

# Machine learning : méthodes et solutions

Redha Moulla

27 février - 1 mars 2024

# Objectifs de la formation

- Comprendre les différents modèles d'apprentissage.
- Modéliser un problème pratique sous forme abstraite.
- Identifier les méthodes d'apprentissage pertinentes pour résoudre un problème.
- Appliquer et évaluer les méthodes identifiées sur un problème.
- Faire le lien entre les différentes techniques d'apprentissage

# Plan de la formation

- Aperçu de l'intelligence artificielle
- Apprentissage supervisé
- Apprentissage non supervisé
- Modèles bayésiens
- Chaînes de Markov
- Deep learning
- Mise en production des modèles de machine learning

# Qu'est-ce que l'intelligence artificielle ?

# Définition littérale

## 1. Intelligence

Ensemble des fonctions mentales ayant pour objet la connaissance conceptuelle et relationnelle.

- Larousse

## 2. Artificielle

Qui est produit de l'activité humaine (opposé à la nature).

- Larousse

# Qu'est-ce que l'intelligence ?

La notion d'intelligence recouvre plusieurs facultés cognitives :

- ➊ **Raisonnement** : La capacité à résoudre des problèmes et à faire des déductions logiques.
- ➋ **Apprentissage** : L'aptitude à acquérir de nouvelles connaissances et à s'améliorer grâce à l'expérience.
- ➌ **Perception** : La compétence pour reconnaître et interpréter les stimuli sensoriels.
- ➍ **Compréhension** : L'habileté à saisir le sens et l'importance de divers concepts et situations.
- ➎ **Mémorisation** : La faculté de stocker et de rappeler des informations.
- ➏ **Créativité** : Le pouvoir d'inventer ou de produire de nouvelles idées, de l'originalité dans la pensée.

Mais est-ce que l'intelligence est réductible à des facultés mesurables ?

# Conférence de Dartmouth ?

Article

AI Magazine Volume 27 Number 4 (2006) (© AAAI)

## A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

August 31, 1955

John McCarthy, Marvin L. Minsky,  
Nathaniel Rochester,  
and Claude A. Shannon

■ The 1956 Dartmouth summer research project on artificial intelligence was initiated by this August 31, 1955 proposal, submitted by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon. The original typewritten document of 17 pages plus a title page. Copies of the typewritten are housed in the archives at Dartmouth College and Stanford University. The first 5 pages state the proposal, and the remaining pages give qualifications and interests of the four who proposed the study. In the interest of brevity, this article reproduces only the proposal itself, along with the short autobiographical statements of the proposers.

**W**e propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use lan-

guage, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer. The following are some aspects of the artificial intelligence problems.

### 1. Automatic Computers

If a machine can do a job, then an automatic calculator can be programmed to simulate the machine. The speed and memory capacities of present computers may be insufficient to simulate many of the higher functions of the human brain, but the major obstacle is not lack of machine capacity, but our inability to write programs taking full advantage of what we have.

### 2. How Can a Computer be Programmed to Use a Language

It may be speculated that a large part of human thought consists of manipulating words according to the rules of meaning and rules of inference. From this point of view, forming a generalization consists of admitting a new

*"We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer."*

# L'intelligence artificielle selon John McCarthy

*"It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."*

— John McCarthy

*"Il s'agit de la science et de l'ingénierie de la création de machines intelligentes, en particulier des programmes informatiques intelligents. Cela est lié à la tâche similaire d'utiliser les ordinateurs pour comprendre l'intelligence humaine, mais l'IA n'a pas besoin de se limiter aux méthodes qui sont biologiquement observables."*

— John McCarthy



# Le Test de Turing

Le Test de Turing, développé par Alan Turing en 1950, est une mesure de l'intelligence d'une machine. Le test évalue si une machine peut imiter l'intelligence humaine au point qu'un interrogateur humain ne peut pas distinguer de manière fiable si la communication vient d'une personne ou d'un programme informatique.

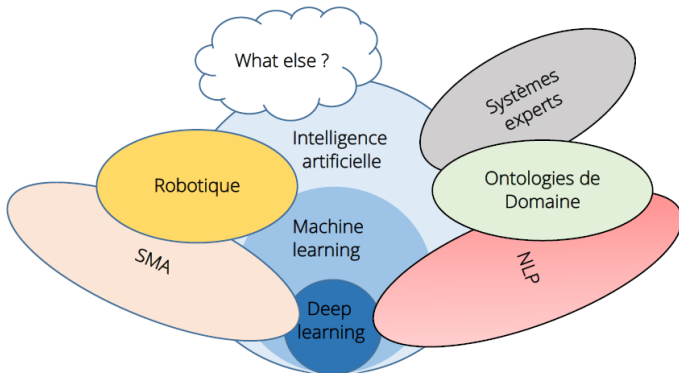
Les points clés du Test de Turing sont :

- Un interrogateur humain engage une conversation avec un humain et une machine, chacun étant caché de la vue de l'interrogateur.
- Si l'interrogateur ne peut pas déterminer systématiquement quelle est la machine, celle-ci est considérée comme ayant passé le test.
- Le test ne mesure pas la connaissance ou la capacité à être véridique, mais plutôt la capacité de reproduire le comportement humain.

*“L'intelligence artificielle n'existe pas.”*

— Luc Julia

# Définition pragmatique de l'intelligence artificielle ?



# IA forte vs IA faible

La distinction entre IA forte et IA faible se réfère à deux approches conceptuelles différentes dans le domaine de l'intelligence artificielle.

## **IA faible :**

- Aussi connue sous le nom d'IA "étroite", elle est conçue pour effectuer des tâches spécifiques et ne possède pas de conscience.
- Les systèmes d'IA faible agissent et réagissent uniquement en fonction des instructions programmées et des algorithmes spécifiques.
- Exemples : assistants virtuels, systèmes de recommandation, reconnaissance vocale.

## **IA forte :**

- Vise à créer des machines dotées de conscience, de compréhension et d'esprit, similaires à l'intelligence humaine.
- L'IA forte serait capable d'apprendre, de raisonner, de résoudre des problèmes et de prendre des décisions indépendamment.
- À ce jour, l'IA forte reste un objectif à atteindre, qui fait l'objet de recherches intensives.

# IA connexionniste vs IA symbolique

## **Intelligence artificielle symbolique :**

Systèmes basés sur des règles et des symboles pour imiter le raisonnement humain.

- Logique
- Ensemble de règles
- Orientée connaissance
- *Knowledge driven*

## **Intelligence artificielle connexionniste**

: Modèles inspirés du cerveau humain pour apprendre des tâches à partir de données.

- Probabiliste
- Apprentissage machine
- Orientée données
- *Data driven*

# Machine learning

# Définition de l'Apprentissage automatique

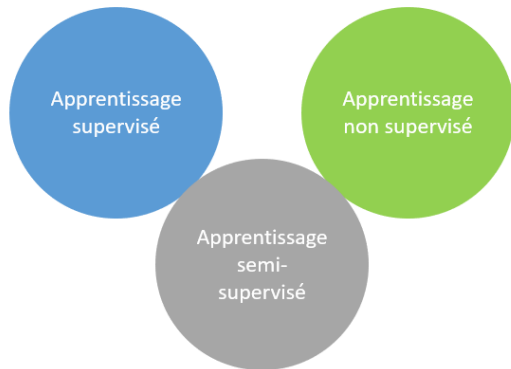
L'apprentissage automatique est une branche de l'intelligence artificielle qui donne aux machines la capacité d'apprendre de données et de s'améliorer avec l'expérience sans être explicitement programmées pour des tâches spécifiques.

Il englobe plusieurs types d'apprentissage :

- **Supervisé** : Les algorithmes apprennent à partir de données étiquetées pour faire des prédictions ou classifications.
- **Non supervisé** : L'apprentissage est effectué sur des données non étiquetées pour trouver des structures cachées.
- **Semi-supervisé** : Combine des éléments des deux premiers types en utilisant une petite quantité de données étiquetées et une grande quantité de données non étiquetées.
- **Par renforcement** : Les modèles apprennent à prendre des décisions en maximisant une récompense à travers des interactions.

Ces méthodes permettent aux machines d'effectuer des tâches telles que la reconnaissance d'image, la traduction automatique et la conduite autonome.

# Typologies d'apprentissage automatique





# L'apprentissage supervisé

**L'apprentissage supervisé** consiste à apprendre un modèle qui associe une étiquette (*label*) à un ensemble de caractéristiques (*features*).

- **Inputs** : un jeu de données *annotées* pour entraîner le modèle.
  - Exemple : des textes (tweets, etc.) avec les *sentiment* associés, positifs ou négatifs.
- **Output** : une étiquette pour un point de donnée inconnu par le modèle.

L'apprentissage supervisé se décline lui-même en deux grandes familles :

- **La classification** : prédire une catégorie ou une classe.
  - Exemple : prédire l'étiquette d'une image (chat, chien, etc.), le sentiment associé à un texte, le centre d'intérêt d'un client à partir de ses commentaires, etc.
- **La régression** : prédire une valeur continue (un nombre réel typiquement).
  - Exemple : prédire le prix d'un appartement, la lifetime value d'un client, etc.

# L'apprentissage non supervisé

**L'apprentissage non supervisé** se réfère à l'utilisation de modèles d'apprentissage automatique pour identifier des patterns et des structures dans des données qui ne sont pas étiquetées.

Principales typologies de l'apprentissage non supervisé :

- **Clustering** : Regroupement de points de données similaires ensemble. Exemple : segmentation de marché, regroupement social.
- **Réduction de dimensionnalité** : Techniques pour simplifier les données tout en préservant leur structure. Exemple : analyse en composantes principales (ACP).

L'apprentissage non supervisé est essentiel pour obtenir des insights lorsque les étiquettes de données sont indisponibles ou coûteuses à obtenir.

# L'apprentissage semi-supervisé

**L'apprentissage semi-supervisé** combine des éléments des approches supervisées et non supervisées. Il utilise un petit ensemble de données étiquetées et un plus grand ensemble de données non étiquetées pour former des modèles.

Cette méthode est particulièrement utile quand :

- Les données étiquetées nécessitent des ressources coûteuses pour les obtenir, mais les données non étiquetées sont abondantes.
- L'ajout d'un peu d'information étiquetée peut améliorer significativement la performance de modèles entraînés avec des données non étiquetées.

Les applications typiques incluent :

- Amélioration de l'exactitude des modèles de classification et de régression.
- Développement de systèmes de recommandation plus performants.
- Traitement de langage naturel et analyse de sentiment lorsque les annotations complètes ne sont pas disponibles.

L'apprentissage tente d'exploiter "le meilleur des deux mondes" de l'étiquetage et de la découverte de structure.

# L'apprentissage par renforcement

**L'apprentissage par renforcement** est une approche de l'apprentissage automatique où un agent apprend à prendre des décisions en interagissant avec un environnement. L'objectif est de maximiser une récompense cumulative.

Principes clés de l'apprentissage par renforcement :

- **Exploration vs Exploitation** : L'agent doit explorer l'environnement pour découvrir des actions qui maximisent la récompense, tout en exploitant ses connaissances actuelles pour prendre des décisions avantageuses.
- **Politique** : Une stratégie qui guide l'agent à choisir une action à partir d'un état donné.
- **Récompense** : Un signal immédiat reçu après chaque action, qui aide à évaluer la performance.
- **Valeur** : Une estimation de la récompense future attendue, tenant compte des récompenses passées et actuelles.

L'apprentissage par renforcement est largement utilisé dans les domaines tels que :

- Robotique, pour des tâches d'automatisation et de navigation.
- Optimisation de la chaîne logistique et gestion de portefeuille en finance.

# Apprentissage supervisé

# Principes de l'apprentissage supervisé

L'apprentissage supervisé est une approche du machine learning où le modèle apprend à partir de données étiquetées pour prédire la sortie pour des données inédites. Un principe clé de cette approche est la réduction de l'espace des solutions :

- **Espace des solutions** : L'ensemble de toutes les prédictions possibles que le modèle peut faire. Dans l'apprentissage supervisé, cet espace est progressivement réduit en apprenant des exemples étiquetés.
- **Optimisation** : Le processus d'apprentissage vise à trouver la fonction de prédiction optimale qui minimise une fonction de coût, réduisant ainsi l'espace des solutions possibles à celle qui est la plus probable pour générer des prédictions correctes.
- **Généralisation** : L'objectif final est de construire un modèle qui non seulement performe bien sur les données d'entraînement, mais qui est également capable de généraliser à de nouvelles données.

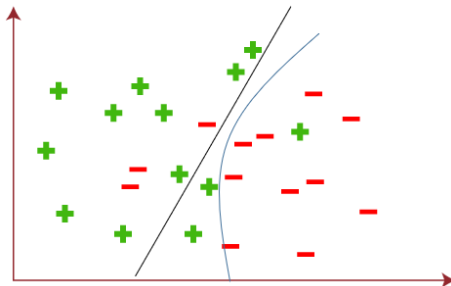
# Détermination d'un modèle

La construction d'un modèle en apprentissage supervisé repose sur trois piliers essentiels qui guident le processus d'apprentissage et déterminent sa réussite :

- **Données** : L'ensemble d'exemples étiquetés utilisés pour l'entraînement du modèle. La qualité, la quantité et la représentativité de ces données sont cruciales pour la capacité du modèle à apprendre et à généraliser à de nouvelles observations.
- **Fonction objective** : Aussi connue sous le nom de fonction de perte ou de coût, elle quantifie l'erreur entre les prédictions du modèle et les valeurs réelles. L'objectif de l'apprentissage est de minimiser cette fonction, guidant ainsi le modèle vers une meilleure performance.
- **Hypothèses (biais inductif)** : Les hypothèses préalables sur la forme du modèle et les relations dans les données. Ces hypothèses concernent le choix de l'algorithme d'apprentissage, et toute préconception sur la distribution des données. Ces hypothèses dirigent l'espace de recherche des solutions possibles et influencent directement la capacité du modèle à apprendre et à généraliser.

# Espace des solutions

Comment choisir la fonction  $f$  dans l'ensemble des solutions  $\mathcal{F}$  ?



- A priori sur la classe du modèle (biais inductif)
- Minimisation du risque empirique



# Biais inductif

Le biais inductif est l'ensemble des a priori (hypothèses) qui déterminent la classe de modèles à laquelle appartient la fonction  $f$ . Ces hypothèses traduisent les connaissances ou les biais de la personne qui construit le modèle.

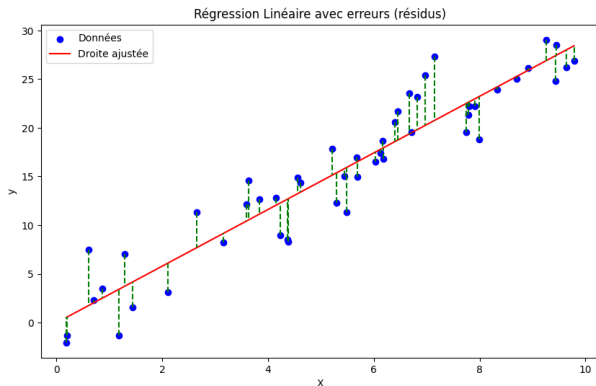
Exemple de biais inductif :

- Hypothèse de linéarité
- Hypothèse de proches voisins
- Hypothèse de maximum de marge
- Etc.

# Minimisation du risque empirique

Une fois la classe  $\mathcal{C}$  est choisie, la fonction  $f \in \mathcal{C}$  est déterminée en minimisant une fonction coût (erreur)  $L$ .

$$R_n(h) = \frac{1}{n} \sum_{i=1}^n L(h(x_i), y_i)$$



# Fonctions de coût

Le choix de la fonction de coût dépend de la nature du problème et des données. On distingue deux types de fonctions de coût.

## Régression

- L'erreur quadratique :  $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
- L'erreur absolue :  $L(\hat{y}, y) = |\hat{y} - y|$

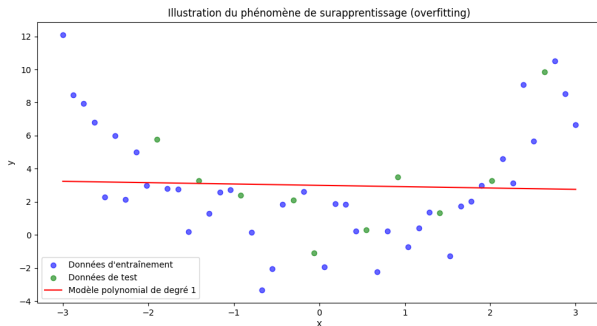
## Classification

- Entropie croisée :  $L(\hat{y}, y) = -y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$
- L'erreur Hinge :  $L(\hat{y}, y) = \max(0, 1 - \hat{y}y)$

# Sous-apprentissage

## Definition

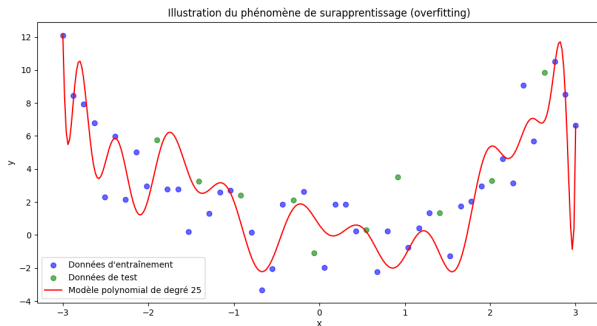
On dit qu'un modèle de machine learning est en régime de sous-apprentissage (underfitting) lorsqu'il n'arrive pas à capturer la complexité (l'information) présente dans le jeu de données d'entraînement.



# Sur-apprentissage

## Definition

On dit qu'un modèle de machine learning est en régime de sur-apprentissage (overfitting) lorsqu'il n'arrive pas à généraliser à des données non encore observées, i.e. lorsqu'il est trop adapté aux données d'entraînement.



# Compromis biais-variance

Soit un ensemble de  $n$  observations  $x_1, x_2, \dots, x_n$  avec les labels correspondants  $y_1, y_2, \dots, y_n$  tels que :  $y_i = f(x_i) + \epsilon_i$ , où  $\epsilon$  a une moyenne nulle et une variance  $\sigma^2$ .

On peut montrer que :

$$E\left((f(x) - y)^2\right) = \text{Biais}\left(f(x)\right) + \text{Var}\left(f(x)\right) + \sigma^2$$

$$\text{où } \text{Biais}\left(f(x)\right) = E(f(x) - y)^2$$

Pour remédier au problème de sur-apprentissage, on applique généralement ce que l'on appelle une régularisation.

# Métriques de performance : régression

On dispose d'un certain nombre de métriques pour évaluer les performances des modèles de machine learning. Celles-ci peuvent être divisées en deux catégories.

## Régression

- L'erreur quadratique moyenne (MSE) : elle est définie comme la moyenne des carrés des écarts entre les prédictions et les valeurs observées.

$$MSE = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2$$

- La racine carrée de l'erreur quadratique moyenne (RMSE) :

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2}$$

# Métriques de performance : classification 1/2

**Accuracy** : L'accuracy est la métrique de base qui permet d'évaluer les performances d'un modèle de classification. Elle est définie comme :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

**Matrice de confusion** : La matrice de confusion est une représentation permettant d'offrir plus de finesse par rapport à l'accuracy, notamment quand le jeu de données est déséquilibré (présence de classes majoritaires). Elle compare les prédictions du modèle avec les valeurs réelles et est structurée comme suit :

		Valeur Prédite	
		Positif	Négatif
Valeur Réelle	Positif	Vrai Positif (VP)	Faux Négatif (FN)
	Négatif	Faux Positif (FP)	Vrai Négatif (VN)



# Métriques de performance : classification 1/2

A partir de la matrice de confusion, on peut dériver d'autres métriques :

- Précision : elle est définie comme la proportion des prédictions correctes parmi toutes les prédictions positives :

$$\text{Précision} = \frac{VP}{VP + FP}$$

- Rappel (recall) : il représente la proportion des vrais positifs correctement prédits par le modèle.

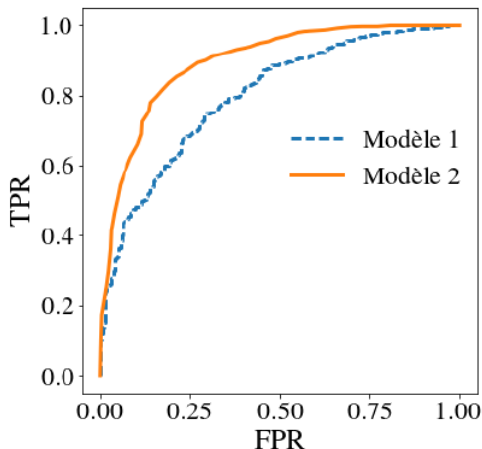
$$\text{Rappel} = \frac{VP}{VP + FN}$$

- Score F1 (F1-score) : Le score F1 est défini comme la moyenne harmonique de la précision et du rappel.

$$\text{Score F1} = 2 \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

# Métriques de performance : courbe ROC

- **ourbe ROC (Receiver-Operator Characteristic)** : Elle décrit l'évolution de la proportion des vrais positifs en fonction de celle des faux positifs.



# Sélection de modèle

Pour sélectionner le modèle le plus pertinent par rapport à une métrique donnée, on applique la méthodologie suivante :

- On partitionne le jeu de données disponible en trois parties : un jeu d'entraînement, un jeu de validation et un jeu de test.
- On entraîne  $M$  modèles sur le jeu d'entraînement.
- On évalue les performances respectives des  $M$  modèles sur le jeu de validation et on sélectionne le meilleur.
- Le modèle sélectionné est ensuite évalué sur le jeu de test. Idéalement, le jeu de test est ainsi utilisé une seule fois.

# Validation croisée K-fold

La validation croisée est une méthode plus robuste pour évaluer les performances des modèles. Il y a deux manières d'appliquer une validation croisée : K-fold et Leave-One-Out (LOO).

La validation croisée K-fold s'effectue selon la méthodologie suivante :

- On partitionne le jeu de données  $\mathcal{D}$  en  $K$  parties ayant approximativement la même taille.
- Pour chaque partie  $\mathcal{D}_k$ , on entraîne le modèle sur l'ensemble des données restantes  $\bigcup_{i \neq k} \mathcal{D}_i$ . Ce modèle est ensuite évalué sur  $\mathcal{D}_k$ .
- On considère la moyennes des performances du modèles sur les différents  $\mathcal{D}_k$ .

La validation croisée K-fold permet ainsi de prendre en compte la variabilité qu'il peut y avoir dans les données. Par ailleurs, on peut également avoir une estimation de la variance du modèle.

# Validation croisée Leave-One-Out

La validation croisée Leave-One-Out est un cas particulier de la validation croisée  $k$ -fold où le nombre de parties (folds) est également au nombre d'observations ( $K = n$ ).

La validation croisée LOO s'effectue selon la méthodologie suivante :

- On partitionne le jeu de données  $\mathcal{D}$  en  $n$  parties ayant chacune  $n - 1$  observations.
- On entraîne le modèle sur chacune des parties  $\mathcal{D}$  (ayant  $n-1$  observations) et on l'évalue sur l'observation restante.
- On considère la moyennes des performances du modèles sur les différents  $\mathcal{D}_k$ .

La validation croisée LOO présente l'avantage que les modèles sont entraînés sur des jeux de données d'une taille plus grande. Par ailleurs, le nombre de modèles obtenus est plus grand également, ce qui peut plaider pour davantage de robustesse. Cependant, ces modèles sont entraînés sur des jeux de données plus similaires les uns par rapport aux autres. D'autre part, les jeux de test étant composés d'une seule observation, les performances risquent de présenter variabilité plus grande.

# Bootstrap

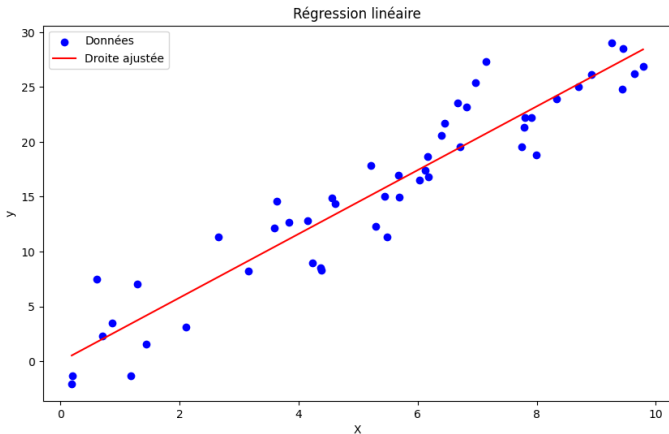
Le Bootstrap est une technique d'évaluation de modèle qui consiste à partitionner le jeu de données  $\mathcal{D}$  en  $K$  échantillons  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$ . Chaque jeu de données est obtenu en tirant  $n$  observations avec remise.

La performance du modèle est alors obtenue en moyennant sur ses différentes performances sur les  $K$  jeux de données.

# Régression linéaire simple

Soit un ensemble de  $n$  observations  $x_1, x_2, \dots, x_n$  avec les labels correspondants  $y_1, y_2, \dots, y_n$ , on cherche le modèle linéaire qui ajuste le mieux ces données.

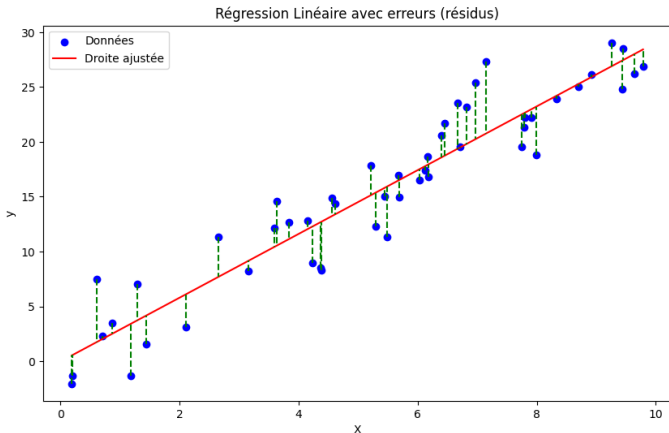
$$\hat{y} = \beta_0 + \beta_1 x$$



# Résidus

## Definition

Soit  $\hat{y}_i = \beta_0 + \beta_1 x_i$  la  $i$  ième prédiction du modèle. Le  $i$  ième résidu, noté  $e_i$ , est alors défini comme l'erreur de prédiction sur  $i$  ième observation :  $e_i = y_i - \hat{y}_i$ .





# Méthode des moindres carrés

On considère la somme des carrés des résidus, notée  $RSS$  :

$$RSS = e_1^2 + e_2^2 + \dots + e_n^2$$

L'approche par moindre carrés consiste à estimer les coefficients  $\beta_0$  et  $\beta_1$  en minimisant la  $RSS$ . Autrement dit :

$$\arg \min_{\beta_0, \beta_1} \left( \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2 \right)$$

Le problème peut être résolu d'une manière analytique. On obtient :

$$\begin{aligned} \beta_1 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \\ \beta_0 &= \bar{y} - \beta_1 \bar{x} \end{aligned} \tag{1}$$

# Régression linéaire multiple

On considère  $n$  observations  $X^1, X^2, \dots, X^n$  où chaque observation  $X^i$  est désormais un vecteur ayant  $p$  composantes ( $p$  variables explicatives).

$$X^i = \begin{pmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_p^i \end{pmatrix}$$

La régression linéaire s'écrit alors :

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

Les coefficients  $\beta_0, \beta_1, \dots, \beta_p$  sont déterminés par la méthode des moindres carrés :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \sum_{i=1}^n \left( y^i - \left( \beta_0 + \sum_{j=1}^p \beta_j x_j^i \right) \right)^2$$

# L'équation normale

La régression linéaire peut être écrite sous une forme vectorielle :

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$$

où  $X$  est appelée matrice de design.

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \dots & x_p^1 \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_1^n & \dots & x_p^n \end{pmatrix}$$

La somme des carrés des résidus est donnée par :

$$RSS = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

On peut montrer que si la matrice de design  $X$  est de rang plein, alors :

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

# Régression polynomiale

La régression linéaire peut prendre en compte les dépendances non linéaires entre les variables explicatives  $x_1, x_2, \dots, x_p$  et la variable expliquée  $y$ . Lorsque cette dépendance prend la forme d'un polynôme de degré  $d$ , la régression linéaire s'écrit alors :

$$\hat{y} = \beta_{00} + \sum_{j=1}^p \beta_{ij} x_j + \sum_{j=1}^p \beta_{ij} x_j^2 + \dots + \sum_{j=1}^p \beta_{ij} x_j^d$$

Les coefficients  $\beta_{ij}$  peuvent être de la même manière, avec la méthode des moindres carrés.

**Remarque :** On peut utiliser n'importe quelle fonction non linéaire pour transformer les variables explicatives ( $\cos$ ,  $\ln$ , etc.). Le modèle reste tout de même linéaire (linéarité par rapport aux coefficients).

# Choix de modèle

Il y a plusieurs manières d'évaluer la pertinence d'un modèle de régression linéaire. Le coefficient de détermination  $R^2$  mesure l'ajustement du modèle. Il est donné par :

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

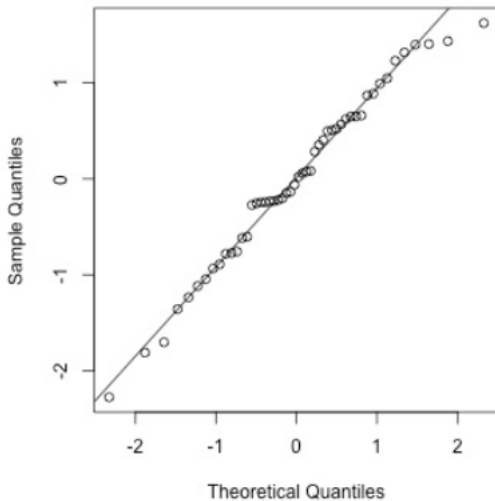
Pour une régression linéaire multiple, on préférera cependant le coefficient de détermination ajusté  $R_a^2$ .

$$R_a^2 = 1 - \frac{n-1}{n-k-1} * (1 - R^2)$$

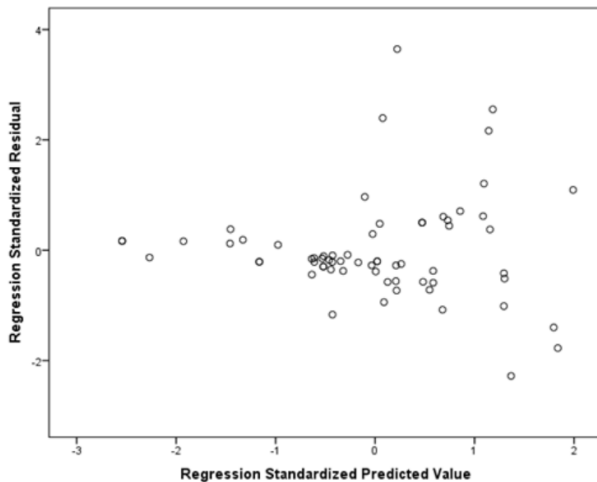
où  $n$  est le nombre d'observations et  $k$  le nombre de variables.

Des critères comme le AIC et le BIC sont également utilisés pour sélectionner un modèle.

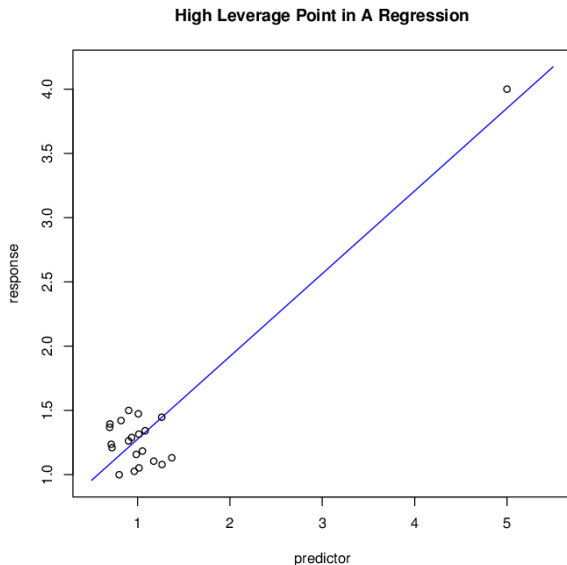
# Diagnostic de la régression linéaire : normalité des résidus



# Diagnostic de la régression linéaire : Homoscédasticité



# Diagnostic de la régression linéaire : effet levier





# Régression Ridge

La régularisation Ridge, dite également régularisation  $L_2$ , permet de remédier au problème de surapprentissage en imposant une contrainte sur les coefficients  $\beta$  lors de la minimisation du risque empirique.

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left( \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 \right)$$

sous la contrainte :

$$\sum_{j=1}^p \beta_j^2 \leq Cte$$

Le problème peut être écrit d'une manière plus compacte :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left( \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right)$$

où le paramètre  $\lambda$  contrôle la force la régularisation. Plus  $\lambda$  est grand, plus le modèle est régularisé.

# Formulation vectorielle de la régression Ridge

La somme des carrés des résidus associée à la régression Ridge s'écrit de la manière suivante :

$$RSS = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

Les coefficient  $\vec{\beta}$  peuvent alors être calculés en minimisant la  $RSS$ .

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

On peut alors montrer que :

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

où  $\mathbf{I}$  est la matrice identité de dimension  $p \times p$ .

# Formulation vectorielle de la régression Ridge

La matrice de design  $X$  de dimensions  $n \times n$  peut être décomposée en valeurs singulières (SVD) de la manière suivante :

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

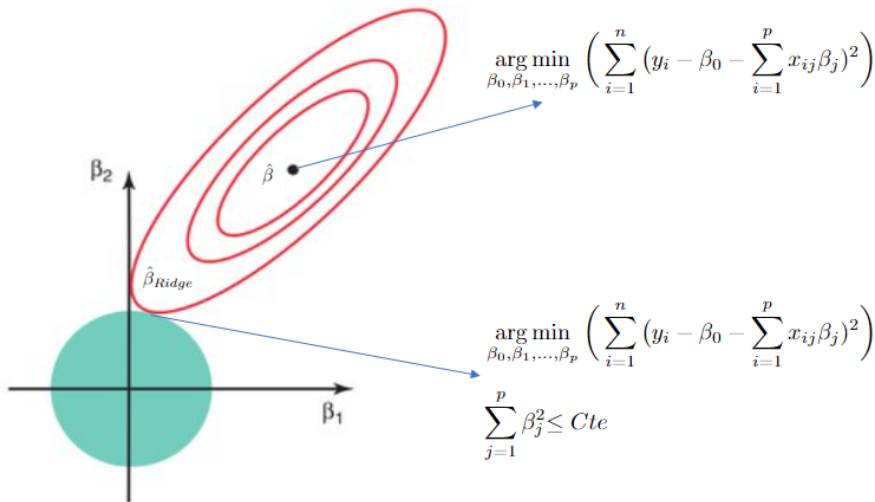
où  $\mathbf{U}$  et  $\mathbf{V}$  sont des matrices orthogonales de dimensions respectives  $n \times p$  et  $p \times p$ .

On peut montrer que :

$$\begin{aligned}\mathbf{X}\boldsymbol{\beta} &= \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \\ &= \sum_{j=1}^p u_j \frac{d_j^2}{d_j^2 + \lambda} u_j^T \vec{y}\end{aligned}\tag{2}$$

On peut noter que le paramètre de régularisation  $\lambda$  tend à réduire l'influence des variables explicatives associées à une faible valeur singulière.

# Interprétation géométrique de régression Ridge



# Régression Lasso

La régularisation Lasso, dite également régularisation  $L_1$ , force les coefficients associés à des variables explicatives ayant une moindre importance vers zéro. C'est ainsi une technique de réduction de la dimensionalité du problème pour avoir un modèle avec peu de variable (plus simple et plus explicable).

La régression Lasso est formalisée de la manière suivante :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left( \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 \right)$$

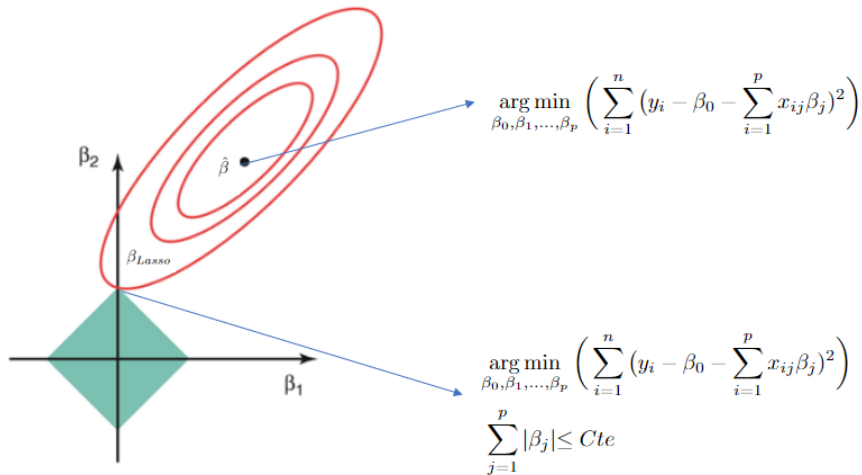
sous la contrainte :

$$\sum_{j=1}^p |\beta_j| \leq Cte$$

Où

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left( \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j| \right)$$

# Interprétation géométrique de régression Lasso



# Elastic net

Elastic net combine les deux approches, Ridge et Lasso, pondérées avec un paramètre  $\alpha \in [0, 1]$ .

Le problème s'écrit ainsi :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left( \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 \right)$$

sous la contrainte :

$$\sum_{j=1}^p (1 - \alpha) |\beta_j| + \alpha \beta_j^2 \leq Cte$$

Ou

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left( \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \left( \sum_{j=1}^p (1 - \alpha) |\beta_j| + \alpha \beta_j^2 \right) \right)$$

# Régression logistique : introduction

La régression logistique est une technique d'analyse statistique utilisée pour modéliser la probabilité d'une variable dépendante binaire. C'est un cas particulier de modèle linéaire généralisé qui est utilisé pour des problèmes de classification.

Principes de la régression logistique :

- **Variable dépendante** : On cherche la probabilité que la variable dépendante ( $y$ ) appartienne à une classe (0 ou 1, vrai ou faux, succès ou échec). Autrement dit, on cherche à modéliser  $P(y = 1)$  en fonction des variables dépendantes (explicatives)  $x$ .
- **Odds ratio** : Plus concrètement, on cherche à exprimer la cote anglaise (odd ratio) en fonction des variables dépendantes ( $x$ ).

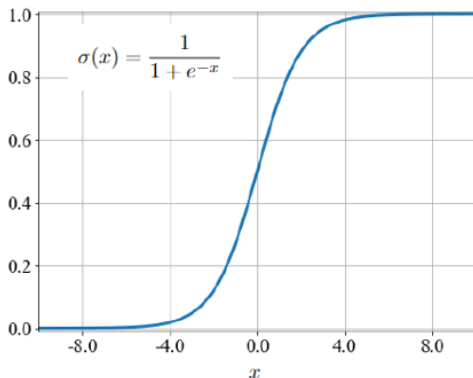
$$\ln \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$



# Régression logistique : fonction sigmoïde

Après quelques simplifications, on peut écrire la probabilité  $p(x)$  (la probabilité pour que  $y$  soit un succès par exemple) :

$$p(\mathbf{x}) = \frac{1}{1 + e^{-(\beta^T \mathbf{x})}}$$



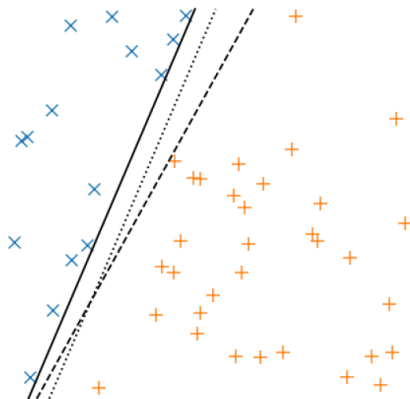
# Calcul des coefficients de la régression logistique

Les coefficients de la régression logistique peuvent être calculés en minimisant le risque empirique par rapport à une fonction de coût sous forme d'entropie croisée :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left( - \sum_{i=1}^n y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i) \right)$$

# Machines à vecteurs de support (SVM)

Considérons un problème de classification binaire. On recherche l'hyperplan séparateur qui maximise la marge  $\gamma$  entre les deux classes. La marge  $\gamma$  étant définie comme la distance entre cet hyperplan et les observations les plus proches.

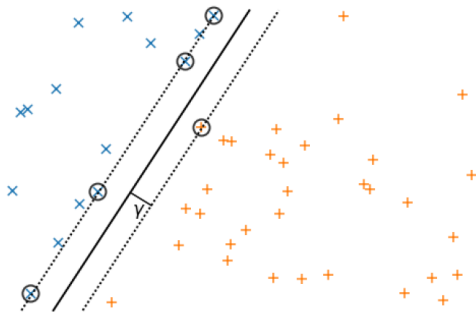


# Position du problème

Soient  $n$  observations  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  et  $n$  poids associés  $w_1, w_2, \dots, w_n$ .

On cherche la fonction  $(w^T x + b)$  telle que lorsque  $h(x) \geq 1$  alors  $x$  appartient à la classe 1 et lorsque  $h(x) \leq -1$  alors  $x$  appartient à la classe 0.

L'hyperplan d'équation  $h(x) = 0$  est ainsi le plan séparateur pour les deux classes.



# Formulation primale

La distance entre un point  $x_k$  et sa projection sur le plan séparateur est donnée par :

$$\gamma_k = y_k \frac{w^T x_k + b}{\|w\|}$$

Les poids  $w$  et le biais  $b$  peuvent être calculés en résolvant le problème d'optimisation suivant :

$$\arg \max_{w,b} \left( \frac{1}{\|w\|} \min_k (y_k (w_k^T x_k + b)) \right)$$

En pratique, il est plus simple de résoudre le problème équivalent :

$$\arg \min \frac{1}{2} \|w\|^2$$

sous la contrainte :

$$y_k (w^T x_k + b) \geq 1$$

# Formulation duale 1/2

Le problème d'optimisation peut être résolu à l'aide des multiplicateurs de Lagrange  $\alpha$ . Le lagrangien s'écrit :

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{k=1}^n \alpha_k \left( y_k (w^T x + b) \right)$$

En minimisant le lagrangien par rapport à  $w$  et  $b$

$$\begin{aligned} \nabla_w L(w, b, \alpha) &= 0 \\ \frac{\partial L(w, b, \alpha)}{\partial b} &= 0 \end{aligned} \tag{3}$$

on obtient :

$$\begin{cases} \sum_{k=1}^n \alpha_k y_k x_k = w \\ \sum_{k=1}^n \alpha_k y_k = 0 \end{cases}$$

## Formulation duale 2/2

En injectant les équations précédentes dans la formule du lagrangien, on obtient :

$$L(w, b, \alpha) = \sum_{k=1}^n \alpha_k - \frac{1}{2} \sum_{k,j} y_k y_j \alpha_k \alpha_j x_k^T x_j$$

Le problème duale s'écrit ainsi :

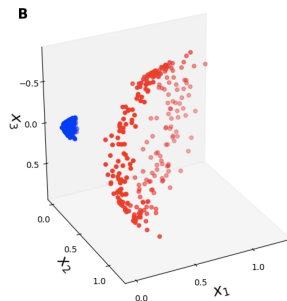
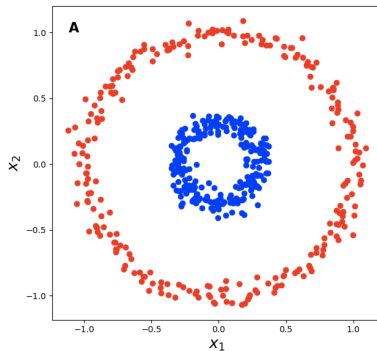
$$\arg \max_{\alpha} \left( \sum_{k=1}^n \alpha_k - \frac{1}{2} \sum_{k,j} y_k y_j \alpha_k \alpha_j x_k^T x_j \right)$$

sous les contraintes

$$\begin{cases} \alpha_k \geq 0, \quad k = 1, 2, \dots, n \\ \sum_{k=1}^n \alpha_k y_k = 0 \end{cases}$$

# SVM à noyau

On considère un problème de classification non linéaire. L'astuce du noyau consiste à augmenter la dimension du problème, pour le résoudre ensuite avec un séparateur linéaire dans le nouvel espace.





# Formulation duale des SVM à noyau

Le problème dual pour les SVM à noyau s'écrit alors :

$$\arg \max_{\alpha} \left( W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j K(x_i, x_j) \right)$$

sous les contraintes

$$\begin{cases} \alpha_k \geq 0, \quad k = 1, 2, \dots, n \\ \sum_{k=1}^n \alpha_k y_k = 0 \end{cases}$$

où la fonction noyau est donné par :

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

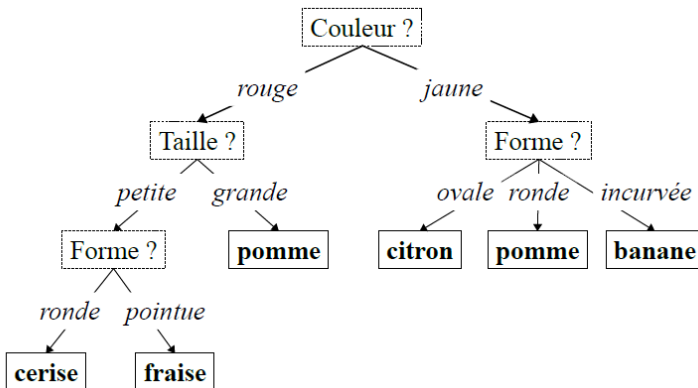
# Exemples de noyaux

Il existe une variété de noyaux pour les SVM. Les plus utilisés étant les noyaux gaussien et polynomial.

- Noyau gaussien :  $k(x, z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2})$
- Noyau polynomial :  $K(x, z) = (x^T z + 1)^d$

# Arbre de décision

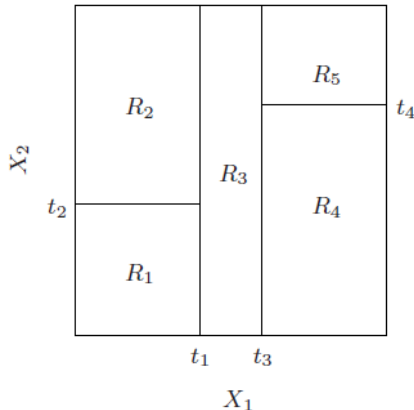
Les arbres de décisions sont des modèles dont le processus de décision est hiérarchique et prend la forme d'un arbre.



# Entraînement des arbres de décision

Les arbres de décisions sont généralement entraînés à l'aide de la technique CART (Classification And Regression Trees).

Etant donné un ensemble d'observations  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , les arbres de décision partitionnent cet espace en plusieurs régions  $R_1, R_2, \dots, R_m$ .



# Régression

Pour une régression, l'approche adoptée pour entraîner un arbre de décision consiste à chercher la meilleure variable  $x_j$  et le meilleur point de partitionnement  $s$  par rapport à cette variable en considérant le problème d'optimisation suivant :

$$\arg \min_{j,s} \left( \sum_{x_i \in R_l(j,s)} (y_i - y_l(j,s))^2 + \sum_{x_i \in R_r(j,s)} (y_i - y_r(j,s))^2 \right)$$

où  $y_l(j,s)$  et  $y_r(j,s)$  sont les moyennes des labels associées à chacune des deux régions formées par le partitionnement.

Il est important ici de noter que nous considérons comme critère d'optimisation la somme des erreurs quadratiques :

$$\sum_{i=1}^n (y_i - f(x_i))^2$$

# Classification

L'approche pour la classification est similaire à celle pour la régression ; la fonction de coût quadratique est remplacée cette fois par une fonction mesurant l'impureté au niveau des feuilles (le degré d'hétérogénéité des labels).

On considère donc le problème d'optimisation suivant :

$$\arg \min_{j,s} \left( \sum_{x_i \in R_l(j,s)} \left( \frac{|R_l(j,s)|}{n} \text{Imp}(R_l(j,s)) + \frac{|R_r(j,s)|}{n} \text{Imp}(R_r(j,s)) \right) \right)$$

On définit la proportion d'exemples d'entraînement qui appartiennent à la classe  $c$  par :

$$p_{mk} = \frac{1}{|R_m|} \sum_{x_i \in R_m} \delta(y_i, k)$$

Différentes fonctions mesurant l'impureté peuvent être considérées :

- Erreur de classification :  $1 - p_{mk}$
- Indice de Gini :  $\sum_{k=1}^K p_{mk}(1 - p_{mk})$
- Entropie croisée :  $-\sum_{k=1}^K p_{mk} \ln(p_{mk})$

# Méthodes ensemblistes (bagging)

Le bagging vise à améliorer les performances des modèles en termes de robustesse (réduction de la variance). Il repose sur deux principes :

- Les prédictions issues d'un ensemble de modèles, même faibles, sont plus robustes que celles issues d'un seul modèle.
- L'échantillonnage aléatoire améliore la robustesse du modèle.

Soit un ensemble de  $n$  observations  $\mathcal{D} = (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ . Le bagging consiste à construire un modèle prédictif selon les étapes suivantes :

- On construit  $B$  échantillons en bootstrap (tirage avec remise) à partir de  $\mathcal{D}$ .
- On entraîne autant de modèles sur chaque échantillon.
- On combine les prédictions issues des  $B$  modèles (vote majoritaire pour la classification et calcul de la moyenne pour la régression).

# Forêts aléatoires (random forests)

La technique des forêts aléatoires (random forests) consiste à appliquer une approche de type bagging sur les arbres de décision.

L'algorithme random forests suit cette procédure :

- Tirer par bootstrap  $B$  échantillons de tailles  $n$  à partir de l'ensemble  $D$ .
- Pour chaque échantillon tiré, construire un arbre en répétant les étapes suivantes jusqu'à atteindre  $n_{min}$ .
  - Tirer d'une manière aléatoire  $m$  variables parmi les  $p$  variables.
  - Sélectionner la meilleure variable avec le meilleur point de partitionnement.
  - partitionner le noeud en deux sous-branches.
- Agréger les arbres construits.

Les prédictions sont agrégées selon qu'il s'agisse de régression ou de classification :

- Régression : moyenne  $f^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$
- Classification : vote majoritaire.



# Remarques

- L'algorithme de random forests intègre nativement une forme de validation croisée. Les performances mesurées sur  $\bigcup_{b_i \neq b_k}$  (out of bag ou OOB) sont souvent proches de celles que l'on pourrait mesurer avec une validation croisée.
- Le nombre de variable tirées pour chaque noeud est généralement donné par  $\sqrt{p}$  pour la classification et  $\frac{p}{3}$  pour la régression. Cet hyperparamètre dépend cependant du problème considéré.
- Lorsque le nombre de variable est élevé alors que le nombre de variables réellement pertinente est faible, la probabilité que les  $p$  variables sélectionnées pour chaque partitionnement incluent des variables pertinentes devient faible, et les performances du modèle en termes de généralisation peuvent se détériorer considérablement.
- L'algorithme random forests permet de restituer des informations sur l'importance des variable (feature importance).

# Boosting

Le boosting est une méthode d'apprentissage ensembliste qui combine plusieurs modèles faibles pour créer un modèle global plus fort. Il se concentre sur la conversion d'apprenants faibles en apprenants forts.

## Principes de Base :

- Construire séquentiellement des modèles faibles, généralement des arbres de décision.
- Chaque modèle suivant essaie de corriger les erreurs du modèle précédent.
- Les modèles sont pondérés en fonction de leur précision et combinés pour obtenir le modèle final.

## Avantages :

- Amélioration de l'exactitude et de l'efficacité des prédictions.
- Bonne performance sur des ensembles de données variés.
- Réduction des risques de surapprentissage comparé aux modèles individuels.

# Adaptive boosting (Adaboost)

**Adaptive Boosting (Adaboost)** : Méthode itérative pour améliorer un ensemble de modèles faibles.

## Algorithme :

- ① Initialisez les poids des observations :  $D_1(i) = \frac{1}{n}$ .
- ② Pour  $t = 1, 2, \dots, T$  :
  - Entraînez un modèle faible  $h_t(x)$ .
  - Calculez l'erreur :  $\epsilon_t = \sum D_t(i)[y_i \neq h_t(x_i)]$ .
  - Calculez le poids du modèle :  $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ .
  - Mettez à jour les poids :  $D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$ .
- ③ Modèle final :  $H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right)$ .

## Remarques :

- Adaboost ajuste les poids des observations pour se concentrer sur les erreurs difficiles.
- Les modèles sont combinés en fonction de leur précision.

# Gradient Boosting

**Gradient Boosting** : Construction itérative d'un modèle additif en minimisant une fonction de perte.

## Processus Itératif :

- ① Commencez avec un modèle initial :  $f_0(x)$ .
- ② Pour  $t = 1, 2, \dots, T$  :
  - Calculez les résidus :  $r_{ti} = - \left[ \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x)=f_{t-1}(x)}$ .
  - Entraînez un modèle faible  $h_t(x)$  sur  $r_{ti}$ .
  - Trouvez  $\gamma_t$  qui minimise  $L(y_i, f_{t-1}(x_i) + \gamma h_t(x_i))$ .
  - Mettez à jour :  $f_t(x) = f_{t-1}(x) + \gamma_t h_t(x)$ .
- ③ Modèle final :  $f_T(x)$ .

## Remarques :

- Chaque modèle successif corrige les erreurs du modèle précédent.
- Les arbres de décision sont généralement utilisés comme modèles faibles.

# Prétraitement des données

# Prétraitement des données

Le prétraitement des données est une étape cruciale en machine learning qui vise à rendre les données plus appropriées pour le modèle. Cela inclut:

- Nettoyage des données
- Gestion des valeurs manquantes
- Normalisation et standardisation
- Encodage des variables catégorielles

# Nettoyage des données

Le nettoyage des données implique de détecter et de corriger (ou supprimer) les erreurs et les incohérences pour améliorer la qualité des données. Cela inclut:

- Correction des erreurs de saisie
- Identification et traitement des valeurs aberrantes
- Suppression des doublons

# Normalisation et standardisation

La mise à l'échelle des données est essentielle pour de nombreux algorithmes de machine learning.

- **Normalisation** redimensionne les valeurs dans un intervalle  $[0, 1]$  ou  $[-1, 1]$ .
- **Standardisation** redimensionne les données pour qu'elles aient une moyenne de 0 et un écart type de 1.



# Encodage des variables catégorielles

Les modèles de machine learning nécessitent des entrées numériques pour qu'elles soient traitées par la machine. Les variables catégorielles doivent donc être transformées en variables numériques. Il existe généralement deux approches :

- **Encodage One-Hot** crée une nouvelle colonne pour chaque catégorie.
- **Encodage ordinal** attribue un nombre unique à chaque catégorie selon un ordre spécifique.

# Gestion des valeurs manquantes

Les valeurs manquantes représentent l'absence d'information dans un ensemble de données et constituent un défi courant en machine learning et analyse de données. Elles peuvent survenir pour diverses raisons, telles que :

- Erreurs de saisie des données
- Perte de données lors de la transmission
- Non-réponse dans les enquêtes ou questionnaires
- Suppression intentionnelle de données pour des raisons de confidentialité

La gestion appropriée des valeurs manquantes est cruciale pour maintenir la qualité et la fiabilité des modèles prédictifs. Elle implique des techniques telles que l'imputation, la suppression des observations manquantes, ou l'utilisation de modèles capables de gérer directement les données manquantes.

# Typologies des données manquantes

Comprendre la nature des données manquantes est crucial pour choisir la méthode de traitement appropriée.

- **MCAR (Missing Completely At Random)** : La probabilité qu'une donnée soit manquante est la même pour toutes les observations. L'absence de données est totalement indépendante des données observées ou manquantes.
- **MAR (Missing At Random)** : La probabilité qu'une donnée soit manquante dépend des données observées et non des données manquantes elles-mêmes.
- **MNAR (Missing Not At Random)** : La probabilité qu'une donnée soit manquante dépend des données manquantes elles-mêmes.

La distinction entre ces catégories influence la stratégie d'imputation et l'analyse des données.

# Stratégies de base pour les données manquantes

Selon la typologie, différentes stratégies peuvent être adoptées :

- **Pour MCAR et MAR :**

- Suppression de lignes ou de colonnes
- Imputation simple (moyenne, médiane, mode, etc.)

- **Pour MNAR :**

- Techniques d'imputation plus sophistiquées
- Modélisation spécifique pour estimer les valeurs manquantes

L'identification du type de données manquantes aide à choisir la méthode la plus adaptée et à minimiser le biais introduit par l'imputation.

# Imputation simple vs. Imputation multiple

- **Imputation simple :**

- Remplace les valeurs manquantes par une estimation (moyenne, médiane, etc.).
- Facile à implémenter mais ne tient pas compte de l'incertitude autour des valeurs imputées.

- **Imputation multiple :**

- Génère plusieurs ensembles complets de données en remplaçant les valeurs manquantes par un ensemble de valeurs plausibles.
- Permet d'estimer l'incertitude autour des imputations et offre des estimations plus robustes.

L'imputation multiple est particulièrement utile pour les données MAR et MNAR, où l'incertitude autour des valeurs manquantes doit être prise en compte.

# Techniques avancées d'imputation

Des techniques avancées peuvent mieux gérer la complexité des données manquantes :

- **Imputation KNN (k-Nearest Neighbors)** : Utilise les k observations les plus similaires pour imputer les valeurs manquantes.
- **Imputation par modèles prédictifs** : Utilise des modèles comme les arbres de décision, forêts aléatoires ou réseaux de neurones pour prédire les valeurs manquantes.
- **Imputation par chaînes de Markov Monte Carlo (MCMC)** : Une approche probabiliste qui génère des valeurs plausibles pour les données manquantes en se basant sur leur distribution.

Ces méthodes tentent de modéliser la structure sous-jacente des données pour une imputation précise et pour minimiser le biais.

# Considérations finales sur le traitement des données manquantes

- Identifier la typologie des données manquantes est essentiel pour choisir la méthode d'imputation appropriée.
- Les méthodes d'imputation doivent être choisies en tenant compte de l'impact sur la distribution des données et sur les analyses ultérieures.
- L'imputation multiple est recommandée pour obtenir des estimations plus robustes et évaluer l'incertitude autour des valeurs imputées.
- L'exploration des données et la compréhension du contexte sont cruciales pour interpréter correctement les raisons derrière les données manquantes et pour choisir la méthode d'imputation la plus adéquate.
- Il est important de documenter le processus d'imputation et d'évaluer l'impact de différentes stratégies sur les résultats du modèle.

Considérer le traitement des données manquantes comme une composante intégrale de la préparation des données peut améliorer significativement la qualité et la fiabilité des analyses de machine learning.

# Apprentissage non supervisé



# Apprentissage non supervisé

Dans l'apprentissage non supervisé, on considère  $n$  observations sans labels. On s'intéresse fondamentalement à la probabilité jointe de ces observations.

On peut distinguer deux grandes catégories d'apprentissage non supervisé :

- Clustering (partitionnement) : cela consiste à partitionner les  $n$  observations en  $K$  groupes pertinents (généralement le critère de pertinence a une signification d'un point de vue métier).
- Réduction de dimension : il s'agit de trouver une représentation des données originelles dans un nouvelles espace de plus petite dimension. Cela peut être effectué à différentes fins : visualisation des données, compression des données, amélioration des performances du modèles (modèles plus robuste, plus explicable, etc.).

# Méthode des k-moyennes (k-means)

Soit un ensemble de  $n$  observations  $(x_1, x_2, \dots, x_n)$ . La méthode des k-means vise à partitionner cet ensemble selon  $k$  groupes en minimisant la variance globale à l'intérieur des clusters. Plus formellement, le problème des k-means s'écrit :

$$\arg \min_{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K} \sum_{k=1}^K \sum_{x \in \mathcal{C}_k} \|x - u_k\|^2$$

Une telle optimisation est cependant difficile et très coûteuse en temps.

# Algorithme de Lloyd

Considérons  $n$  observations  $x_1, x_2, \dots, x_n$  et un nombre déterminé  $K$  de clusters, l'algorithme de Lloyd consiste à exécuter les étapes suivantes :

- On choisit aléatoirement  $k$  centroïdes parmi les  $n$  observations.
- On affecte chaque observation  $x_i$  au cluster dont le centroïde est plus proche.

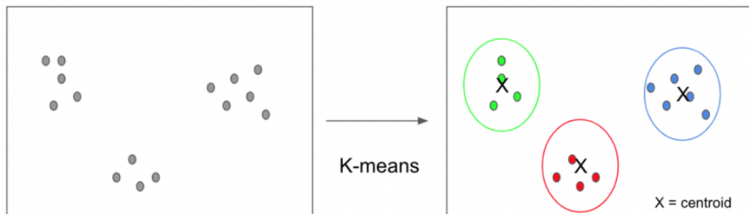
$$k(x_i) = \arg \min_{k=1,2,\dots,K} \|x_i - u_k\|$$

- recalculer les centroïdes de chaque cluster avec la nouvelle configuration.

$$\mu_k = \frac{1}{C_k} \sum_{x_i \in C_k} x_i$$

- On répète la procédure jusqu'à convergence (jusqu'à ce que les centroïdes soient stables).

L'algorithme de Lloyd tente de regrouper les données en clusters en minimisant les distances entre les points d'un même cluster tout en maximisant les distances entre points appartenant à différents clusters.

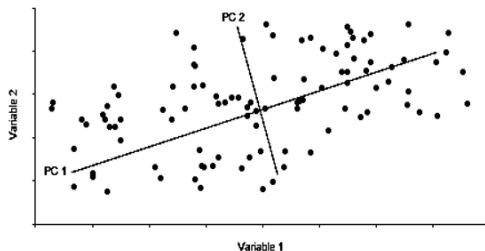


# Remarques

- L'algorithme des k-means étant basé sur une distance euclidienne, il est nécessaire de normaliser les données avant de l'exécuter.
- L'algorithme des k-means est très sensible aux données aberrantes (outliers). Il faut donc considérer les données d'une manière attentive. Cependant, cela permet également d'utiliser l'algorithme des k-means pour la détection automatique des outliers.
- Les centroïdes étant initialisés d'une manière aléatoire, les clusters obtenus ne sont pas stables ; les clusters peuvent changer d'une exécution à l'autre. Il existe cependant une variante plus stable, appelée k-means++, qui permet de sélectionner les centroïdes d'une manière semi-aléatoire.
- Il est possible de partitionner les données avec une métrique plus générale que la distance euclidienne. On peut définir un algorithme k-means à noyau sur un espace de Hilbert pour aller au-delà de la métrique euclidienne.
- K-means n'est pas adapté aux données en grande dimension.

# Analyse en Composantes Principales (ACP)

L'Analyse en Composantes Principales (ACP) est une technique statistique de réduction de dimensionnalité. Elle transforme les données en un nouveau système de coordonnées où la plus grande variance est capturée sur les premiers axes, appelés composantes principales.



# Formulation mathématique de l'ACP

Soit  $X$  une matrice de données de dimension  $n \times p$  ( $n$  observations,  $p$  variables), centrée (moyenne nulle). L'ACP cherche à trouver les vecteurs propres et les valeurs propres de la matrice de covariance  $C = \frac{1}{n-1} X^T X$ .

Les composantes principales sont données par les vecteurs propres  $u_k$  de  $C$ , ordonnés par leurs valeurs propres correspondantes  $\lambda_k$  en ordre décroissant.

La  $k$ -ième composante principale de l'ensemble de données est donnée par :

$$Z_k = Xu_k$$

Les valeurs propres  $\lambda_k$  représentent la variance expliquée par chaque composante principale.

# Décomposition en valeurs propres

La matrice de covariance  $C$  peut être décomposée comme suit :

$$C = VLV^T$$

où  $V = [u_1, u_2, \dots, u_p]$  est la matrice des vecteurs propres et  $L$  est une matrice diagonale des valeurs propres  $\lambda_k$ .

La contribution de chaque composante principale à la variance totale est donnée par :

$$\frac{\lambda_k}{\sum_{i=1}^p \lambda_i}$$



# Interprétation des composantes principales

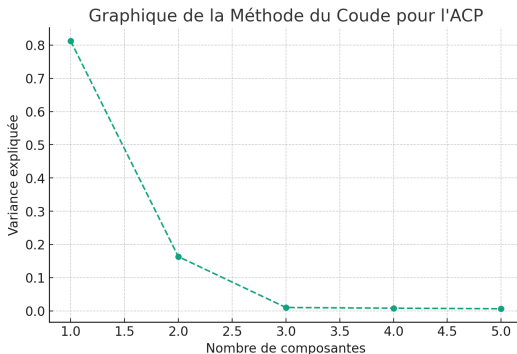
Chaque composante principale est une combinaison linéaire des variables d'origine. L'interprétation des composantes principales peut être réalisée en examinant les coefficients (charges) de ces combinaisons linéaires.

- Plus le coefficient absolu d'une variable est grand dans une composante, plus cette variable contribue à la variance capturée par cette composante.
- La direction et la magnitude des composantes principales peuvent être visualisées sur un biplot.

# Choix du nombre de composantes principales

Le nombre de composantes à retenir est déterminé en fonction du pourcentage de variance totale que l'on souhaite expliquer.

- La méthode du coude (Scree plot) : un graphique montrant la proportion de la variance expliquée en fonction du nombre de composantes.
- Critère de Kaiser : retenir les composantes avec une valeur propre supérieure à 1.



# Réduction de dimensionnalité

L'ACP est souvent utilisée pour réduire la dimensionnalité des données en conservant les composantes qui capturent la majorité de la variance.

- Cela permet une visualisation simplifiée des données en 2D ou 3D.
- La réduction de dimensionnalité peut également aider à améliorer l'efficacité des algorithmes d'apprentissage supervisé.

L'ACP a de nombreuses applications dans différents domaines :

- Analyse de données en biologie, finance, marketing.
- Traitement d'images et de signaux.
- Reconnaissance de motifs et classification.

# Limitations de l'ACP

Bien que l'ACP soit un outil puissant, elle présente certaines limitations :

- Sensibilité aux outliers.
- Difficulté d'interprétation des composantes si les variables sont fortement corrélées.
- La réduction de dimensionnalité peut entraîner une perte d'informations importantes.

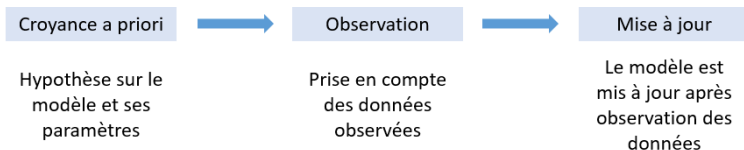
# Méthodes bayésiennes

# Introduction aux méthodes bayésiennes

Les méthodes bayésiennes fournissent un cadre complet pour construire des modèles prédictifs et gérer les incertitudes liées à leurs paramètres. Elles sont basées sur le théorème de Bayes pour mettre à jour les connaissances ou croyances antérieures avec de nouvelles données observées.

L'approche bayésienne repose sur deux concepts :

- 1 **La probabilité a priori** : La connaissance ou l'hypothèse initiale sur les paramètres avant d'observer les données. Elle reflète les croyances préalables sur les paramètres du modèle.
- 2 **La probabilité a posteriori** : Les paramètres du modèle (plutôt leur probabilité) sont mis à jour des paramètres après avoir pris en compte les nouvelles données observées en utilisant le théorème de Bayes.



# Formulation mathématique de l'approche bayésienne

Concrètement, la probabilité du modèle est mise à jour en utilisant le théorème de Bayes :

$$P(\theta|\text{données}) = \frac{P(\text{données}|\theta) \cdot P(\theta)}{P(\text{données})}$$

où :

- $P(\theta|\text{données})$  est la probabilité a posteriori des paramètres  $\theta$  étant donné les données.
- $P(\text{données}|\theta)$  est la vraisemblance des données observées sous l'hypothèse des paramètres  $\theta$ .
- $P(\theta)$  est la probabilité a priori des paramètres avant d'observer les données.
- $P(\text{données})$  est la probabilité marginale des données, agissant comme une constante de normalisation.

Le calcul de  $P(\text{données})$ , la constante de normalisation, est généralement très délicat et nécessite des approximations.

# Estimation des paramètres dans les modèles bayésiens

L'entraînement d'un modèle bayésien revient à estimer les distributions a posteriori des paramètres sur la base des données observées. Cette distribution a posteriori capture notre compréhension et notre incertitude concernant les paramètres après avoir pris en compte les informations fournies par les données.

$$\theta_{\text{post}} = \arg \max_{\theta} P(\theta | \text{données})$$

Les méthodes d'estimation des paramètres incluent :

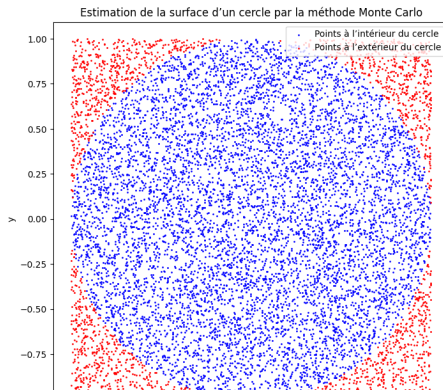
- ➊ **Méthodes analytiques** : Dans certains cas, la distribution a posteriori peut être calculée directement à partir de formules analytiques, souvent lorsque le modèle prior et la vraisemblance sont conjugués.
- ➋ **Méthodes d'échantillonnage** : Lorsque les solutions analytiques ne sont pas réalisables, des techniques d'échantillonnage telles que la chaîne de Markov Monte Carlo (MCMC) sont utilisées pour générer des échantillons de la distribution a posteriori.
- ➌ **Approximation variationnelle** : Une alternative aux méthodes d'échantillonnage, où l'objectif est de trouver une distribution simple qui se rapproche au mieux de la distribution a posteriori complexe.



# Intuition derrière les techniques de Monte Carlo

Le graphique ci-dessus illustre l'utilisation de la méthode de Monte Carlo pour estimer la surface d'un cercle unitaire. En générant aléatoirement des points dans le carré  $[1, 1] \times [1, 1]$  et en comptant la proportion de ces points qui se trouvent à l'intérieur du cercle (points bleus), on peut estimer la surface du cercle.

L'estimation de la surface du cercle obtenue à partir de cette simulation est de 3.1492, ce qui est proche de la valeur théorique de  $\pi = 3.1416$ .



# Régression bayésienne

La régression bayésienne est une extension des méthodes de régression classiques qui incorpore l'incertitude dans les estimations des paramètres de régression. En utilisant l'approche bayésienne, chaque paramètre de régression est traité comme une variable aléatoire avec sa propre distribution a priori, qui est mise à jour en une distribution a posteriori en tenant compte des données observées.

La formulation de base de la régression bayésienne linéaire est :

$$y = \mathbf{X}\beta + \epsilon$$

où  $y$  est le vecteur de réponse,  $\mathbf{X}$  est la matrice de conception,  $\beta$  est le vecteur des coefficients de régression, et  $\epsilon$  est le terme d'erreur, souvent supposé suivre une distribution normale avec une moyenne de zéro.

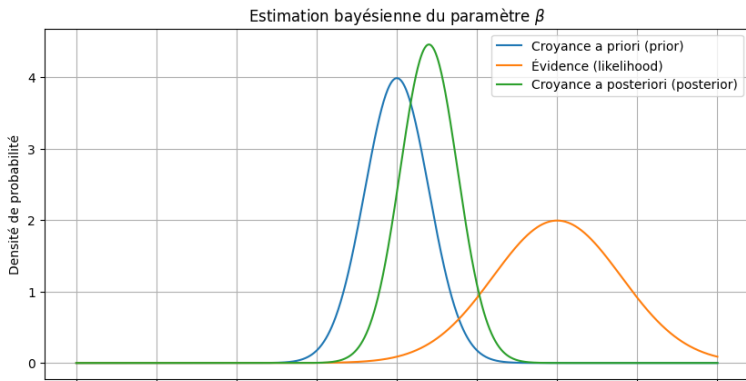
Le processus d'estimation bayésienne se concentre sur la détermination de la distribution a posteriori des coefficients  $\beta$  :

$$P(\beta|y, \mathbf{X}) \propto P(y|\beta, \mathbf{X}) \cdot P(\beta)$$

# Avantages de la régression bayésienne

La régression bayésienne permet de :

- Incorporer des connaissances a priori dans les estimations des coefficients.
- Obtenir une mesure de l'incertitude pour chaque estimation de coefficient à travers la distribution a posteriori.
- Faire des prédictions et calculer des intervalles de prédiction qui prennent en compte l'incertitude de tous les paramètres du modèle.



# Classification Bayésienne

La classification bayésienne repose sur l'utilisation de la probabilité conditionnelle et le théorème de Bayes pour prédire la catégorie d'une nouvelle observation. Elle est particulièrement efficace dans les situations où la dimensionnalité des données est élevée ou lorsque les données sont incomplètes.

La formule fondamentale de la classification bayésienne est le calcul de la probabilité a posteriori pour chaque classe  $C_k$  donnée une observation  $\mathbf{x}$  :

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k) \cdot P(C_k)}{P(\mathbf{x})}$$

où :

- $P(C_k|\mathbf{x})$  est la probabilité a posteriori de la classe  $C_k$  étant donné l'observation  $\mathbf{x}$ .
- $P(\mathbf{x}|C_k)$  est la vraisemblance de l'observation  $\mathbf{x}$  dans la classe  $C_k$ .
- $P(C_k)$  est la probabilité a priori de la classe  $C_k$ .
- $P(\mathbf{x})$  est la probabilité marginale de l'observation  $\mathbf{x}$ , souvent calculée comme la somme des vraisemblances de  $\mathbf{x}$  sur toutes les classes pondérée par leur probabilité a priori.

# Classification avec Naive Bayes

Le classificateur Naive Bayes est un modèle probabiliste basé sur le théorème de Bayes, avec une hypothèse simplificatrice d'indépendance conditionnelle entre les caractéristiques données la classe.

Points clés :

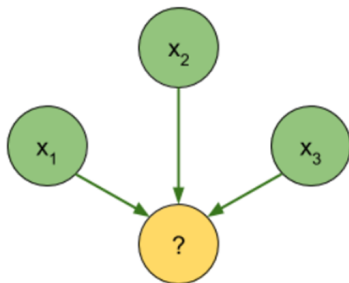
- Facile à construire et particulièrement utile pour de très grands ensembles de données.
- Malgré son hypothèse naïve d'indépendance, il peut être étonnamment efficace.
- Largement appliqué dans la classification de documents, le filtrage anti-spam et les systèmes de recommandation.

# Hypothèse d'indépendance dans Naive Bayes

L'hypothèse d'indépendance de Naive Bayes suppose que la présence (ou l'absence) d'une caractéristique particulière est indépendante de la présence (ou l'absence) de toute autre caractéristique, donnée la classe.

Cela simplifie le calcul de  $P(x|C_k)$  car :

$$P(x_1, \dots, x_n | C_k) = P(x_1 | C_k) \times \dots \times P(x_n | C_k)$$



# Avantages et limites de Naive Bayes

## Avantages :

- Efficacité en temps et en espace, même avec de grands ensembles de données.
- Performance robuste et souvent compétitive avec des classificateurs plus sophistiqués.
- Bonne performance sur des données multidimensionnelles et catégorielles.

## Limites :

- L'hypothèse d'indépendance peut être irréaliste et peut affecter la performance.
- Moins performant si les caractéristiques sont corrélées.
- Peut être biaisé si l'ensemble de données d'entraînement ne représente pas bien toutes les classes.

# Introduction aux réseaux bayésiens

Les réseaux bayésiens sont des modèles graphiques probabilistes qui représentent les relations de dépendance conditionnelle entre un ensemble de variables aléatoires. Ils combinent les principes de la théorie des graphes et de l'inférence bayésienne pour modéliser la structure causale de données complexes.

Caractéristiques clés :

- **Nœuds** : Représentent les variables aléatoires dans le domaine étudié.
- **Arcs** : Indiquent les dépendances conditionnelles entre les variables.
- **Tableaux de probabilité conditionnelle (CPT)** : Quantifient les relations entre les variables.

Les réseaux bayésiens permettent de réaliser des inférences sur des variables inobservées, de prédire les conséquences de interventions et d'apprendre la structure à partir des données.



# Inférence dans les réseaux bayésiens

L'inférence dans les réseaux bayésiens consiste à calculer la distribution de probabilité d'un ensemble de variables d'intérêt, étant donné certaines observations. Ce processus est fondamental pour la prise de décision et la prédiction dans les modèles bayésiens.

Méthodes d'inférence :

- **Inférence exacte** : Utilise des algorithmes comme l'élimination de variables ou le message passing (algorithme de Junction Tree) pour calculer précisément les distributions de probabilité.
- **Inférence approximative** : Emploie des méthodes comme le sampling de Monte Carlo et l'inférence variationnelle lorsque l'inférence exacte est impraticable due à la complexité du réseau.

L'objectif est de déterminer  $P(X|E = e)$ , la probabilité de la variable  $X$  étant donné l'évidence  $E = e$ .

# Apprentissage des réseaux bayésiens

L'apprentissage dans les réseaux bayésiens peut être divisé en deux grandes catégories : l'apprentissage des paramètres et l'apprentissage de la structure.

**Apprentissage des paramètres** : Se concentre sur l'estimation des tableaux de probabilité conditionnelle (CPT) pour un réseau dont la structure est déjà connue. Utilise souvent des méthodes d'estimation maximum de vraisemblance ou bayésienne.

**Apprentissage de la structure** : Vise à découvrir la structure du réseau (c.-à-d., quels arcs doivent exister) à partir des données. Cela peut être réalisé à l'aide d'approches basées sur les scores, comme le critère d'information bayésien (BIC), ou des approches basées sur la contrainte, qui testent l'indépendance conditionnelle entre les variables.

# Apprentissage des paramètres - exemple

L'apprentissage des paramètres consiste à ajuster les valeurs des paramètres d'un modèle pour qu'il s'adapte au mieux aux données. Prenons l'exemple d'un réseau bayésien simple qui prédit la probabilité de jouer au tennis en fonction des conditions météorologiques.

## Exemple :

- La structure du modèle est déjà définie : les nœuds représentent des variables comme "Temps" (ensoleillé, nuageux) et "Jouer au tennis" (oui, non).
- L'apprentissage des paramètres ajusterait les probabilités conditionnelles, telles que  $P(\text{Jouer au tennis} | \text{Temps} = \text{ensoleillé})$ , en se basant sur des données historiques.
- Objectif : Maximiser la correspondance entre les prédictions du modèle et les résultats réels observés dans les données d'entraînement.

Cet apprentissage affinera la capacité du modèle à prédire avec précision si oui ou non une personne jouera au tennis, en fonction des conditions météorologiques observées.

# Apprentissage de la structure - exemple

L'apprentissage de la structure cherche à déterminer la meilleure manière d'organiser les relations entre les variables dans un modèle. Considérons l'exemple d'une étude visant à comprendre les facteurs influençant la santé cardiaque.

## Exemple :

- Sans connaissances préalables spécifiques, nous disposons d'un ensemble de variables : "Activité physique", "Régime alimentaire", "Poids", "Tension artérielle" et "Santé cardiaque".
- L'apprentissage de la structure peut utiliser des données d'observation pour établir des liens potentiels, par exemple, découvrir si "Poids" agit comme un médiateur entre "Régime alimentaire" et "Santé cardiaque".
- Méthodes : Analyse de corrélation, critères d'information comme BIC, ou algorithmes de recherche structurelle.

Cette approche peut révéler des insights précieux sur les relations causales et les dépendances entre les facteurs de risque et la santé cardiaque, guidant ainsi des interventions préventives plus efficaces.

# Chaînes de Markov

# Introduction aux chaînes de Markov

Les chaînes de Markov, nommées d'après le mathématicien Andreï Markov, sont un modèle mathématique pour décrire une séquence d'événements où la probabilité de chaque événement dépend uniquement de l'état atteint dans l'événement précédent.

Caractéristiques clés :

- **Propriété de Markov** : La mémoire sans état, où la probabilité de transition vers l'état suivant ne dépend que de l'état actuel.
- **États et transitions** : Définis par un ensemble d'états possibles et les probabilités de passer d'un état à un autre.
- **Matrice de transition** : Chaque cellule  $P_{ij}$  de la matrice représente la probabilité de passer de l'état  $i$  à l'état  $j$ .

# Exemple de chaîne de Markov

Considérons une chaîne de Markov simple pour modéliser les prévisions météorologiques avec deux états possibles : Ensoleillé (E) et Pluvieux (P).

**États :**

- $S = \{E, P\}$

**Matrice de transition** : La matrice de transition  $P$  représente les probabilités de passer d'un type de temps à l'autre d'un jour à l'autre.

$$P = \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix}$$

où  $P_{ij}$  représente la probabilité de passer de l'état  $i$  à l'état  $j$ . Par exemple,  $P_{EP} = 0.1$  signifie qu'il y a 10% de chance qu'un jour ensoleillé soit suivi d'un jour pluvieux.

**Dynamique :**

- Si le temps est ensoleillé aujourd'hui, il y a 90% de chance qu'il soit également ensoleillé demain.
- Si le temps est pluvieux aujourd'hui, les chances sont égales pour que le temps soit ensoleillé ou pluvieux le lendemain.

# Introduction aux chaînes de Markov à états cachés

Les chaînes de Markov à états cachés (HMM) sont des modèles statistiques où l'on suppose que le système à modéliser est un processus de Markov avec des états non observés (cachés).

Caractéristiques clés des HMM :

- **États cachés** : Le modèle a des états que l'on ne peut pas observer directement.
- **Observations** : Chaque état caché produit une observation, qui peut être vue ou mesurée.
- **Transitions** : Les états cachés suivent une chaîne de Markov, avec des probabilités de transition spécifiques.

Les HMM sont largement utilisés pour les séquences temporelles où les événements sont liés, tels que la reconnaissance de la parole, la bioinformatique et la finance.



# Composants d'un HMM

Un modèle de Markov caché est défini par trois séries de paramètres :

- **Matrice de transition**  $A$  : Probabilités de passer d'un état caché à un autre.
- **Matrice d'observation**  $B$  : Probabilités de l'observation donnée l'état caché.
- **Distribution initiale**  $\pi$  : Probabilités des états cachés initiaux.

Notation :

- $A = [a_{ij}]$  où  $a_{ij}$  est la probabilité de transition de l'état  $i$  à l'état  $j$ .
- $B = [b_{ij}]$  où  $b_{ij}$  est la probabilité d'observer l'observation  $j$  dans l'état  $i$ .
- $\pi = [\pi_i]$  où  $\pi_i$  est la probabilité initiale de l'état  $i$ .

Ces matrices forment le cœur d'un HMM et sont utilisées pour le décodage, l'apprentissage et l'évaluation de séquences.

# Algorithmes pour HMM

Les algorithmes clés utilisés dans les HMM sont :

- **Algorithme de Viterbi** : Trouve la séquence la plus probable d'états cachés donnée une séquence d'observations.
- **Algorithme Forward** : Calcule la probabilité de la séquence d'observations donnée jusqu'à un certain point dans le temps.
- **Algorithme Backward** : Calcule la probabilité de la séquence d'observations à venir, en partant d'un état caché.

Ces algorithmes permettent de résoudre les problèmes fondamentaux des HMM : l'évaluation, le décodage et l'apprentissage.

# HMM pour la reconnaissance de la parole - Partie 1

Les Modèles de Markov Cachés (HMM) sont au cœur des systèmes de reconnaissance de la parole. Ils permettent de modéliser les séquences sonores temporelles en prenant en compte les variations temporelles et les incertitudes.

## Fonctionnement d'un HMM en reconnaissance de la parole :

- Les états cachés représentent les phonèmes ou les unités de son plus petites qui composent la parole.
- Les observations sont des vecteurs de caractéristiques acoustiques extraites des trames sonores.
- Les transitions d'états modélisent la séquence et la durée des phonèmes dans le langage naturel.

Les HMMs utilisent ces composantes pour calculer la séquence de phonèmes la plus probable qui correspond à une séquence d'observations acoustiques, permettant ainsi de transcrire la parole en texte.

# HMM pour la reconnaissance de la parole - Partie 2

## Applications spécifiques des HMM en reconnaissance de la parole :

- **Modélisation des mots** : Chaque mot peut être représenté par un HMM, avec des transitions entre les états représentant les phonèmes.
- **Coarticulation** : Gérer l'influence d'un phonème sur un autre, qui est crucial pour la reconnaissance précise.
- **Adaptation au locuteur** : Personnaliser les modèles à la voix et à l'accent d'un utilisateur spécifique.

Les HMM sont particulièrement puissants pour la reconnaissance de la parole en raison de leur capacité à capturer les dynamiques temporelles et à gérer la variabilité inhérente à la parole humaine. Ils sont utilisés dans les assistants vocaux, les systèmes de dictée automatique, et les interfaces homme-machine vocales.

# Applications des HMM en Marketing

Les Modèles de Markov Cachés sont utilisés en marketing pour analyser les parcours client et prédire les comportements d'achat. Ils permettent de comprendre les séquences de décisions cachées derrière les interactions observables des clients.

## Utilisations des HMM en marketing :

- **Segmentation du comportement client** : Identification de séquences de comportement similaires, permettant une segmentation précise des clients.
- **Prédiction de la conversion** : Estimation de la probabilité qu'un client passe à l'achat, basée sur son parcours d'interaction avec divers points de contact marketing.
- **Optimisation de la chaîne de marketing** : Allocation efficace des ressources en analysant les transitions d'états qui maximisent les conversions ou la fidélité.

# Deep learning

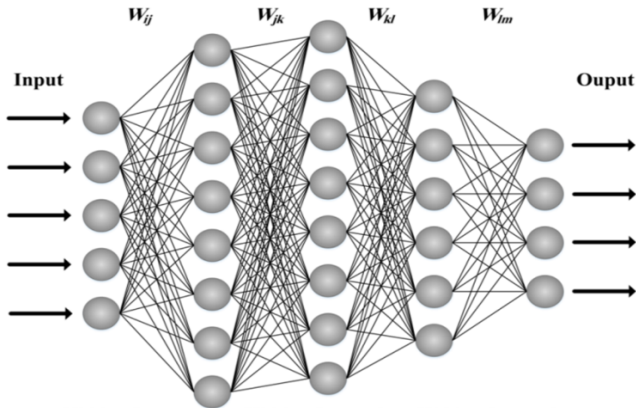
# Introduction aux réseaux de neurones

**Définition** : Les réseaux de neurones sont des modèles computationnels inspirés par le fonctionnement des neurones dans le cerveau humain. Ils sont capables d'apprendre des tâches complexes en modélisant des relations non linéaires entre les entrées et les sorties.

## Caractéristiques :

- **Extraction automatique des features** : Capacité d'adaptation et d'extraction des features à partir des données sans programmation explicite.
- **Modélisation non linéaire** : Aptitude à capturer des relations complexes dans les données.
- **Flexibilité** : Applicables à un large éventail de tâches et de typologies de données (images, langage naturel, données graphiques, etc.).

# Illustration d'un réseau de neurones classique





# Le neurone artificiel

**Modèle Mathématique** : Chaque neurone artificiel effectue une somme pondérée de ses entrées, ajoute un biais, et passe le résultat à travers une fonction d'activation pour obtenir la sortie.

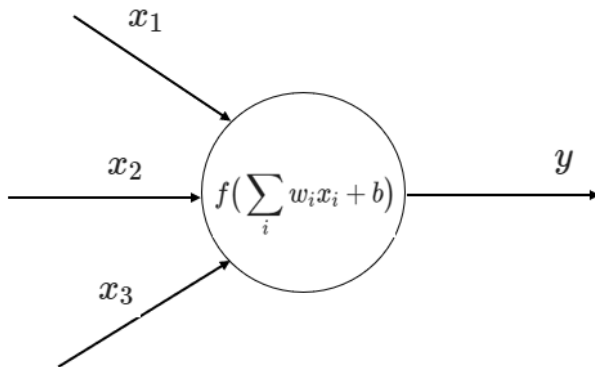
$$z_i = \sum_{j=1}^m w_{ij}x_j + b_i$$

$$a_i = f(z_i)$$

où :

- $x_j$  représente l'entrée du neurone,
- $w_{ij}$  est le poids associé à l'entrée  $x_j$ ,
- $b_i$  est le biais du neurone,
- $f$  est la fonction d'activation,
- $z_i$  est le potentiel d'action pré-synaptique,
- $a_i$  est la sortie activée du neurone.

# Illustration d'un neurone artificiel



# Fonctions d'activation

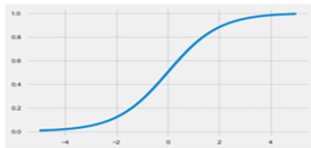
Les fonctions d'activation permettent aux modèles d'apprendre des relations plus complexes en capturant les non-linéarités dans les données.

- **Sigmoïde** :  $\sigma(z) = \frac{1}{1+e^{-z}}$ , plage de sortie (0, 1), utilisée pour la probabilité dans la classification binaire.
- **Tangente Hyperbolique (tanh)** :  $\tanh(z)$ , plage de sortie (-1, 1), version centrée et normalisée de la sigmoïde.
- **ReLU (Unité Linéaire Rectifiée)** :  $f(z) = \max(0, z)$ , non saturante, favorise la convergence rapide et permet d'éviter le problème de disparition des gradients.
- **Leaky ReLU** :  $f(z) = \max(\alpha z, z)$ , variante de ReLU qui permet un petit gradient lorsque  $z < 0$ .
- **Softmax** : Utilisée pour la couche de sortie des problèmes de classification multi-classes.

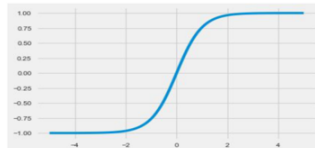
$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

# Illustration des fonctions d'activation

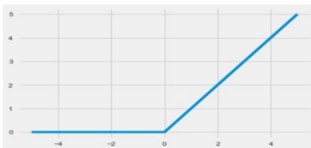
Sigmoïde



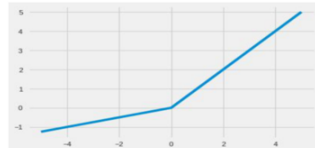
Tanh



ReLU



ReLU paramétrique



# Entraînement d'un réseau de neurones artificiel

**Principe d'Entraînement** : L'entraînement d'un réseau de neurones consiste à ajuster ses poids pour minimiser une fonction de coût qui mesure l'erreur entre les prédictions et les vraies valeurs.

## Descente de gradient stochastique (SGD) :

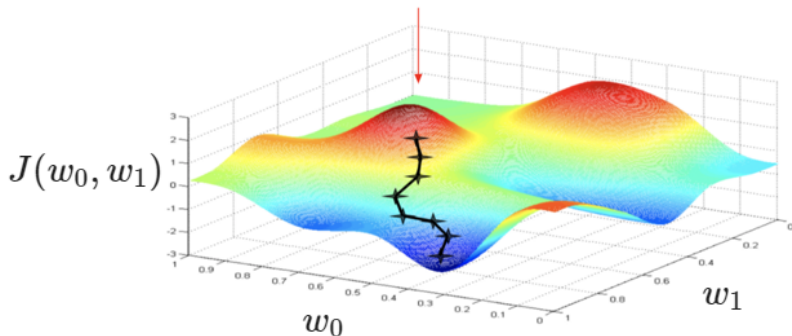
- Méthode d'optimisation utilisée pour mettre à jour les poids du réseau de manière itérative.
- À chaque itération, un sous-ensemble (batch) de données est utilisé pour calculer le gradient de la fonction de coût.
- Les poids sont mis à jour dans la direction opposée du gradient pour réduire l'erreur.

$$w_{new} = w_{old} - \eta \cdot \nabla_w J(w)$$

où :

- $w_{old}$  et  $w_{new}$  sont les valeurs des poids avant et après la mise à jour,
- $\eta$  est le taux d'apprentissage,
- $\nabla_w J(w)$  est le gradient de la fonction de coût par rapport aux poids.

# Illustration graphique de la SGD



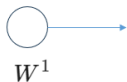
# Entraînement des réseaux de neurones profonds

Dans les réseaux de neurones profonds, l'erreur calculée à la sortie du réseau dépend indirectement des poids des couches cachées. Ce lien indirect rend difficile de savoir comment ajuster ces poids pour réduire l'erreur.

## Implications pour l'Entraînement :

- **Propagation de l'erreur** : Sans un mécanisme pour propager l'erreur de la sortie vers les couches antérieures, il est impossible de déterminer l'impact de chaque poids sur l'erreur finale.
- **Complexité de l'ajustement des Poids** : Chaque poids dans les couches cachées affecte l'erreur de sortie de manière complexe, nécessitant une méthode précise pour leur ajustement.

Couche 1



...

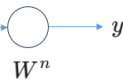
Couche n-2



Couche n-1



Couche n



# Rétropropagation du gradient

Il est facile de calculer les gradients correspondant à la dernière couche  $\frac{\partial J}{\partial W^{(n)}}$  car l'erreur  $J$  dépend immédiatement de  $W^{(n)}$ .



Examinons maintenant le cas de la couche  $n - 1$  :

$$\frac{\partial J}{\partial W^{(n-1)}} = \frac{\partial J}{\partial W^{(n)}} \frac{\partial W^{(n)}}{\partial O_n} \frac{\partial O_n}{\partial W^{(n-1)}}$$

Or

$$\frac{\partial O_n}{\partial W^{(n-1)}} = \frac{\partial O_n}{\partial O_{n-1}} \frac{\partial O_{n-1}}{\partial W^{(n-1)}}$$



# Surapprentissage dans les réseaux de neurones

Le surapprentissage (overfitting) se produit lorsqu'un réseau de neurones apprend trop bien les détails et le bruit des données d'entraînement, au détriment de sa capacité à généraliser sur de nouvelles données.

Le surapprentissage dans les réseaux de neurones peut se produire pour différentes raisons :

- Complexité excessive du modèle
- Manque de diversité dans les données d'entraînement
- Entraînement prolongé

## Stratégies pour atténuer le surapprentissage :

- **Régularisation** : Techniques de régularisation telles que L1/L2, Dropout, pour limiter la complexité du modèle.
- **Dropout** : "Eteindre" un ensemble de neurones choisis aléatoirement pendant l'entraînement.
- **Early Stopping** : Arrêt de l'entraînement lorsque la performance sur un ensemble de validation cesse de s'améliorer.
- **Augmentation de Données** : Augmenter la variété des données d'entraînement pour améliorer la robustesse du modèle.

# Mise en production des modèles de machine learning

# Pipeline traditionnel d'un projet de machine learning

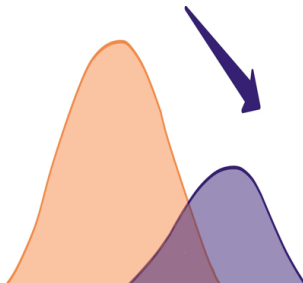


**En réalité, ce processus est itératif.**

# Data drift

Le data drift est le phénomène de changement dans la distribution des données. Il apparaît d'une manière fréquente dans les modèles mis en production. Il peut être dû à plusieurs facteurs :

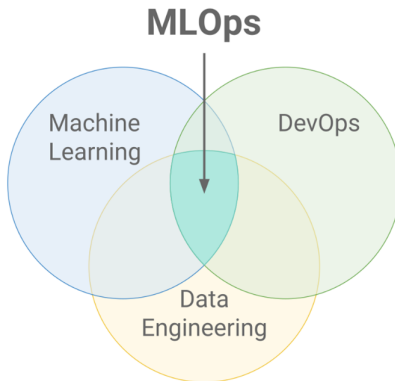
- Différence entre les données de test et les données en production.
- Changement de l'environnement (avant et après le Covid par exemple).
- Changement dans le comportement des utilisateurs ou clients.
- Effets de l'algorithme lui-même sur les utilisateurs.



# MLOps

## Objectifs du MLOps :

- Alignement entre les équipes data scientists, data engineer, etc.
- Reproductibilité et maintenance du code.
- Adaptation aux changements pouvant affecter la distribution des données au fil du temps.
- Etc.



# Outils de MLOps



Google Cloud Platform



vertex.ai



Amazon  
SageMaker



databricks



# Développement en environnement distribué

Les spécificités liées au développement d'un modèle en environnement distribué incluent :

## **Caractéristiques de l'environnement distribué :**

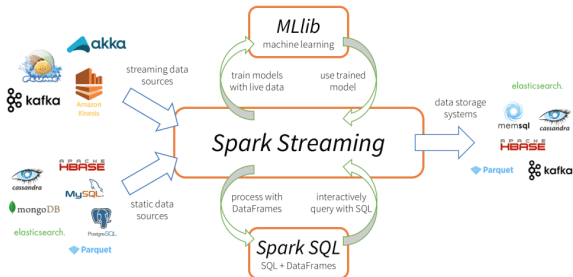
- Gestion de données volumineuses réparties sur plusieurs machines.
- Parallélisation des calculs pour optimiser la vitesse de traitement.
- Synchronisation des opérations pour assurer la cohérence des résultats.

# Déploiement Big Data avec Spark et MLlib

Apache Spark et sa bibliothèque MLlib offrent un écosystème puissant pour déployer des modèles de Machine Learning à grande échelle.

## Avantages de Spark et MLlib :

- Traitement rapide en mémoire des grandes données.
- API haut niveau pour la construction de pipelines de Machine Learning.
- Intégration aisée avec les systèmes de stockage de données Big Data.





# Bonnes pratiques de mise en production

- Intégration continue et déploiement continu (CI/CD).
- Validation de la qualité du code.
- Monitoring de la qualité des données en production.
- Monitoring des performances de l'algorithme en production.
- Déploiement de modèles simples quand cela est possible.
- Optimisation et veille sur les ressources Cloud.

# Bibliographie

- Hastie, T., Tibshirani, R., Friedman, J. H., Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: springer.
- Russell, S. J. (2010). Artificial intelligence a modern approach. Pearson Education, Inc..
- Ng, A. (2000). CS229 Lecture notes. CS229 Lecture notes, 1(1), 1-3.
- Azencott, C. A. (2019). Introduction au machine learning. Dunod.
- Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep learning. MIT press.
- Vapnik, V. (1999). The nature of statistical learning theory. Springer science business media.
- Cortes, C., Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

# Merci de votre attention

redha\_moulla@yahoo.fr