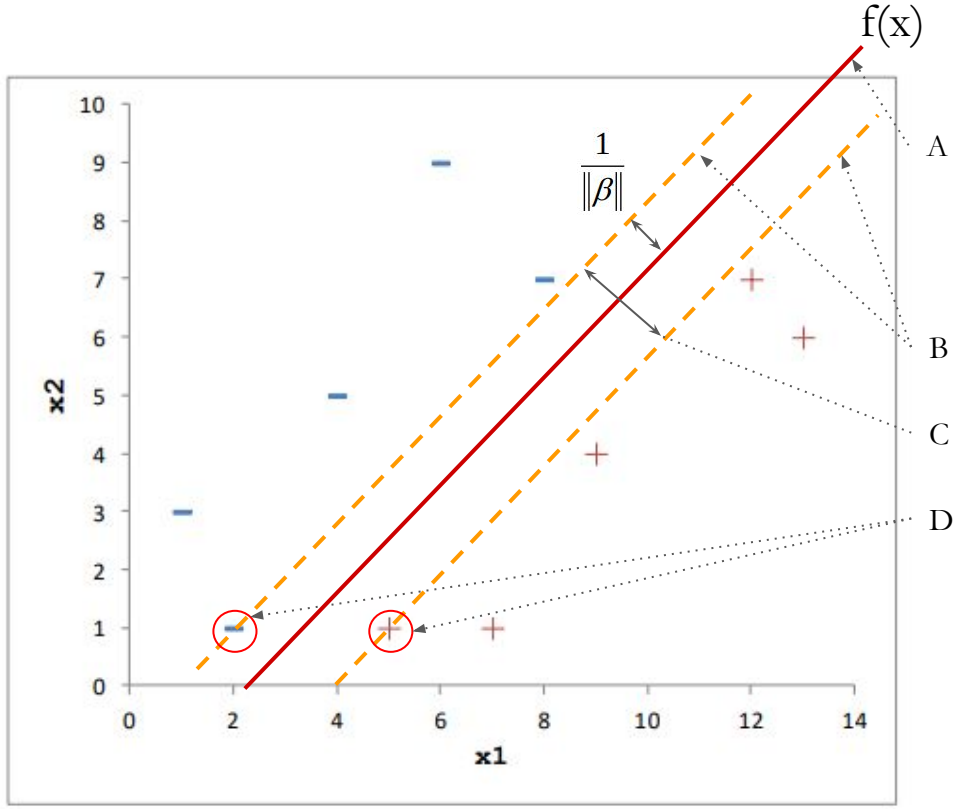
Une machine à vecteur de support = problème d'optimisation convexe (fonction objectif quadratique, contraintes linéaires). Un optimum global existe.

f(x) = 0, max de la marge
f(x) > 0, zone des « + »
f(x) < 0, zone des « - »
f(x) = +1 chemin à droite
f(x)= -1 chemin à gauche

Marge maximum: $\delta = \frac{2}{\|\beta\|}$

Maximiser la marge revient à minimiser le vecteur β:

$$\max \frac{2}{\|\beta\|} \Leftrightarrow \min \|\beta\|$$



$$f(x) = X^T\beta + \beta_0 = x_1\beta_1 + x_2\beta_2 + \beta_0$$

*SVM: Support Vector Machine

Classification
Machine à vecteurs de support linéaire (SVM)

https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

Classification à marge souple (fig 1)

- Équilibre entre un chemin "large" et un minimum d'empiètement (valeur se trouvant dans le chemin et/ou mauvais côté).
- Le paramètre C (paramètre de pénalité) permet de définir la largeur du chemin (si C petit alors le chemin est large avec plus d'empiètement et inversement)

Classification à marge rigide (fig 2)

- Toutes les observations doivent se trouver en dehors du chemin et du "bon" côté,
- Classification possible que si les données sont linéairement séparables,
- Sensible aux points aberrants.

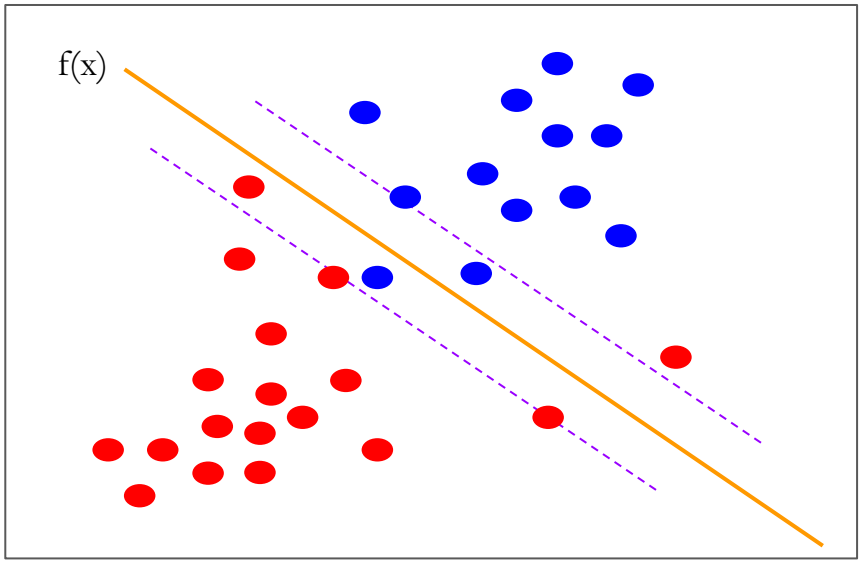


figure 1: marge souple

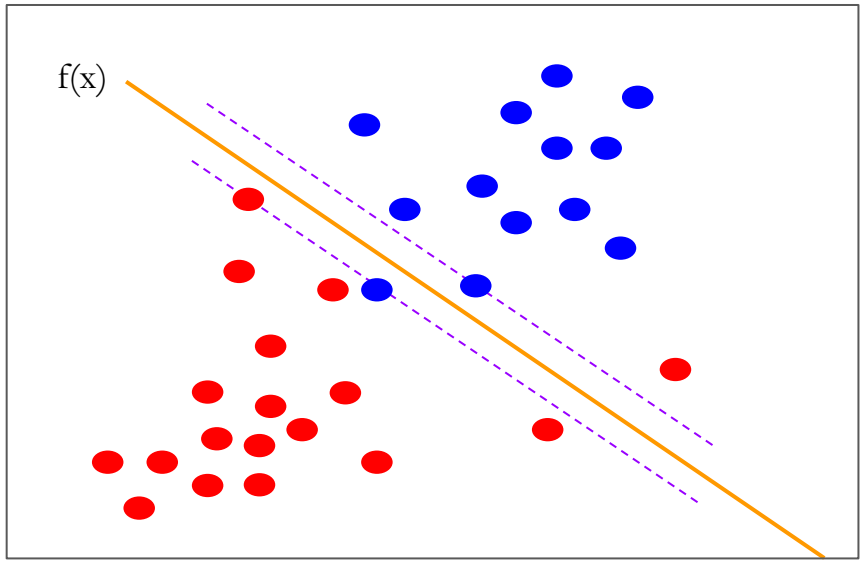
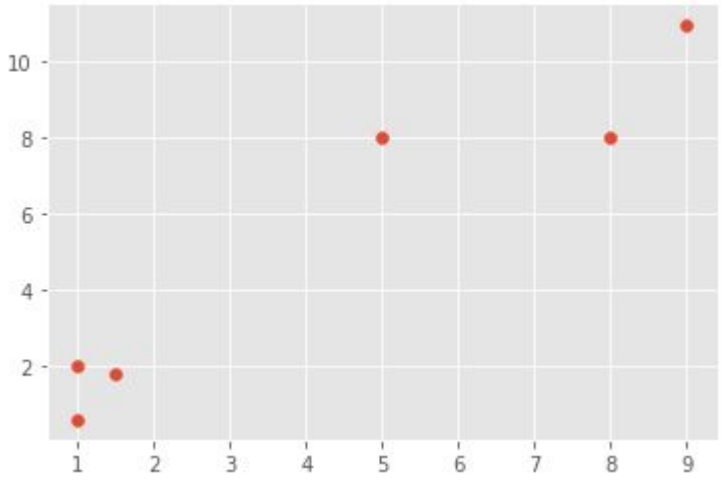
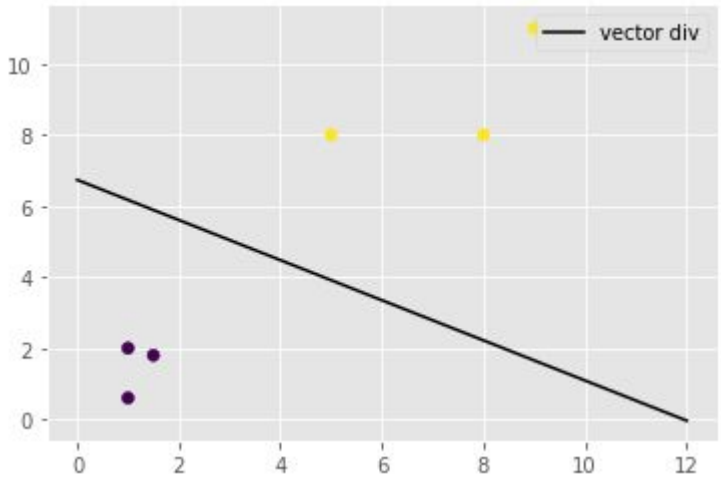


figure 2: marge rigide

https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

Exemple / Classification linéaire SVM





Utiliser SVM pour classifier deux classes avec :

```
X = np.array([[1,2],
              [5,8],
              [1.5,1.8],
              [8,8],
              [1,0.6],
              [9,11]])
```

Etiquettes : y = [0,1,0,1,0,1]

- Analyser la lib et le paramètre “C” de SVM.
- Choisir une classification linéaire et tracer le vecteur.
- Proposer une classification avec quatre classes et tirage aléatoire des classes.
- Comparer avec K-NN

Ressources:

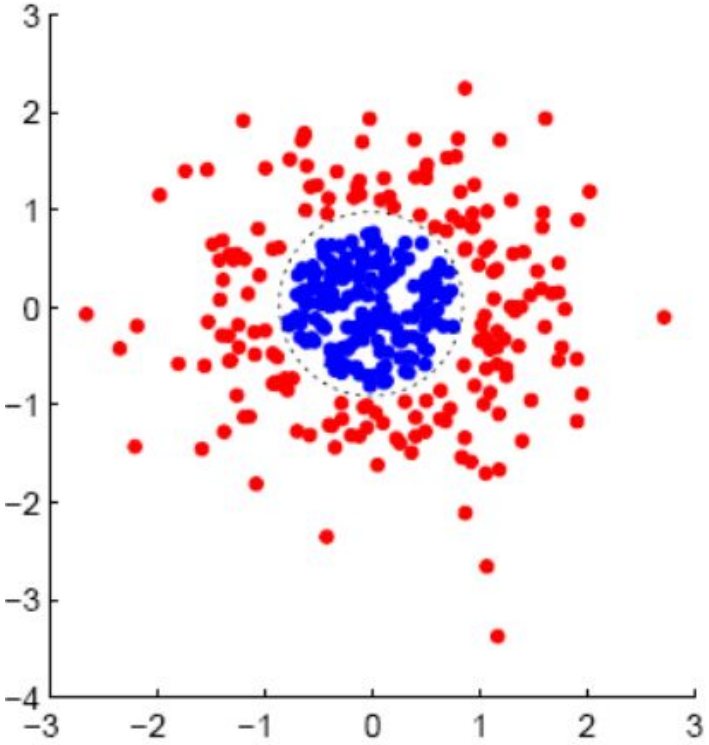
https://scikit-learn.org/stable/modules/svm.html

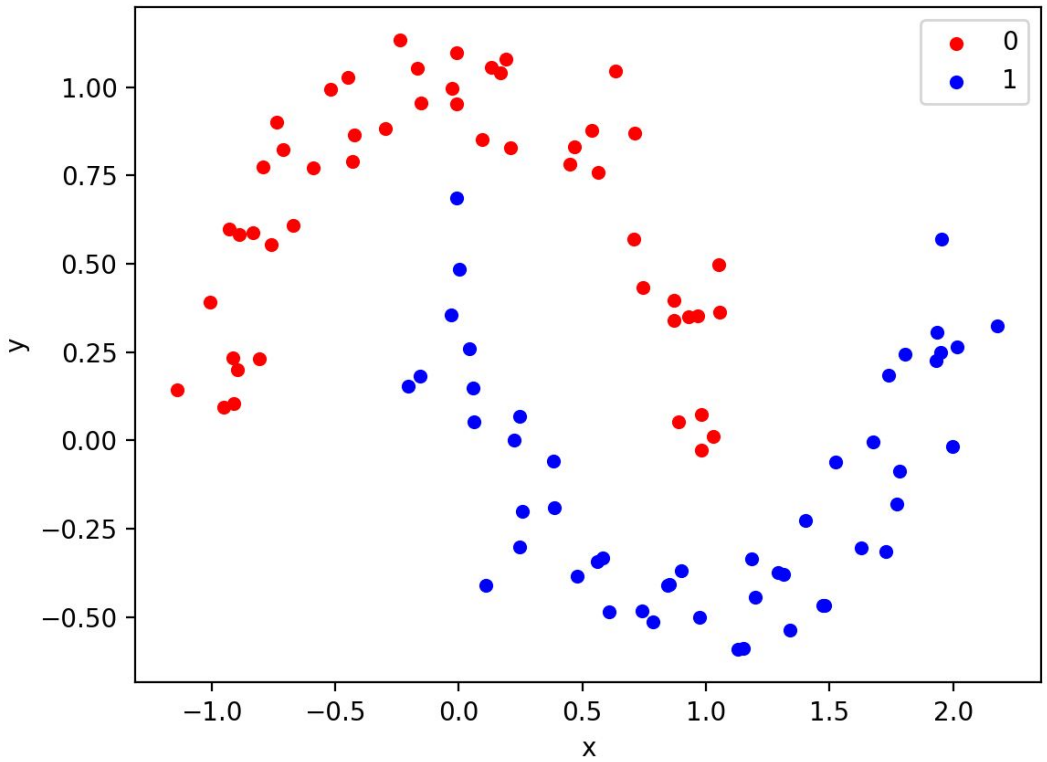
Classification
Machine à vecteurs de support non linéaire (SVM)

http://cs229.stanford.edu/notes/cs229-notes3.pdf
http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf

SVM non linéaire

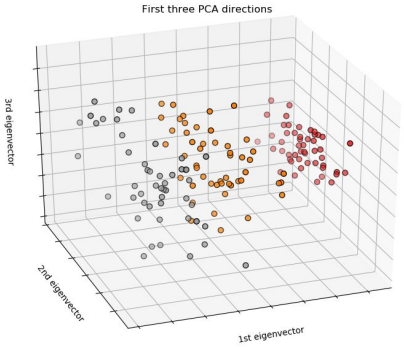
- Les jeux de données ne sont pas linéairement séparables
- Il n'existe pas d'hyperplan capable de séparer correctement les catégories
- Il est impossible de séparer linéairement les données dans cet espace vectoriel

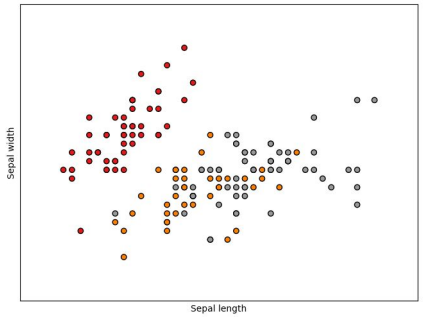


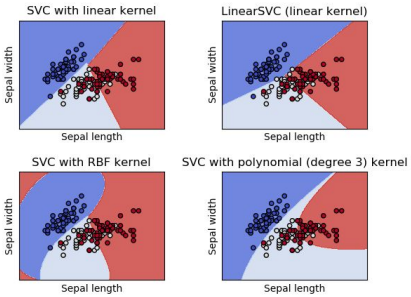


https://www.csie.ntu.edu.tw/~cjlin/libsvm/
https://scikit-learn.org/stable/modules/svm.html

Exemple / Classification non linéaire SVM







Utiliser SVM pour classifier le jeux de données Iris :

Le jeux de données Iris est utilisé comme base de référence dans de nombreux calculs afin de comparer les résultats des modèles de machine learning.

Travail à réaliser:

- Découvrez le jeux de données (afficher les données),
- Analyse PCA,
- Utiliser Scikit-Learn SVM pour analyser les données (utiliser les notebook disponibles).

Utiliser le notebook :
https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html#sphx-glr-auto-examples-svm-plot-iris-svc-py

Ressources:
https://en.wikipedia.org/wiki/Iris_flower_data_set
https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html
https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html?highlight=iris%20data%20set

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Régression linéaire équation normale
- Régression linéaire descente de gradient
- Régression linéaire descente de gradient stochastique (SGD)
- Descente de gradient mini-lot (mini-batch)
- Régression polynomiale
- Machine à vecteurs de support linéaire et non linéaire (SVM)
- **Réseau de neurone**

Apprentissage non supervisé:

- Arbre de décision
- K-NN
- K-MEANS

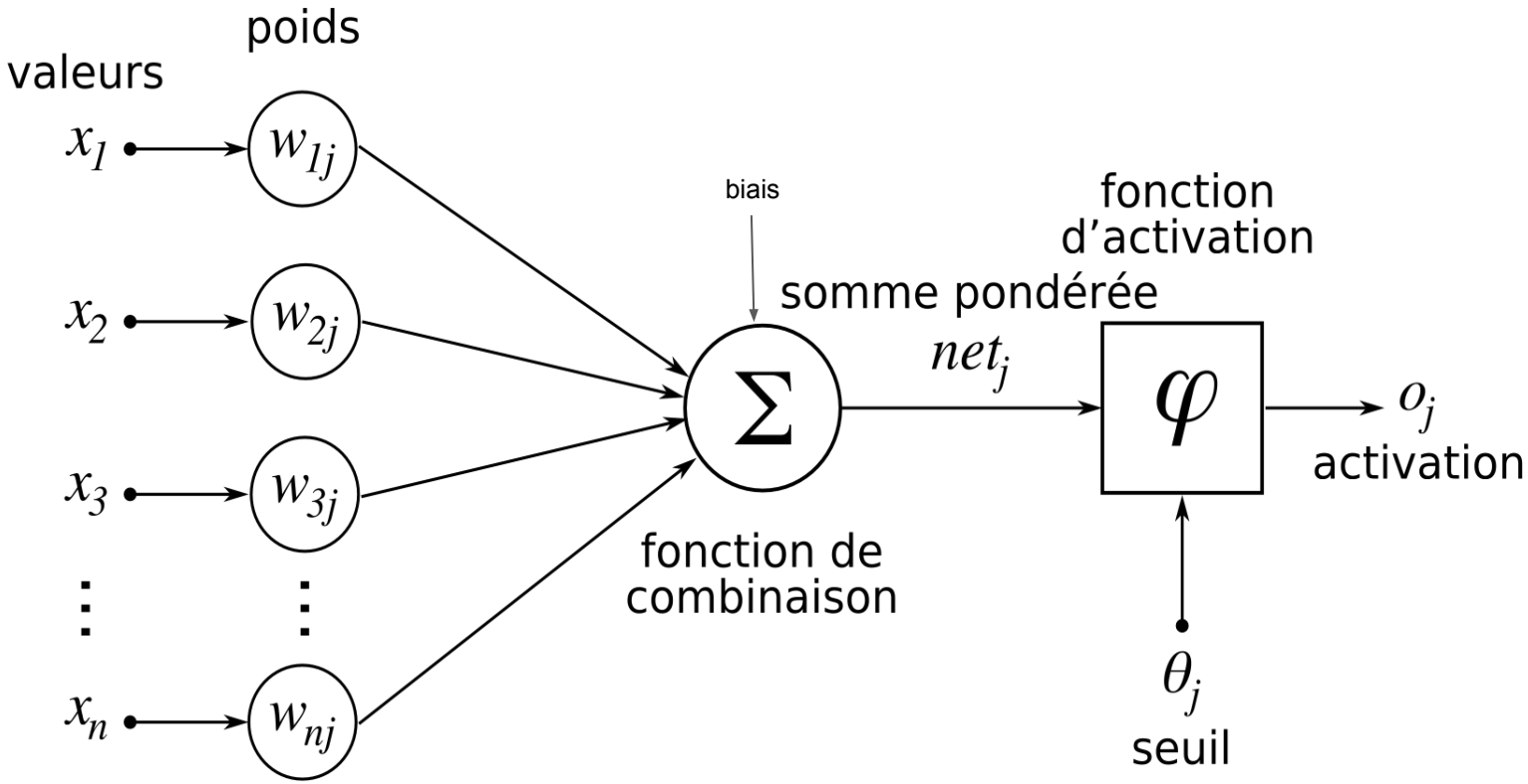
Analyse des séries chronologiques:

- Modèle ARIMA

Classification
Réseau de neurone

https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels

RAPPEL Perceptron:



Classification
Réseau de neurone

https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels

RAPPEL Classification:

$$f(x_1, x_2) = b + w_1 x_1 + w_2 x_2$$

$$f(X) = b + \sum_i w_i x_i$$

Dans le cas de la classification:

$$\text{classification} = \begin{cases} 1 & \text{if } f(X) > 0 \\ 0 & \text{if } f(X) \leq 0 \end{cases}$$

Classification
Réseau de neurone

https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels
https://fr.wikipedia.org/wiki/Fonction_d%27activation
https://fr.wikipedia.org/wiki/Fonction_logistique_(Verhulst)

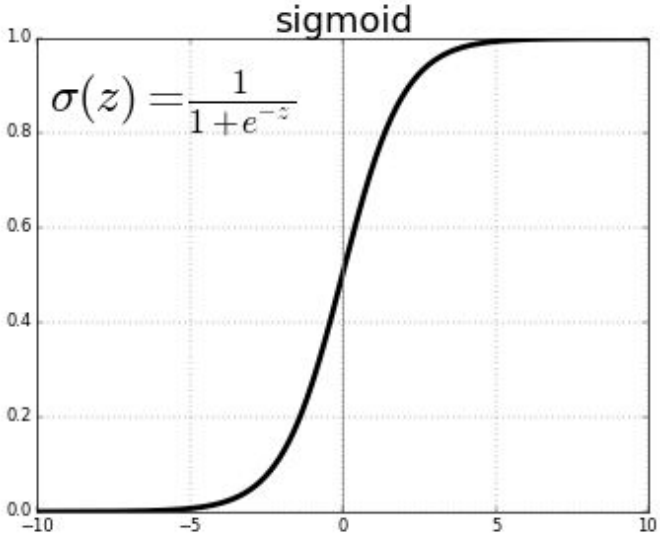
RAPPEL, fonction d'activation: $f(X) = b + \sum_i w_i x_i$

Fonction sigmoïde

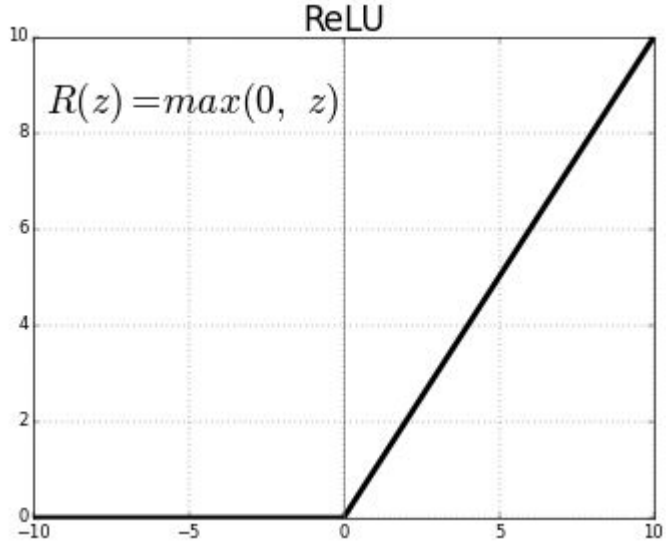
$$f(x) = \frac{1}{1+e^{-x}}$$

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

$$f'(x) = f(x)(1-f(x))$$



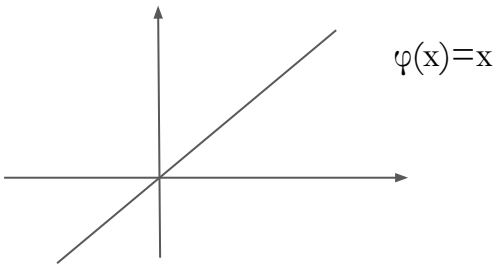
Fonction RELU

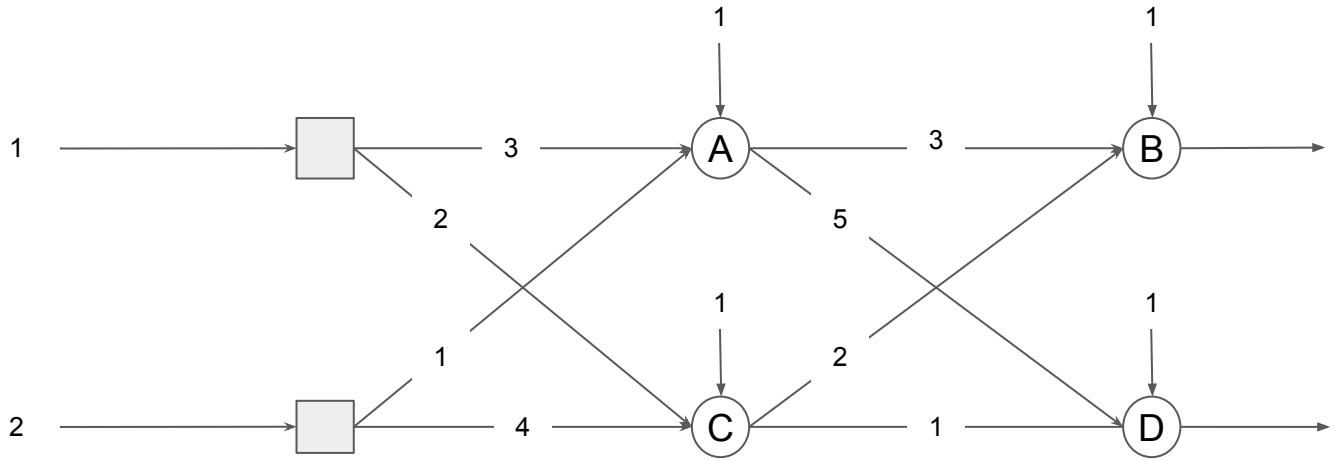


https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels

Propagation avant

Mise en oeuvre avec un réseau à une couche, φ(x) est une fonction d'activation linéaire :





Classification
Réseau de neurone

https://fr.wikipedia.org/wiki/R%C3%A9tropropagation_du_gradient

Apprentissage = ajuster automatiquement les poids des synapses pour retrouver les étiquettes.

Des données sont présentées à l'entrée du réseau de neurones:

- Algorithme de type « **online** » : apprentissage du réseau pour chaque entré,
- Algorithme de type **«batch»** : apprentissage du réseau pour toutes les entrées.

Fonction de coût:

$$C(W, B, S^r, E^r)$$

L'algorithme d'apprentissage de propagation de gradient est une expression de la dérivée partielle de la fonction de coût:

$$\partial C / \partial w$$

Classification
Réseau de neurone supervisé “online”

https://www.u

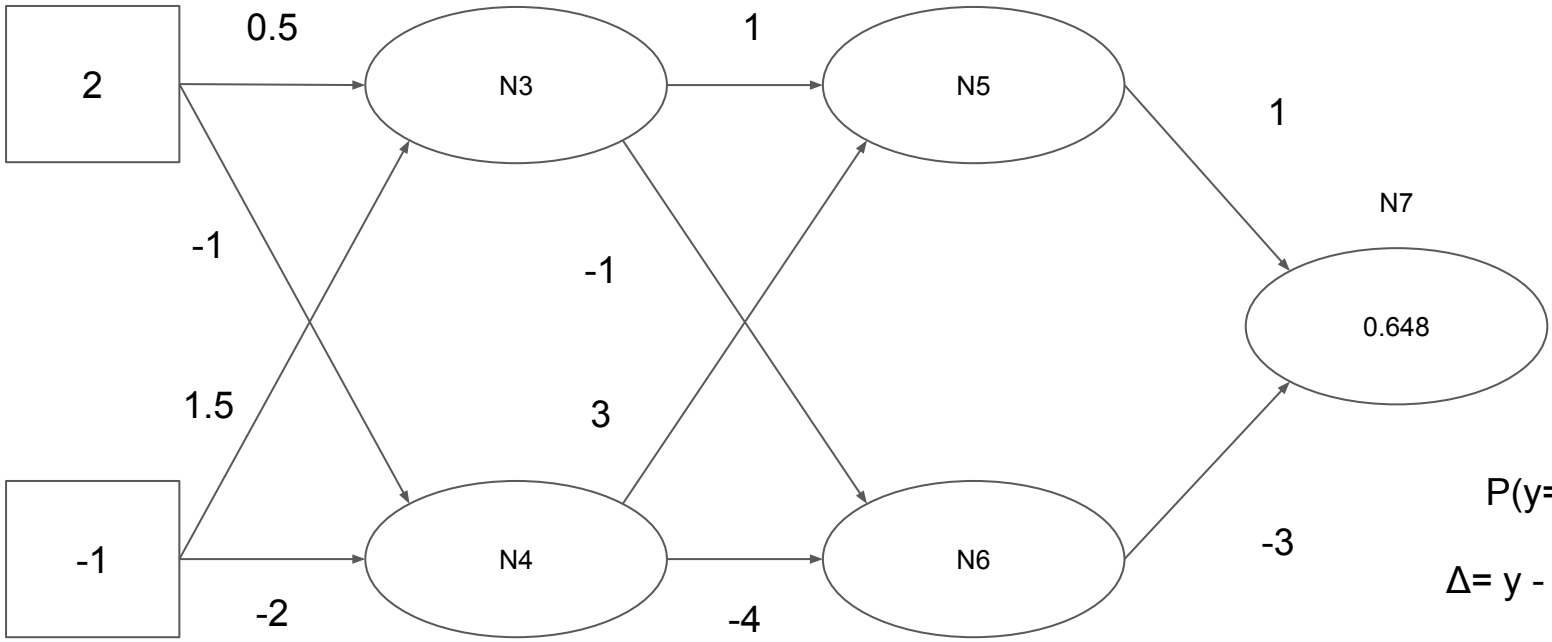
function BACK-PROP-LEARNING(*examples*, *network*) **returns** a neural network
inputs: *examples*, a set of examples, each with input vector **x** and output vector **y**
network, a multilayer network with L layers, weights $w_{i,j}$, activation function g
local variables: Δ, a vector of errors, indexed by network node

for each weight $w_{i,j}$ in *network* **do**
$w_{i,j} \leftarrow$ a small random number
repeat
for each example (**x**, **y**) **in** *examples* **do**
/* *Propagate the inputs forward to compute the outputs* */
for each node i in the input layer **do**
$a_i \leftarrow x_i$
for $\ell = 2$ **to** L **do**
for each node j in layer ℓ **do**
$in_j \leftarrow \sum_i w_{i,j}\, a_i$
$a_j \leftarrow g(in_j)$
/* *Propagate deltas backward from output layer to input layer* */
for each node j in the output layer **do**
$\Delta[j] \leftarrow y_j - a_j \quad (= -\partial Loss/\partial in_j)$
for $\ell = L - 1$ **to** 1 **do**
for each node i in layer ℓ **do**
$\Delta[i] \leftarrow g(in_i)(1 - g(in_i)) \sum_j w_{i,j}\Delta[j]$
/* *Update every weight in network using deltas* */
for each weight $w_{i,j}$ in *network* **do**
$w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$
until some stopping criterion is satisfied
return *network*

Classification
Réseau de neurone supervisé

https://www.usherbrooke.ca

x=[2, -1], y=1, utiliser une activation Logistique, taux apprentissage α=0.1



P(y=1 | X) = ?

Δ= y - a = 1 - 0.648

Propagation avant

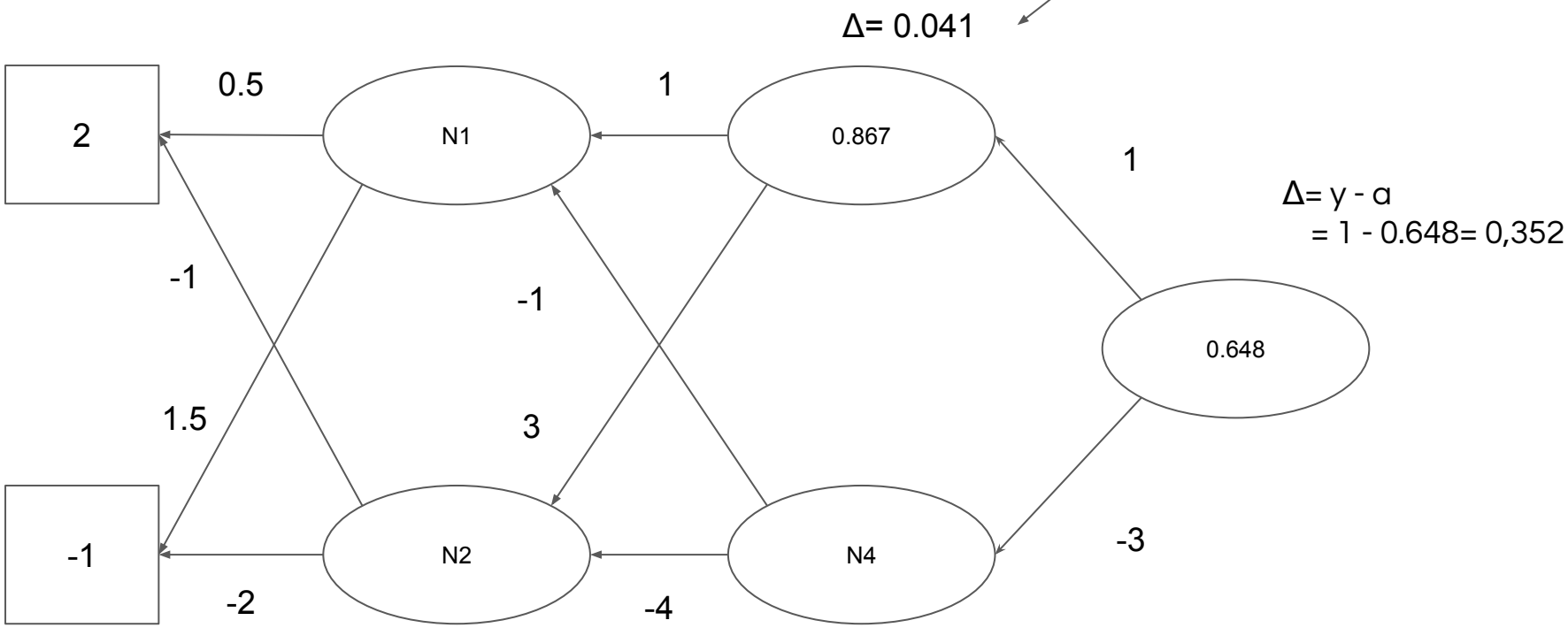
Classification
Réseau de neurone supervisé

https://www.usherbrooke.ca

x=[2, -1] Y=1

dérivée partielle *Logistique*

$$\Delta = 0.867 * (1-0.867) * 1 * 0.352 = 0.041$$



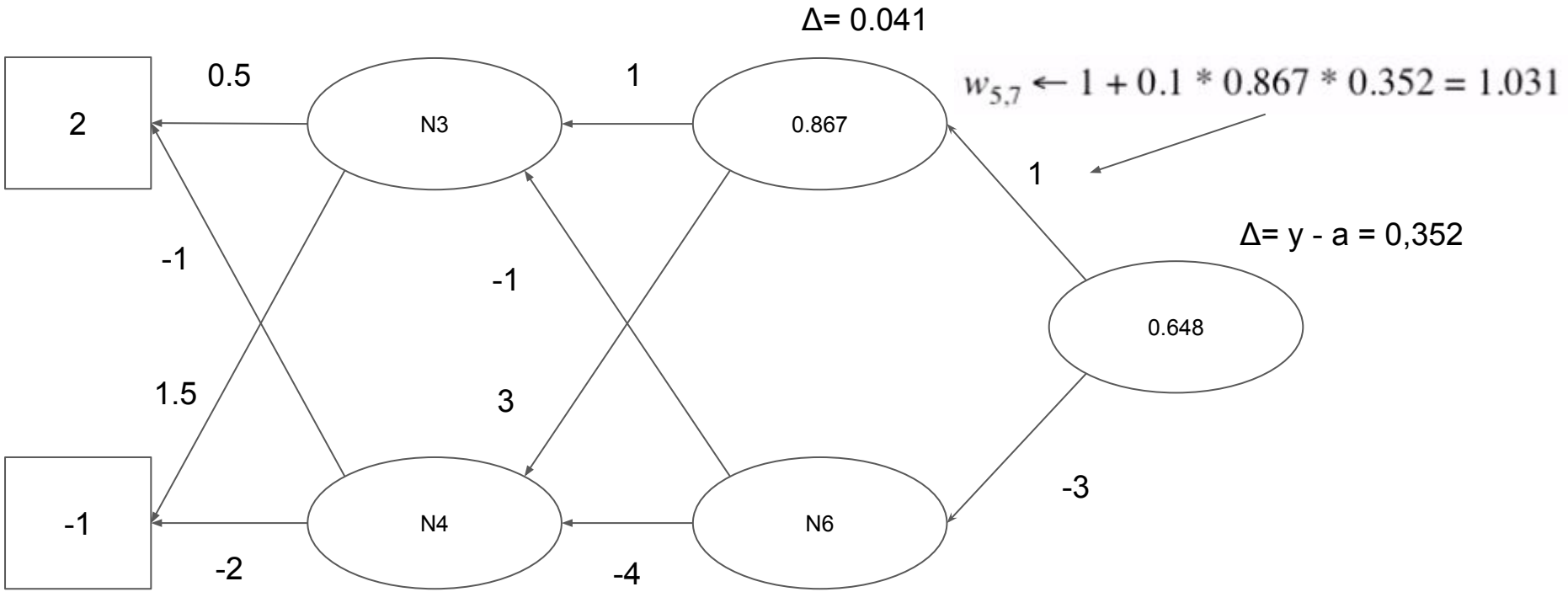
Propagation arrière

$$\Delta[j]=g(in_j)(1-g(in_j))\Sigma_k\ w_{j,k}\Delta[k]$$

Classification
Réseau de neurone supervisé

https://www.usherbrooke.ca

x=[2, -1], Y=1, utiliser une activation *Logistique*, taux apprentissage α=0.1



Mise à jour des points

$$w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$$

Classification

Réseau de neurone supervisé

Exemple / Classification d'images

https://scikit-learn.org/stable/datasets/index.html
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report

digits = load_digits()
n_samples = len(digits.images)
print("Number_of-examples = ", n_samples)

import matplotlib.pyplot as plt
print("\n Plot of first example")
plt.gray()
plt.matshow(digits.images[0])
print("CLOSE PLOT WINDOW TO CONTINUE")
plt.ioff()
plt.show()

# Flatten the images, to turn data in a (samples, feature) matrix:
data = digits.images.reshape((n_samples, -1))

X = data
y = digits.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)

clf = MLPClassifier(hidden_layer_sizes=(10, ), activation='tanh', solver='sgd',
                    alpha=0.00001, batch_size=4, learning_rate='constant', learning_rate_init=0.01,
                    power_t=0.5, max_iter=9, shuffle=True, random_state=11, tol=0.00001,
                    verbose=True, warm_start=False, momentum=0.8, nesterovs_momentum=True,
                    early_stopping=False, validation_fraction=0.1,
                    beta_1=0.9, beta_2=0.999, epsilon=1e-08)
print(clf)

# Train the MLP classifier on training dataset
clf.fit(X_train, y_train)

# Evaluate acuracy on test data
score = clf.score(X_test,y_test)
print("Acuracy (on test set) = ", score)
y_true, y_pred = y_test, clf.predict(X_test)
print(classification_report(y_true, y_pred))
```

Code:

https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html#sphx-glr-auto-examples-classification-plot-digits-classification-py

Classification
Réseau de neurone supervisé

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
https://docs.scipy.org/doc/numpy-1.9.2/reference/routines.random.html

Exemple

figure 1

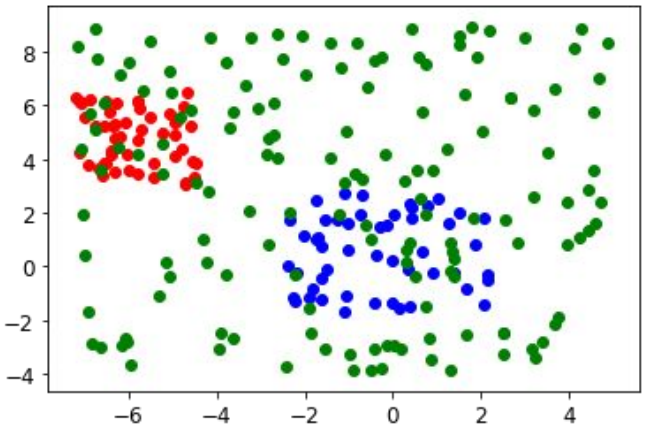
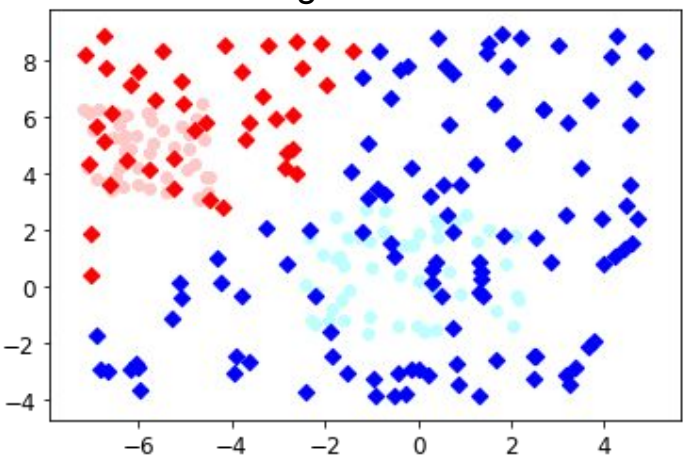


figure 2



Classificateur:

- Définir deux classes C1 et C2 avec un tirage aléatoire uniforme (par exemple points rouges et bleus - figure 1):
  - soit deux listes qui contiennent C1 et C2
  - afficher les points C1 et C2
  - soit une liste qui contient les étiquettes de sortie du réseau
- Utiliser MLPClassifier avec:
  - 3 niveaux de 20 neurones
  - un solveur a descente de gradient (sgd)
  - une activation RELU puis une activation Logistique
- Réaliser une estimation à partir d'un jeux de test (fig2 - point vert)
- Afficher le résultat de l'estimation (figure 2)

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Régression linéaire équation normale
- Régression linéaire descente de gradient
- Régression linéaire descente de gradient stochastique (SGD)
- Descente de gradient mini-lot (mini-batch)
- Régression polynomiale
- Machine à vecteurs de support linéaire et non linéaire (SVM)
- Réseau de neurone

Apprentissage non supervisé:

- **K-NN, K-MEANS**
- Arbre de décision

Analyse des séries chronologiques:

- Modèle ARIMA

https://fr.wikipedia.org/wiki/Recherche_des_plus_proches_voisins

K Nearst Neighbors (K-NN)

- K-NN (K-nearest neighbors) est une méthode d'apprentissage supervisé,
- Principe : "dis moi qui sont tes voisins, je te dirais qui tu es…",
- K-NN ne va pas calculer un modèle prédictif à partir d'un Training Set comme c'est le cas pour la régression linéaire,
- K-NN va chercher les K instances du jeu de données les plus proches des observations,

Si K-NN est utilisé pour la régression, c'est la moyenne (ou la médiane) des variables y des K plus proches observations qui servira pour la prédiction.

Si K-NN est utilisé pour la classification, c'est le mode des variables y des K plus proches observations qui servira pour la prédiction.

Classification
K-NN - Algorithmique

https://fr.wikipedia.org/wiki/Recherche_des_plus_proches_voisins

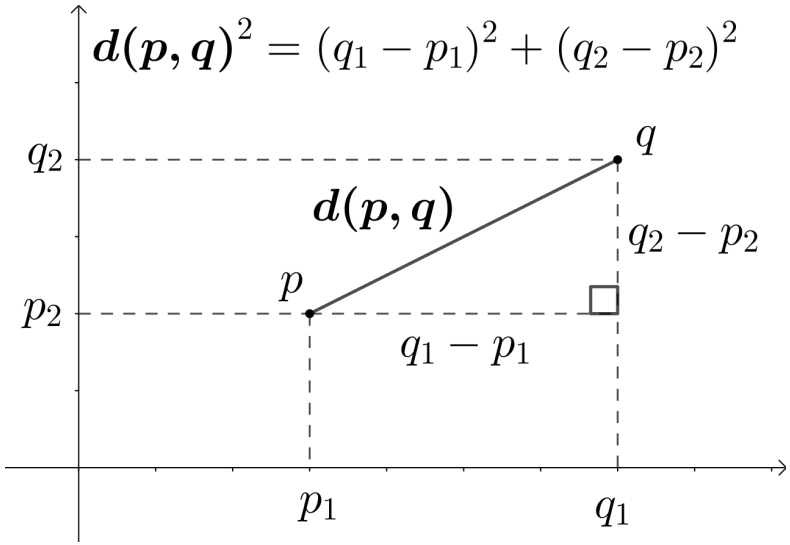
- **K-NN** a besoin d'une fonction de calcul de distance entre deux observations.
- Plus deux points sont proches l'un de l'autre, plus ils sont similaires et vice versa.
- Il existe plusieurs fonctions de calcul de distance, notamment, la distance euclidienne, la distance de Manhattan, la distance de Minkowski, celle de Jaccard, la distance de Hamming…
- Pour les données quantitatives (exemple : poids, salaires, taille, montant de panier électronique etc…) et du même type, utiliser la distance euclidienne

https://en.wikipedia.org/wiki/Euclidean_distance
https://fr.wikipedia.org/wiki/Distance_(math%C3%A9matiques)

La distance Euclidienne:

Sur un espace isotropique (les distances mesurées sont les mêmes dans toutes les directions), la distance euclidienne est la plus utilisée. La distance entre deux points se calcule grâce au théorème de Pythagore:

Avec $p = (p1, p2)$ *and* $q = (q1, q2)$



2 dimensions: $$d(\mathbf{p},\mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

3 dimensions: $$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}.$$

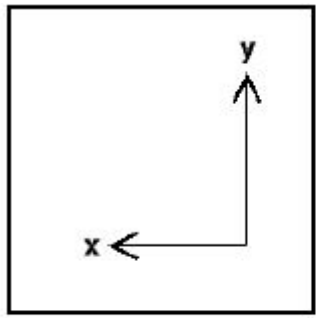
n dimensions: $$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

https://fr.wikipedia.org/wiki/Distance_(math%C3%A9matiques)
https://fr.wikipedia.org/wiki/Distance_de_Manhattan

La distance Manhattan:

La distance de Manhattan: calcule la somme des valeurs absolues des différences entre les coordonnées de deux points :

$$d(A,B) = |X_B - X_A| + |Y_B - Y_A|$$



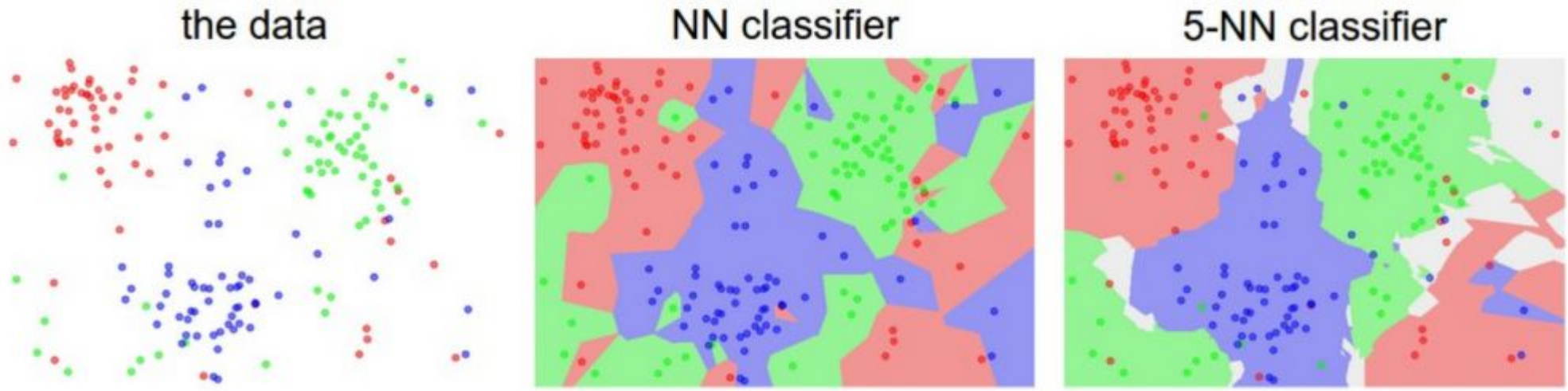
Manhattan

https://fr.wikipedia.org/wiki/Recherche_des_plus_proches_voisins

Comment choisir la valeur K ?

- Le choix de la valeur K à utiliser pour effectuer une prédiction avec K-NN, varie en fonction du jeu de données.
- En règle générale, moins on utilisera de voisins (un nombre K petit) plus on sera sujette au sous apprentissage (**underfitting**).
- Si on utilise K nombre de voisins avec K=N et N étant le nombre d'observations, on risque d'avoir du **overfitting** et par conséquent un modèle qui se généralise mal sur des observations qu'il n'a pas encore vu.

Classification
K-NN - Algorithmique



- A gauche représente des points dans un plan 2D avec trois types d' étiquetages possibles (rouge, vert, bleu).
- Pour le 5-NN classifieur, les limites entre chaque région sont assez lisses et régulières.
- N-NN Classifier, on remarque que les limites sont "chaotiques" et irrégulières. Cette dernière provient du fait que l'algorithme tente de faire rentrer tous les points bleus dans les régions bleues, les rouges avec les rouges etc… c'est un cas d'overfitting.

⇒ On préférera le 5-NN classifier sur le NN-Classifier.

https://fr.wikipedia.org/wiki/Recherche_des_plus_proches_voisins

Conclusion

- K-NN stocke tout le jeu de données pour effectuer une prédiction,
- K-NN ne calcule aucun modèle prédictif et il rentre dans le cadre du “Lazy Learning”,
- K-NN effectue des prédictions juste à temps (à la volée) en calculant la similarité entre un observation en entrée et les différentes observations du jeu de données,

Classification
K-NN - Algorithme

https://fr.wikipedia.org/wiki/Recherche_des_plus_proches_voisins

Données en entrée :

- Un ensemble de données D .
- Une fonction de définition distance d.
- Un nombre entier K

Pour une nouvelle observation X dont on veut prédire sa variable de sortie y:

- Calculer toutes les distances de cette observation X avec les autres observations du jeu de données D
- Retenir les K observations du jeu de données D les proches de X en utilisation le fonction de calcul de distance d:
  - Prendre les valeurs de y des K observations retenues :
  - Si on effectue une régression, calculer la moyenne (ou la médiane) de y retenues
  - Si on effectue une classification , calculer le mode de y retenues
  - Retourner la valeur calculée dans l'étape 3 comme étant la valeur qui a été prédite par K-NN pour l'observation X.

Classification
K-MEANS

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

- Le problème est de diviser un ensemble de points en k groupes appelés clusters,
- On considère la distance d'un point à la moyenne des points de son cluster,
- Les k-moyennes sont notamment utilisées en apprentissage non supervisé où l'on divise des observations en k partitions.
- K représente le nombre de cluster.

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Equation normale
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- K-NN, K-MEANS
- **Arbre de décision**

Analyse des séries chronologiques:

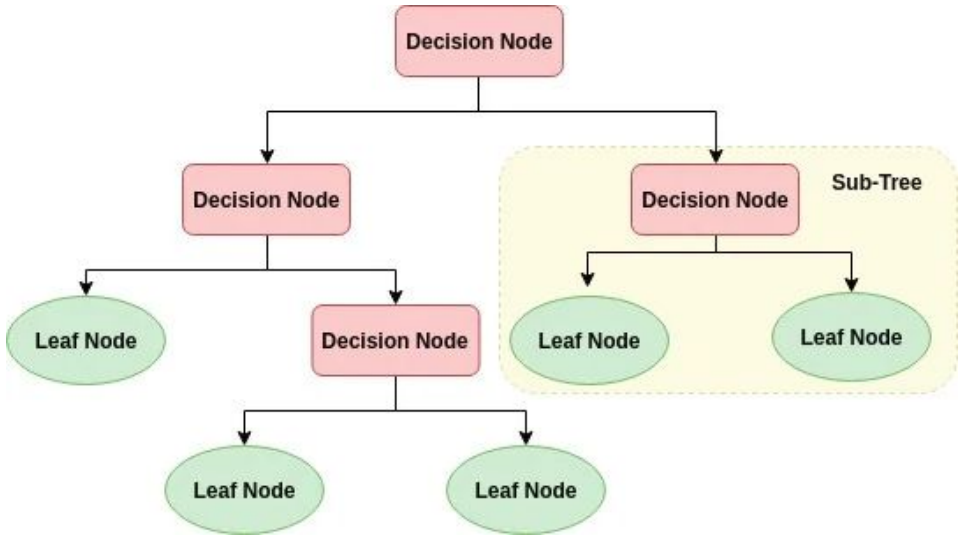
- Modèle ARIMA

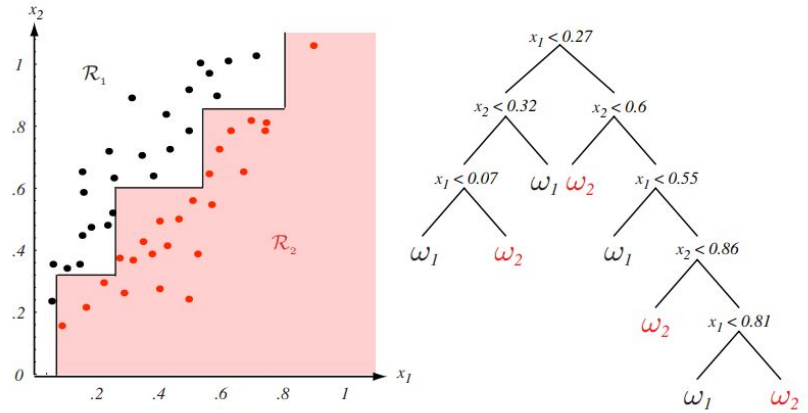
Arbre de décision

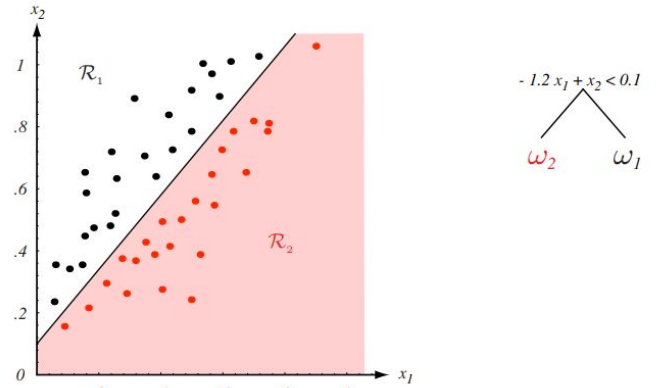
https://scikit-learn.org/stable/modules/tree.html
https://en.wikipedia.org/wiki/Decision_tree_learning
http://cedric.cnam.fr/vertigo/Cours/ml2/tpArbresDecision.html
http://cedric.cnam.fr/vertigo/Cours/ml2/coursArbresDecision.html

Principe

- Les arbres de décision sont des algorithmes d'apprentissage automatique,
- Permet de résoudre des problèmes de classification, de régression,
- Représentation sous forme d'une arbres à plusieurs branches.







fcamps@laas.fr

Arbre de décision

https://scikit-learn.org/stable/modules/tree.html
https://en.wikipedia.org/wiki/Decision_tree_learning
http://cedric.cnam.fr/vertigo/Cours/ml2/tpArbresDecision.html

nœuds: chaque nœud correspond à une question sur un attribut et à un ensemble d'exemples

branches: chaque branche part d'un nœud et correspond à une réponse possible à la question posée en ce nœud

- Algorithme CART (Scikit-Learn) [Breiman et al., 1984] : 2 branches par nœud
- Algorithme ID3 [Quinlan, 1986], C4.5 [Quinlan, 1993] : autant de branches que de valeurs possibles pour l'attribut étudié

feuilles: nœuds d'où ne part aucune branche ; correspond à une classe

Arbre de décision

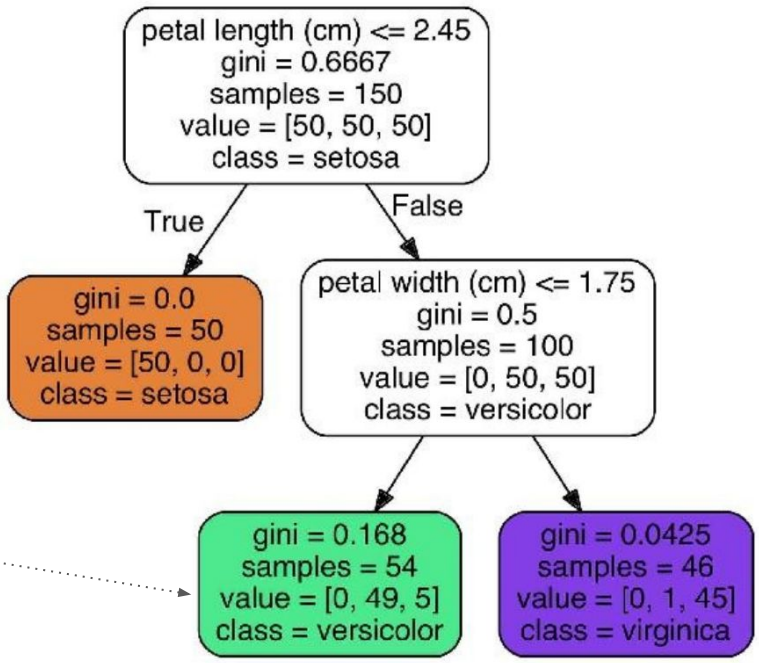
https://scikit-learn.org/stable/modules/tree.html
https://en.wikipedia.org/wiki/Decision_tree_learning
http://cedric.cnam.fr/vertigo/Cours/ml2/tpArbresDecision.html

Impureté du noeud Gini (voir aussi Entropie)

Un nœud possède une impureté. Un nœud pur à un GINI=0, c'est-à-dire que toutes les données d'entraînement qui y aboutissent appartiennent à la même classe.

$$G_i = 1 - \sum p_{ik}^2$$

p_{ik} = Pourcentage d'observation de la classe k parmis toutes les obs

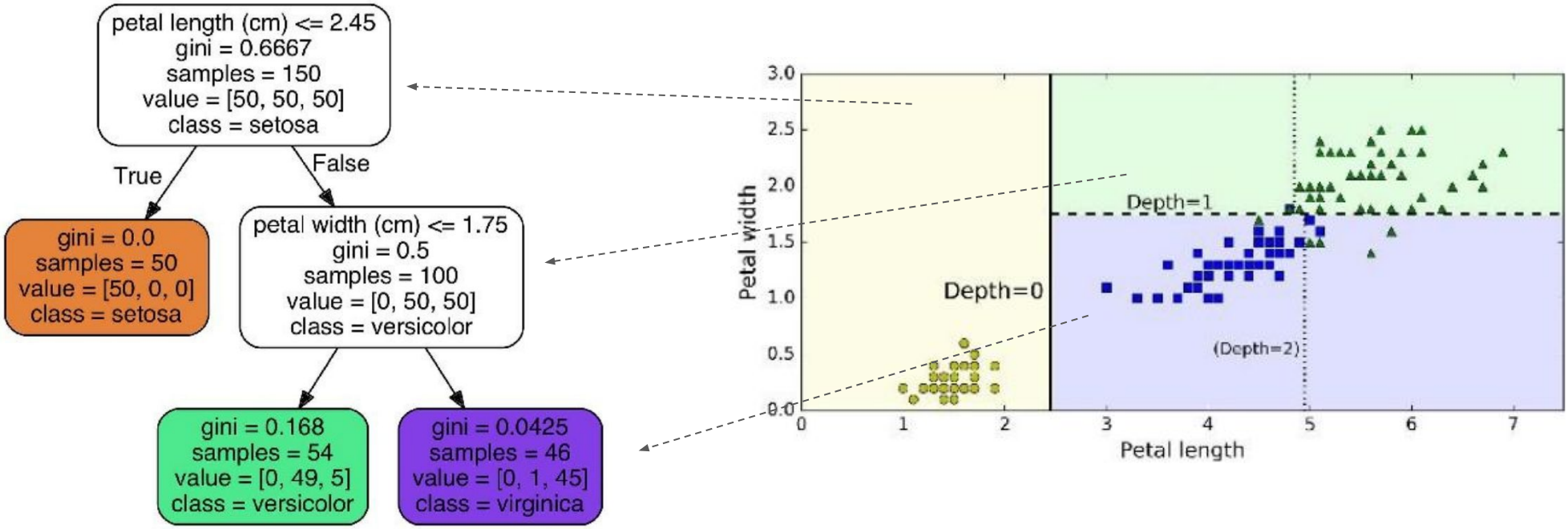


GINI pour la branche de classe versicolor

GINI = 1 - $(0/54)^2$ - $(49/54)^2$ - $(5/54)^2$
= 0.168

Arbre de décision

https://scikit-learn.org/stable/modules/tree.html
https://en.wikipedia.org/wiki/Decision_tree_learning
http://cedric.cnam.fr/vertigo/Cours/ml2/tpArbresDecision.html



https://scikit-learn.org/stable/modules/tree.html
https://en.wikipedia.org/wiki/Decision_tree_learning
http://cedric.cnam.fr/vertigo/Cours/ml2/tpArbresDecision.html

Avantages

- ➔ Les arbres de décision sont faciles à interpréter et à visualiser.
- ➔ Permet de capturer des motifs non linéaires.
- ➔ Nécessite moins de prétraitement des données de l'utilisateur, par exemple, il n'est pas nécessaire de normaliser les colonnes.
- ➔ Peut être utilisé pour la prévision de valeurs manquantes.

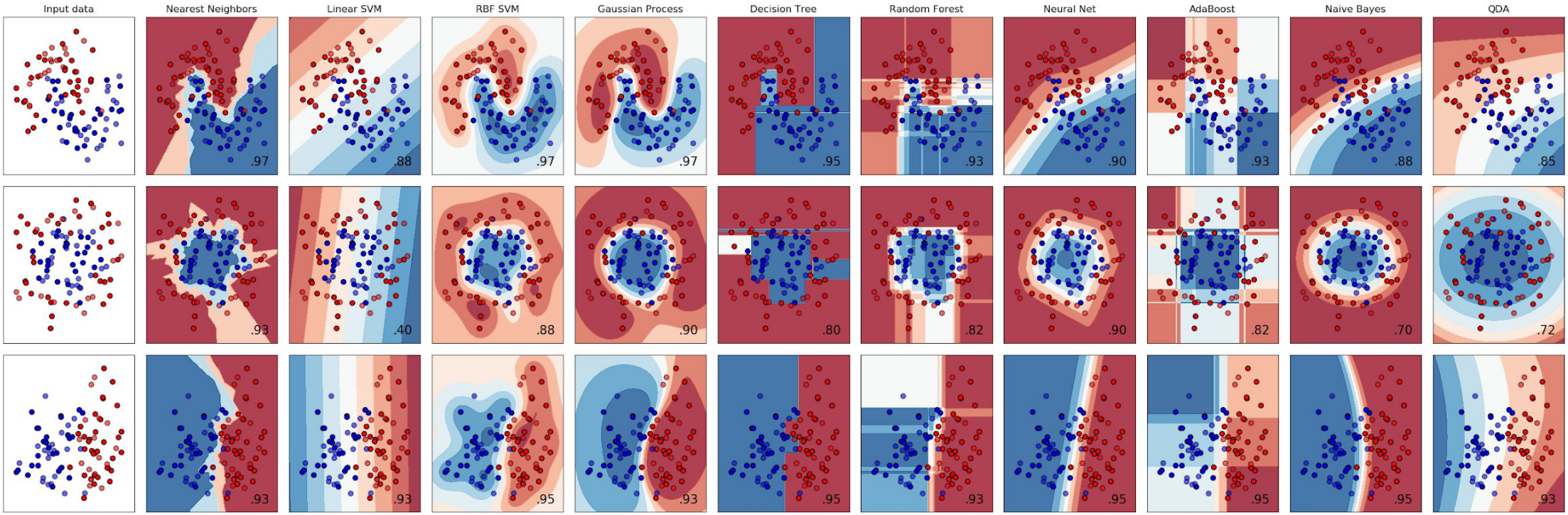
https://scikit-learn.org/stable/modules/tree.html
https://en.wikipedia.org/wiki/Decision_tree_learning
http://cedric.cnam.fr/vertigo/Cours/ml2/tpArbresDecision.html

Inconvénients

- ➔ Sensible aux données bruyantes. Il peut surcharger les données bruyantes.
- ➔ Une petite variation (ou variance) des données peut entraîner un arbre de décision différent. Cela peut être réduit en sachant et en renforçant les algorithmes.
- ➔ Les arbres de décision sont biaisés avec un ensemble de données déséquilibrées, il est donc recommandé d'équilibrer l'ensemble de données avant de créer l'arbre de décision.

https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Scikit-learn - Classification comparatif



Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Equation normale
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

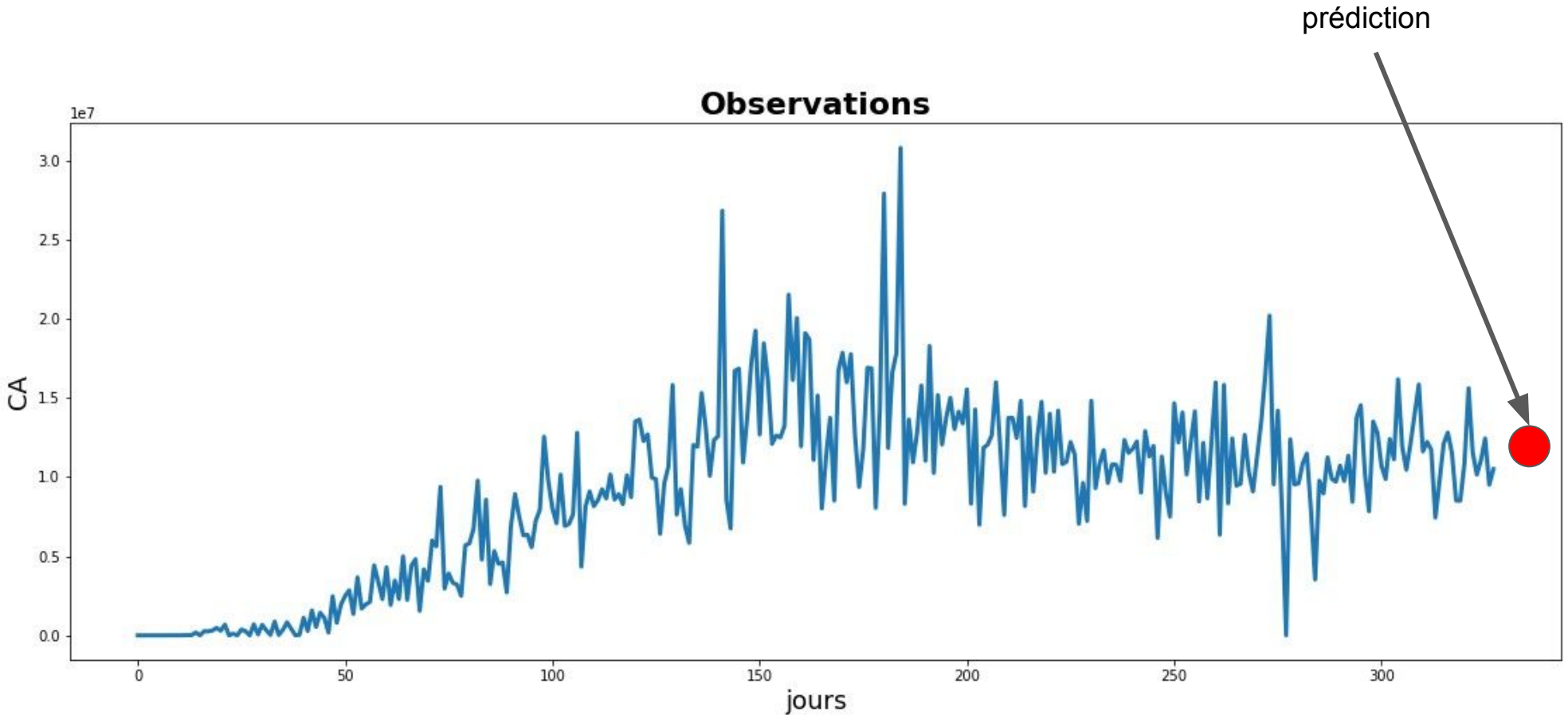
- K-NN, K-MEANS
- Arbre de décision

Analyse des séries chronologiques:

- **Modèle ARIMA**

Modèle ARIMA (AutoRegressive Integrated Moving Average)

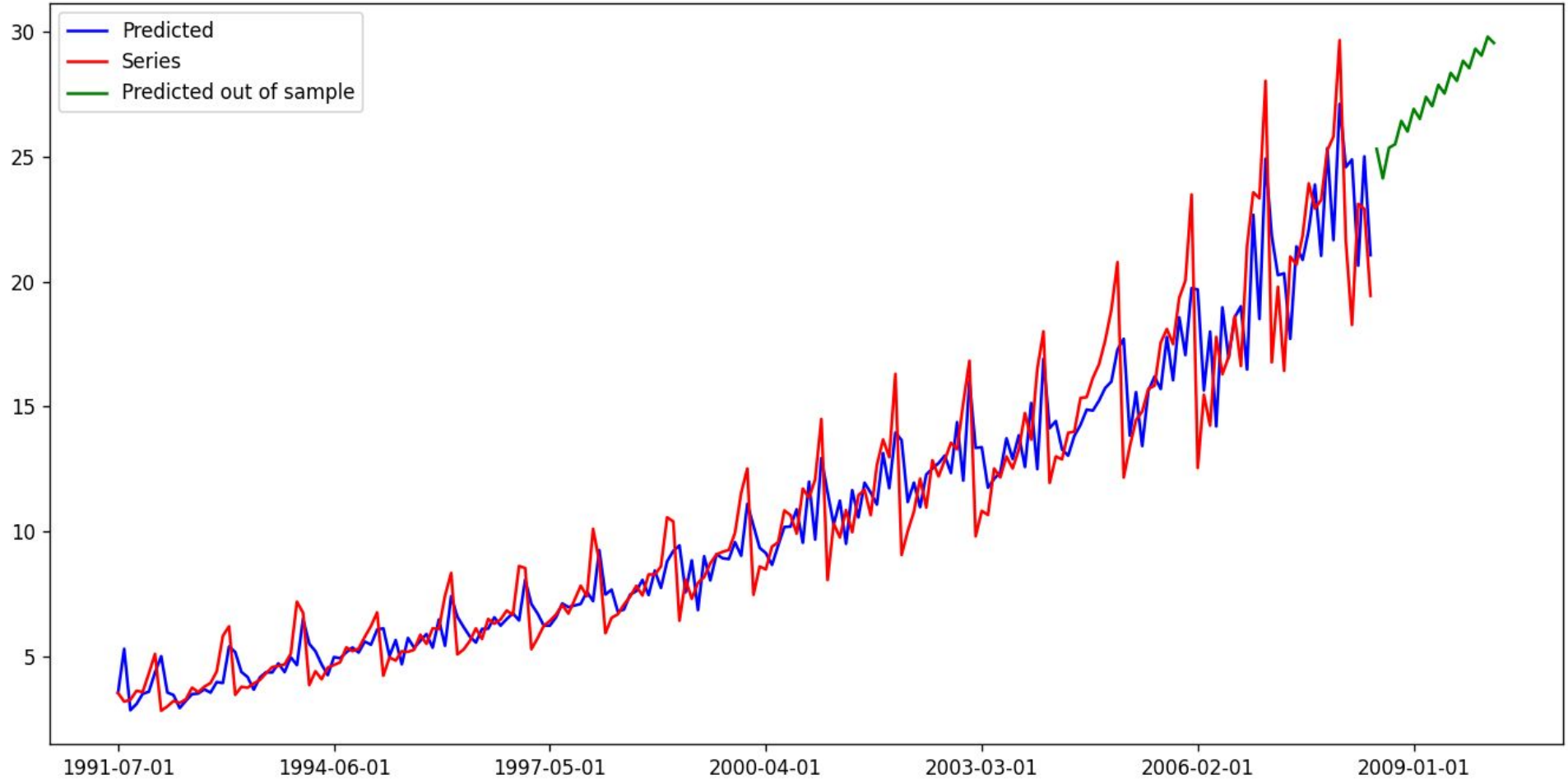
- Un modèle de prévision de séries chronologiques:



https://fr.wikipedia.org/wiki/Mod%C3%A9lisation_des_donn%C3%A9es

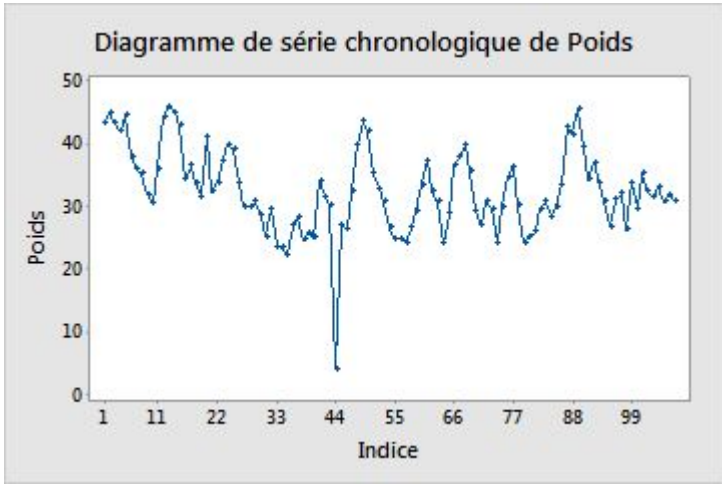
Modèle ARIMA / Exemple

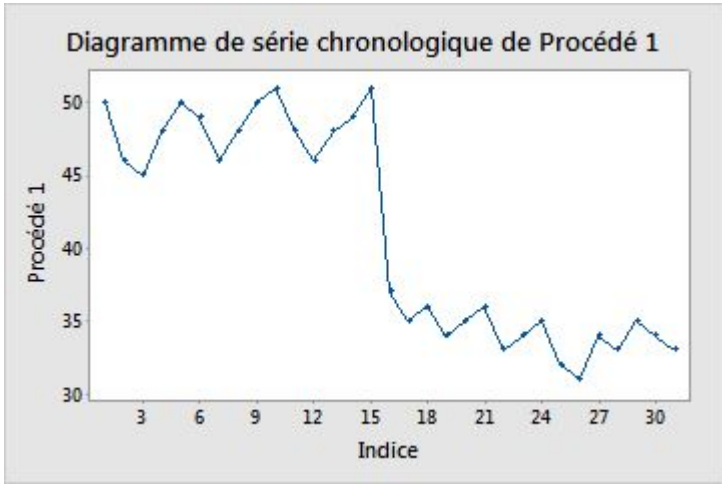
- Analyse de la prédiction



Modèle ARIMA (AutoRegressive Integrated Moving Average)

- Exemple d'utilisation:
  - prédire la météo des sept prochains jours,
  - la performance des stocks pour les prochains jours.
  - identifier les fluctuations des ventes,
  - prévenir des cas de pannes,
  - l'évolution d'une variable dans un système …





Modèle ARIMA (AutoRegressive Integrated Moving Average)

- Modèle simple linéaire,
- Nécessite le traitement de l'ensembles données,
- Très bons résultats,
- Solution analytique stable,
- Solution univariée et multivariée
- Divergence sur les prédictions si le modèle n'est pas mis à jour sur les nouvelles séries.

Modèle ARIMA (AutoRegressive Integrated Moving Average)

- L'une des méthodes de prévision de séries temporelles les plus répandues,
- Un modèle qui prédit les valeurs futures d'une série temporelle sur certains aspects de la structure statistique de la série observée.
- **ARIMA(p,d,q) :**
  - **p**: le nombre de termes auto-régressifs (combinaison linéaire),
  - **d**: le nombre de différences ⇒ (yt - y t-1) -(yt-1-yt-2) ….
  - **q** : le nombre de moyennes mobiles.

Modèle ARIMA / Paramètre d

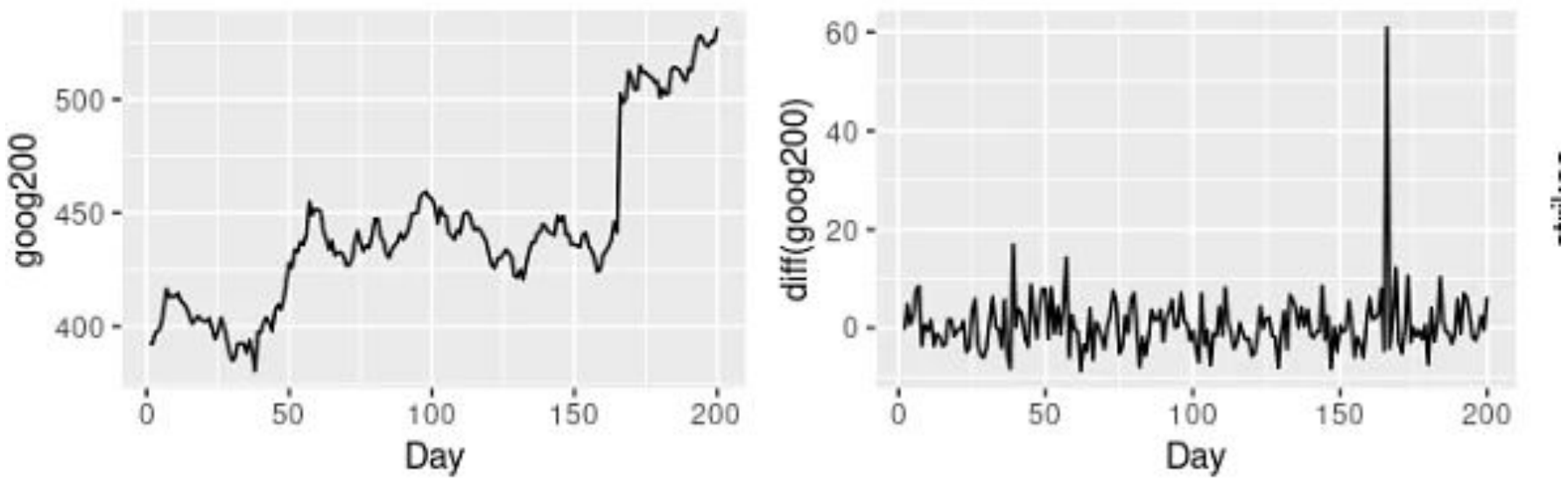
- La différenciation est une méthode pour éliminer une tendance d'une série temporelle,

- La différence d'ordre d=1: $y'_t = y_t - y_{t-1}$

- La différence d'ordre d=2: $y''_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$

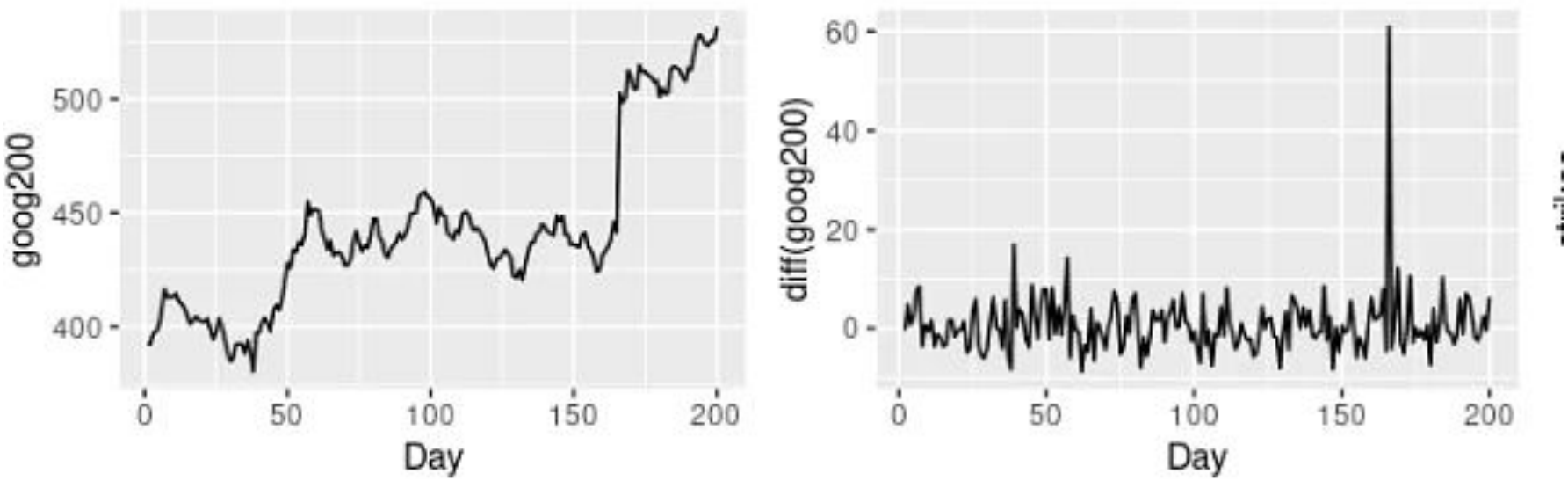
Modèle ARIMA / Paramètre d

- Pour supprimer cette corrélation, ARIMA utilise la différenciation pour rendre les données stationnaires.
- La différenciation, dans sa forme la plus simple, consiste à prendre la différence de deux points de données adjacents:
- **Exemple**: le graphique de gauche ci-dessus montre le cours de l'action de Google pendant 200 jours. Alors que le graphique de droite est la version différenciée du premier graphique.



Modèle ARIMA / Paramètre d

- Le graphique de droite montre l'évolution du stock de Google sur 200 jours.
- Le graphique de gauche: les tendances sont un signe de données de séries chronologiques non stationnaires, tandis qu'aucune tendance ou saisonnalité, ou variance croissante n'est observée dans la deuxième figure, la version différenciée (graphique de droite) est stationnaire.



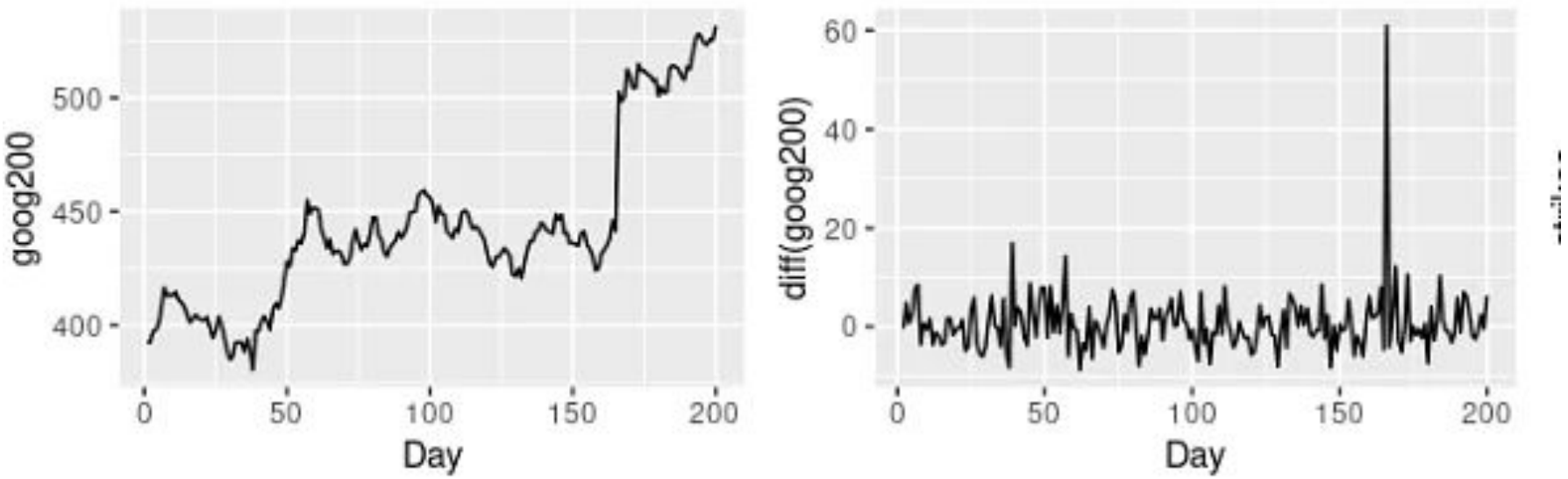
Modèle ARIMA / Paramètre d

- Le graphique de droite peut simplement être modélisé par:

$$y'_t = y_t - y_{t-1}$$

$$y'_t = y_t - y_{t-1} = y_t - By_t = (1 - B)y_t$$

Où B désigne l'opérateur de décalage défini comme $By_t = y_{t-1}$



Modèle ARIMA / Paramètre d

- Cependant, la différenciation pour créer des données stationnaires n'est pas toujours aussi simple,
- Plusieurs itérations de différenciation peuvent aider davantage dans une certaine mesure si nécessaire.
- La différenciation des données d fois créée des données différenciées d'ordre d.
  - si d= 2

$$\begin{aligned} y_t'' &= y_t' - y_{t-1}' \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2}. \end{aligned}$$

Ou

$$y_t'' = y_t - 2y_{t-1} + y_{t-2} = (1 - 2B + B^2)y_t = (1 - B)^2 y_t \,.$$

- **D'un façon générale pour un ordre d:** $(1 - B)^d y_t$

https://www.real-statistics.com/time-series-analysis/autoregressive-processes/augmented-dickey-fuller-test/

Modèle ARIMA / Différenciation d ⇒ Test Dickey Fuller

- Test Dickey Fuller est plus précis que l'observation du graphique d'autocorrélation ACF (Autocorrelation function),
- Le Dickey-Fuller augmenté ou ADF, vise à rejeter l'hypothèse nulle selon laquelle les données de séries chronologiques données ne sont pas stationnaires.
- ADF calcule une valeur p et la compare à une valeur seuil de 0,05. Si la valeur p est inférieure à ce niveau, les données sont stationnaires ; sinon, l'ordre de différenciation est incrémenté de un.

```
1 from statsmodels.tsa.stattools import adfuller
2 result = adfuller(df.value.dropna())
3 print('p-value: ', result[1])
4
5 result = adfuller(df.value.diff().dropna())
6 print('p-value: ', result[1])
7
8 result = adfuller(df.value.diff().diff().dropna())
9 print('p-value: ', result[1])
```

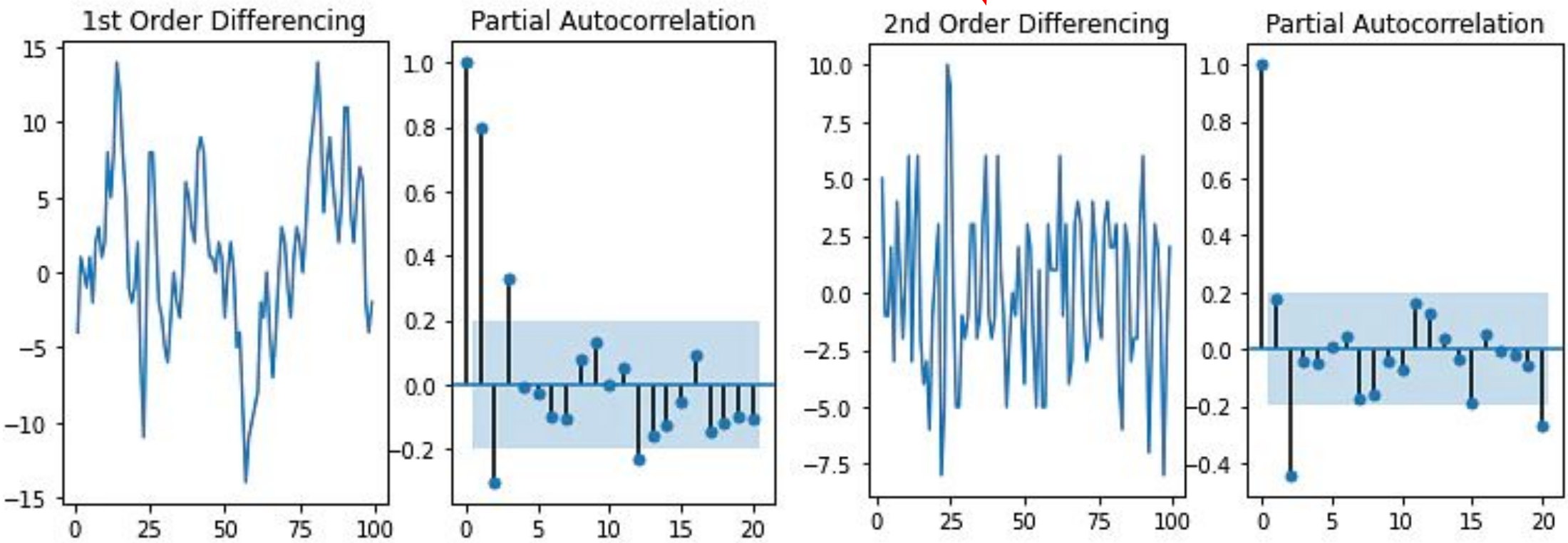
```
p-value:  0.12441935447109487
p-value:  0.07026846015272707
p-value:  2.843428755547158e-17
```

la valeur de p pour le 1er ordre est beaucoup plus proche du seuil 0.05, donc d=1

Modèle ARIMA / Différenciation d ⇒ Test Dickey Fuller

La valeur de d est donc le nombre minimum de différences nécessaires pour rendre la série stationnaire

comportement stationnaire pour d=2



Modèle AR(p)

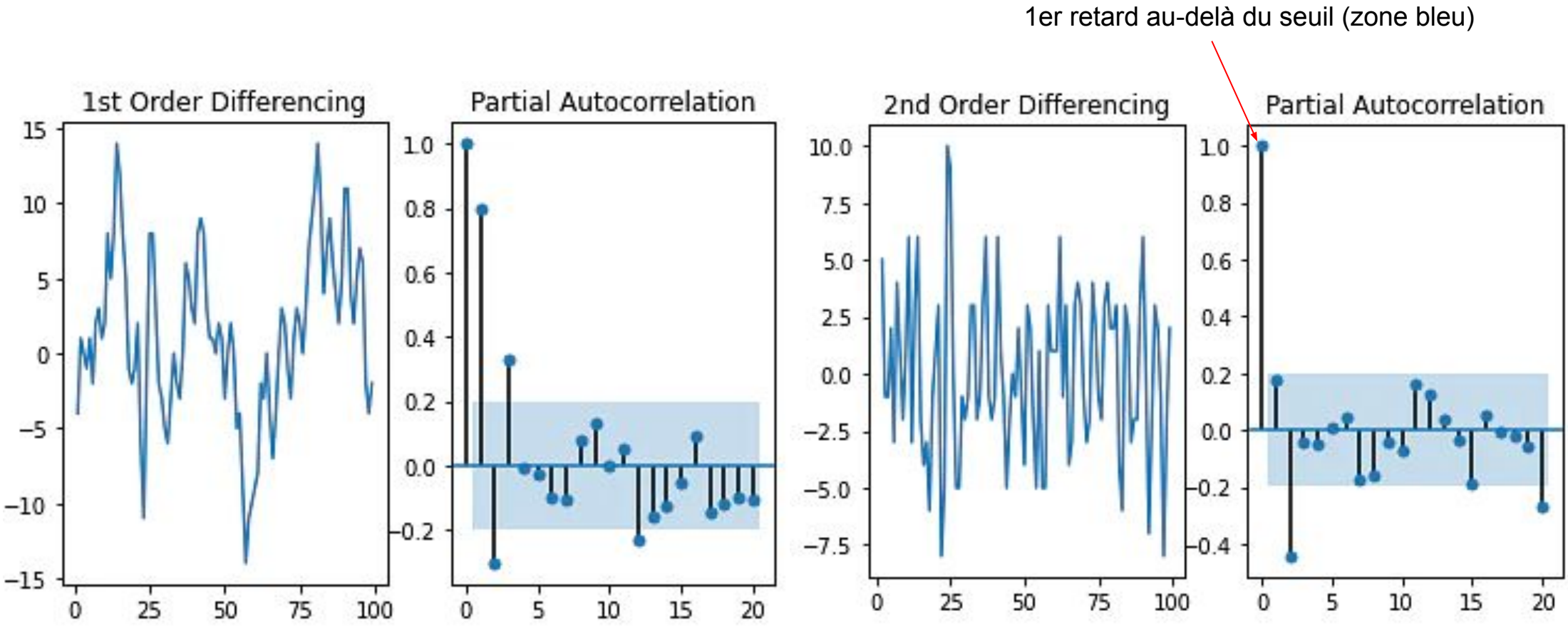
- 'p' est l'ordre du terme 'Auto Regressive' (AR)

$$y_t = \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} + \epsilon_t$$

  - yt est une fonction linéaire des p valeurs précédentes, plus un bruit blanc.
  - Epsilon bruit blanc ; Variable Phi < 1 pour une série stationnaire
  - Pour AR(1) : yt dépend de yt-1, mais yt-1 dépend de yt-2 …

Modèle AR(p)

- p peut être déterminé avec l'analyse de la fonction partielle d'autocorrélation (PACF)
- On considère le retard le plus significatif de la fonction PACF: ici sur le schéma de droite le 1er décalage est le plus significatif. Ainsi, on peut considérer que "p" vaut 1. Le schéma de droite est de l' ordre d=2 (différence).



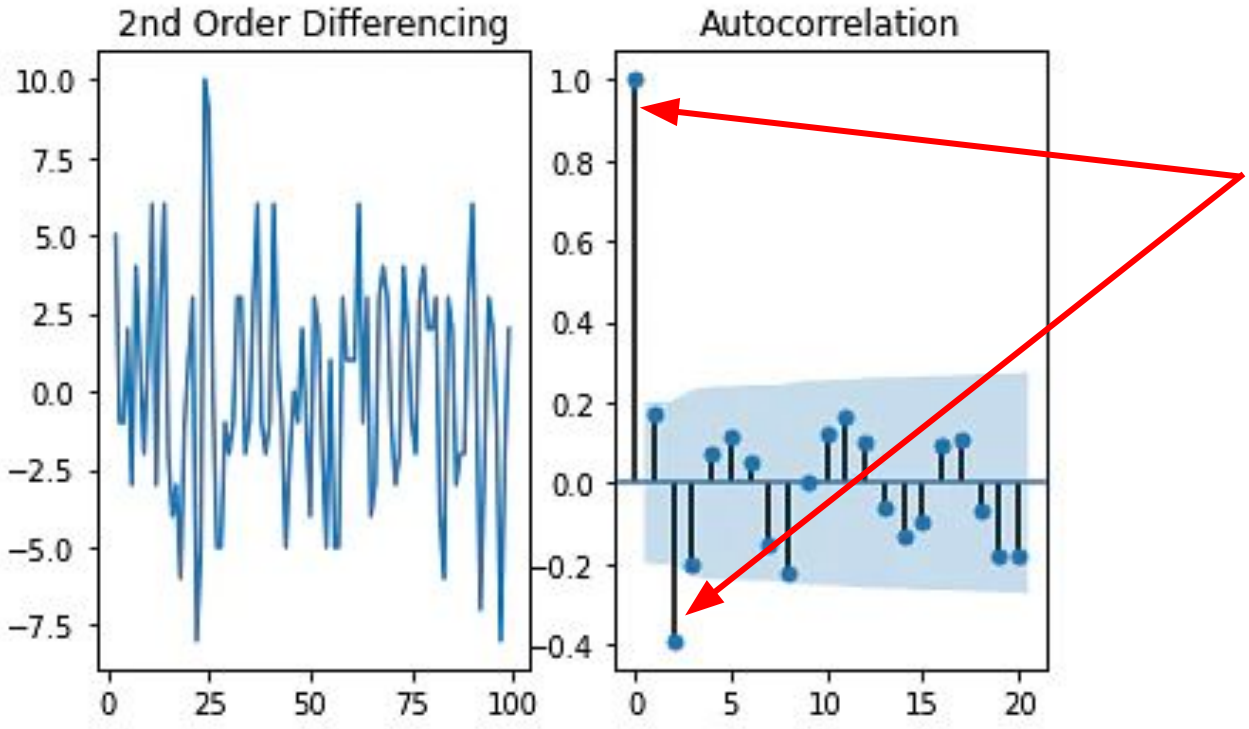
Modèle MA(q)

- Moyenne mobile MA (Moving Average)
- yt= moyenne pondéré de la valeur actuelle et des “q” dernier bruits blanc
- ε_t = bruit blanc à l’instant t
- Θ : coefficient de corrélation

$$y_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q}$$

Modèle MA(q)

- Si les données suivent un modèle de moyenne mobile d'ordre q, l'autocorrélation ACF devient non-significative pour des délais > q
- Analyser tous les pics supérieurs à la zone bleue. Regarder le nombre de décalages franchissant le seuil (zone bleue). En ce sens, à partir de l'image ci-dessous, nous pouvons essayer d'utiliser q=2.



Nombre de décalage franchissant le seuil = 2, donc q=2

Modèle ARIMA(p,d,q)

Combinaison d'un modèle autorégressif d'ordre p
+
Moyenne mobile d'ordre q + résidu ε au temps t
+
Une variable différenciée d fois
+
Constante C

Exemple ARIMA(p=1,d=1,q=2)

$$y'_t = c + \phi_1 y'_{t-1} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

d= 1 différenciée 1 seul fois avec :

$$y_t - y_{t-1} = y'_t = \epsilon_t$$

p= 1 terme autorégressif

q=moyenne mobile d'ordre 2

Modèle ARIMA(p,d,q) / Synthèse des paramètres

- **Paramètre d :** ⇒ Série stationnaire
  - Test ADF (Augmented Dickey Fuller)
  - Courbe ACF ( )
- **Paramètre MA(q) :** ⇒ Moyenne mobile MA (Moving Average)
  - Courbe ACF ( )
- **Paramètre AR(p)** : ⇒ terme 'Auto Regressive'
  - Fonction partielle d'autocorrélation (PACF)

Agenda

Apprentissage supervisé:

- Régression linéaire simple
- Régression linéaire multiple
- Equation normale
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone,
- Machine à vecteurs de support linéaire et non linéaire (SVM),

Apprentissage non supervisé:

- K-NN, K-MEANS
- Arbre de décision

Analyse des séries chronologiques:

- Modèle ARIMA

Outils

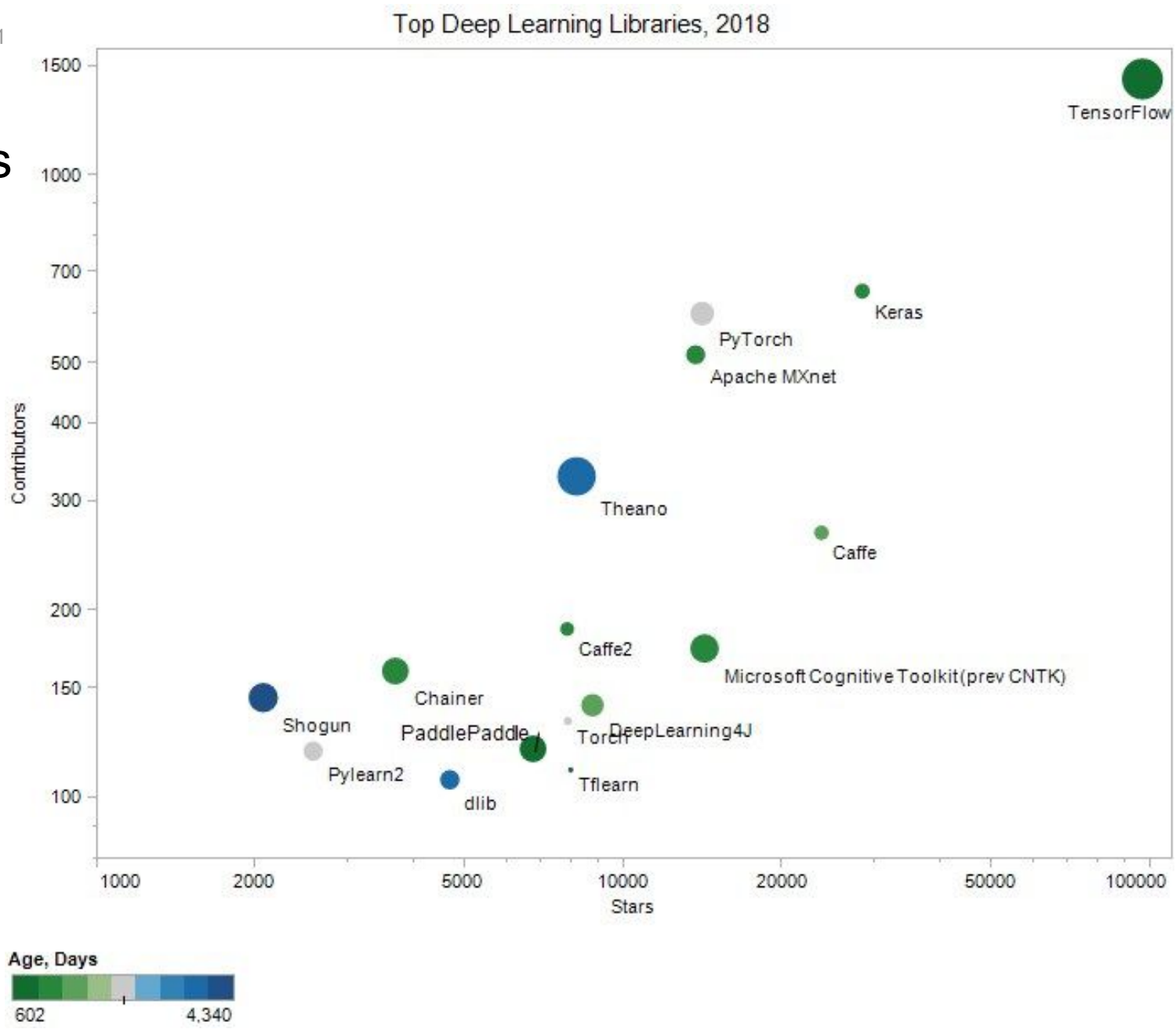
https://www.lemondeinformatique.fr/les-dossiers/lire-effervescence-autour-des-frameworks-ia-1003.html
https://www.entreprises.gouv.fr/files/files/directions_services/etudes-et-statistiques/prospective/Intelligence_artificielle/2019-02-intelligence-artificielle-etat-de-l-art-et-perspectives.pdf

Bibliothèques

Bibliothèque	Langage	Tutoriel	Capacité CNN	Capacité RNN	Architecture	Rapidité	Support GPU multiple	Licence
SCIKIT LEARN	Python	++	++		++	++		BSD
THEANO	Python, C++	++	++	++	+	++	+	BSD
TENSORFLOW	Python	+++	+++	++	+++	++	++	Apache
KERAS	Python	+++	++	+++	++	++	++	MIT
TORCH	Python	+	+++	++	++	+++	++	BSD
CAFFE	C++	+	++		+	+	+	BSD
MXNET	R, Python, Julia, Scala	++	++	+	++	++	+++	Apache
NEON	Python	+	++	+	+	++	+	Apache
CNTK	C++	+	+	+++	+	++	+	MIT

https://www.kdnuggets.com/201

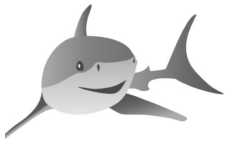
Bibliothèques



Bibliothèques

https://www.gnu.org/software/gsl/

https://www.mlpack.org/



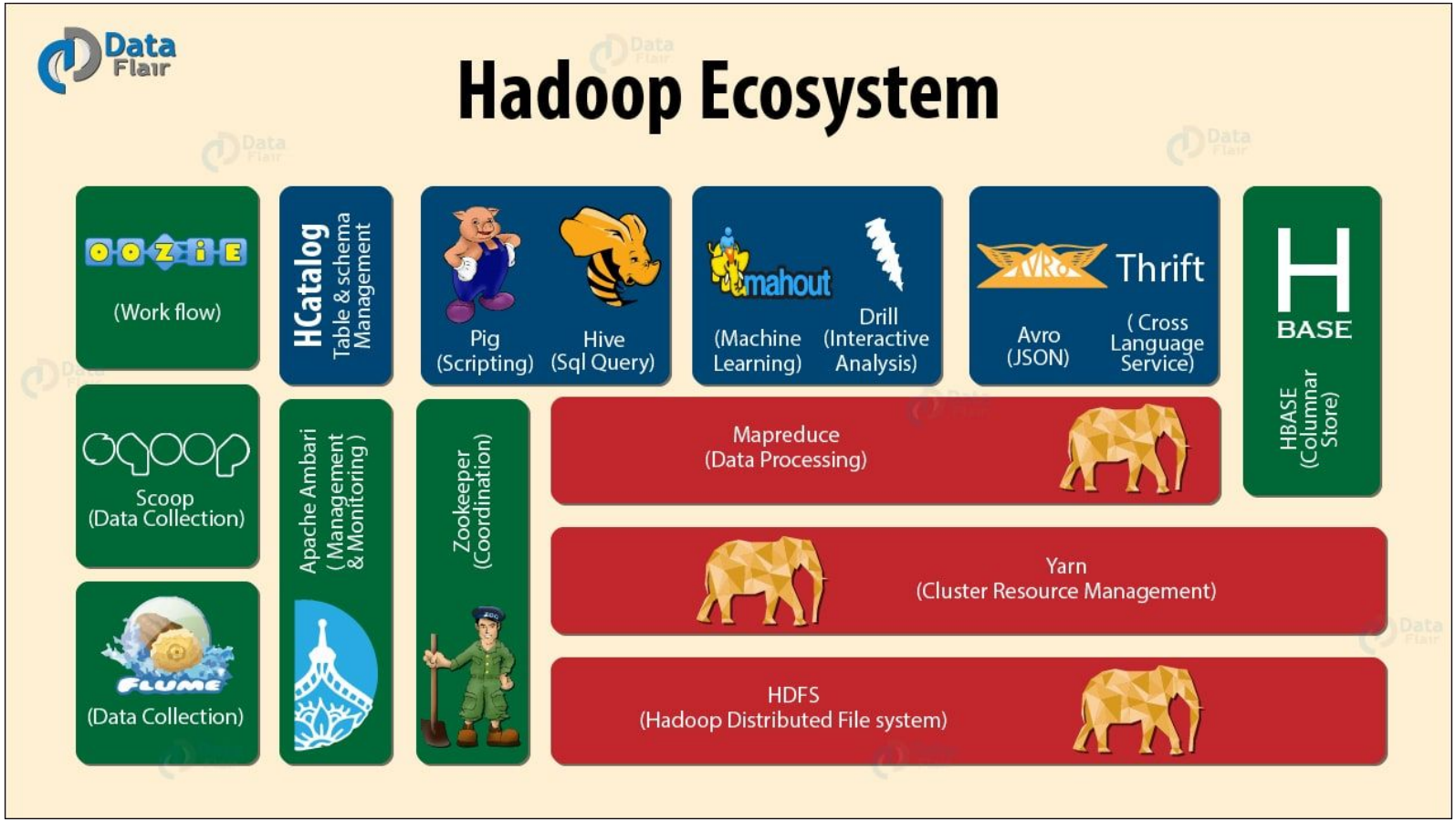
http://image.diku.dk/shark/

https://www.tensorflow.org/

https://www.kaggle.com/

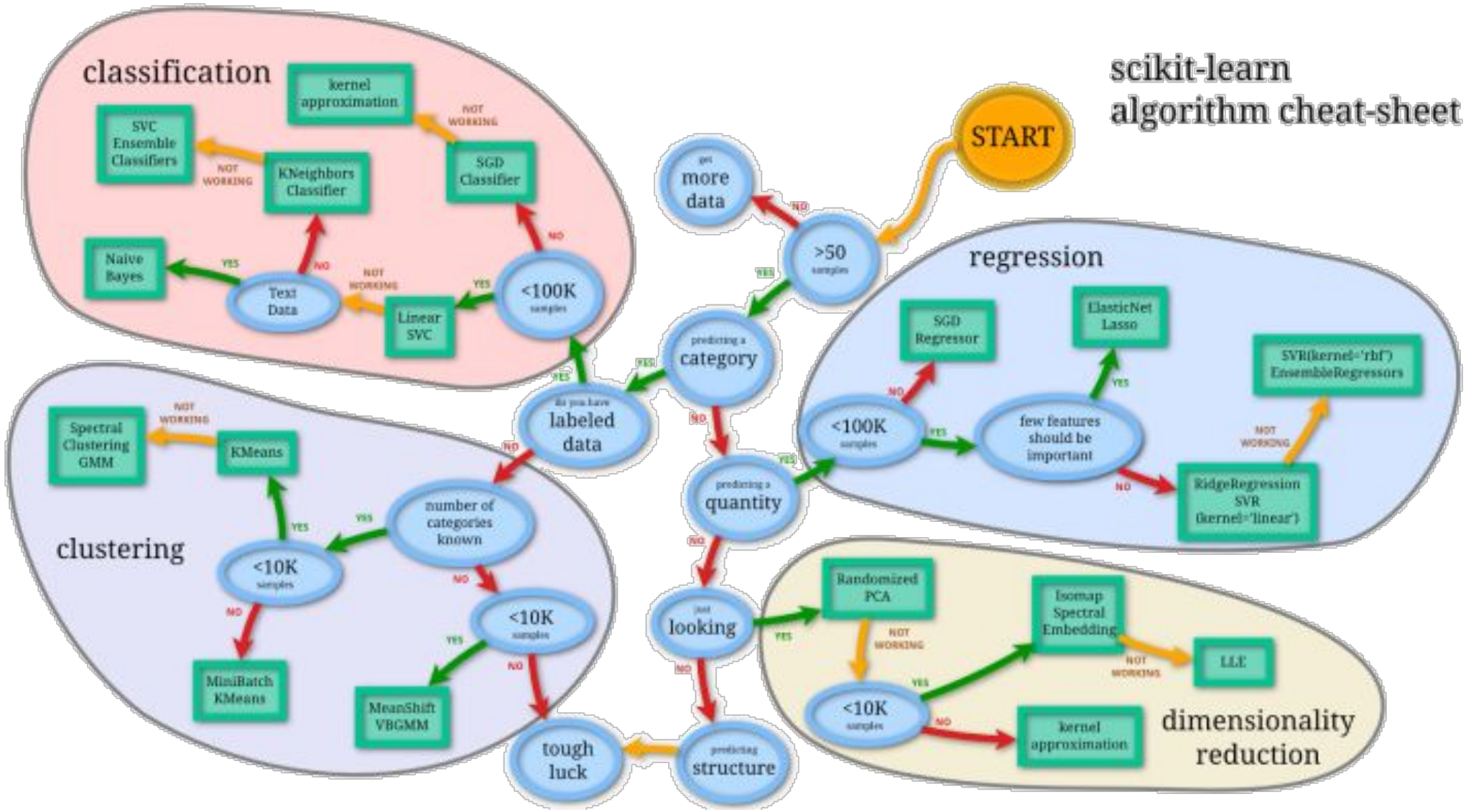
https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/5868381-selectionnez-les-outils-de-data-science-appropries

Le déploiement et la mise en production



scikit-learn

https://scikit-learn.org/stable/
https://scikit-learn.org/stable/user_guide.html
https://scikit-learn.org/stable/supervised_learning.html#supervised-learning



Bibliographie

Intelligence Artificielle - Stuart Russel - Peter Norvig
Machine Learning avec Scikit-Learn - A. Géron

Webographie

https://aws.amazon.com/fr/greengrass/ml/
https://mrmint.fr/
https://developers.google.com/machine-learning/crash-course/glossary?hl=fr
https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning

Python

https://www.python.org/
https://machinelearningmastery.com/load-machine-learning-data-scratch-python/

Open Data

https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research

https://www.data.gouv.fr/fr/
https://www.data.gov/
https://scikit-learn.org/stable/datasets/index.html

http://archive.ics.uci.edu/ml/index.php
https://www.kaggle.com/datasets
http://aws.amazon.com/fr/datasets/

http://dataportals.org/
http://opendatamonitor.eu/
http://quandl.com/
http://archive.ics.uci.edu/ml/datasets.php

http://goo.gl/zDR78y
https://www.reddit.com/r/datasets

https://fr.wikipedia.org/wiki/Iris_de_Fisher#Le_jeu_de_donn%C3%A9es
http://www.python-simple.com/python-scikit-learn/scikit-datasets.php