

Machine learning

Redha Moulla

14 - 18 avril 2025

Plan de la formation

- Qu'est-ce que l'intelligence artificielle ?
- Machine learning supervisé
- Machine learning non supervisé
- Qualité des données
- Introduction au deep learning
- Introduction aux Large Language Models (LLMs)

Qu'est-ce que l'intelligence artificielle ?

Définition littérale de l'intelligence artificielle

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 :

- ① **Raisonnement** : La capacité à résoudre des problèmes et à faire des déductions logiques.
- ② **Apprentissage** : L'aptitude à acquérir de nouvelles connaissances et à s'améliorer grâce à l'expérience.
- ③ **Perception** : La compétence pour reconnaître et interpréter les stimuli sensoriels.
- ④ **Compréhension** : L'habileté à saisir le sens et l'importance de divers concepts et situations.
- ⑤ **Mémorisation** : La faculté de stocker et de rappeler des informations.
- ⑥ **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 ?

Citation d'Aristote

"Si chaque instrument était capable, sur une simple injonction, ou même pressentant ce qu'on va lui demander, d'accomplir le travail qui lui est propre, comme on le raconte des statues de Dédale ou des trépieds d'Héphaïstos, lesquels, dit le poète, : "Se rendaient d'eux-mêmes à l'assemblée des dieux", si, de la même manière, les navettes tissaient d'elles-mêmes, et les plectres pinçaient tout seuls la cithare, alors, ni les chefs d'artisans n'auraient besoin d'ouvriers, ni les maîtres d'esclaves."

— Aristote

Conférence de Dartmouth ?

Articles

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 E. Shannon

The 1956 Dartmouth summer research project on artificial intelligence was organized by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon. The report consists of ten typescript pages plus a title page. Copies of the typescript are bound in a looseleaf binder. The report is available at Stanford University. The first 5 papers state the proposal and give the general goals of the project and interests of the four who proposed the study. In the interest of brevity, this article summarizes the main points of the proposals and bibliographical statements of the proposals.

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 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.

1. Automatic Computers

If a machine can do a job, then an automatic calculator can be programmed to simulate the machine. It may be argued that present computers may be insufficient to simulate many of the higher functions of the mind. This may be true, but it is not because of 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 language consists of words according to rules of reasoning and rules of conjecture. From this point of view, learning 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

Le Test de Turing

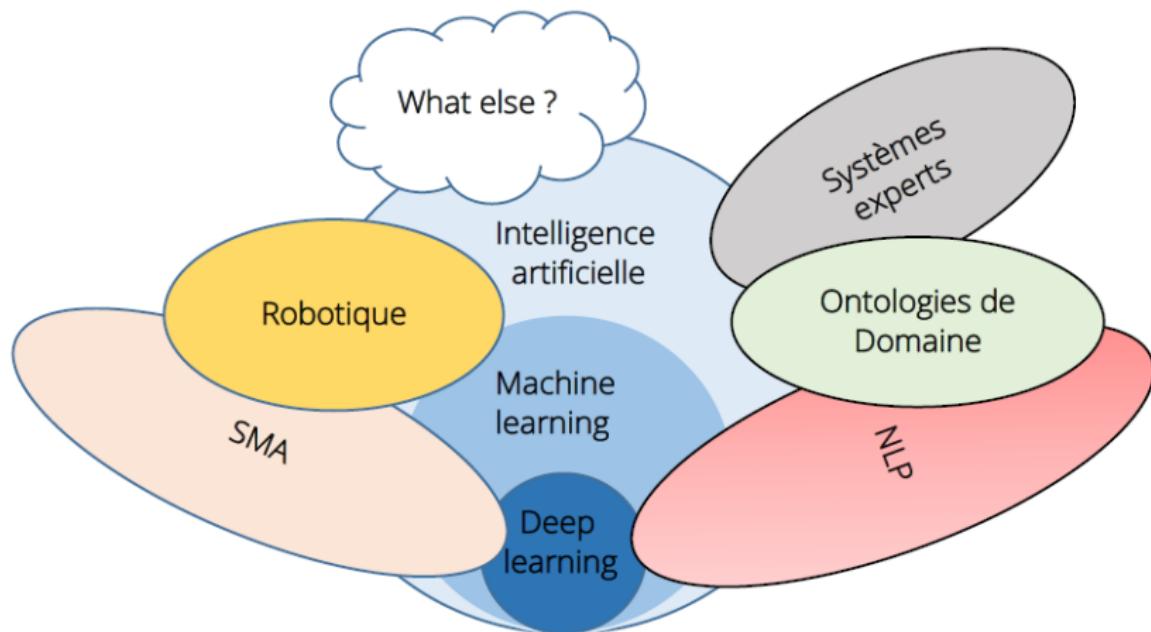
Le Test de Turing, développé par Alan Turing en 1950, est une tentative de mesurer l'intelligence d'une machine, plus précisément de la faculté d'une machine à penser. Cette dernière n'étant pas si évidente à mesurer, le test substitue finalement à la faculté de penser celle de traiter le langage naturel comme un humain.

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 vérifique, mais plutôt la capacité de reproduire le comportement humain.

Définition pragmatique de l'intelligence artificielle

Il s'agit d'un ensemble de techniques qui permettent à la machine d'accomplir des tâches qui requièrent traditionnellement une intelligence humaine.



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

Intelligence artificielle connexionniste

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

- Probabiliste
- Apprentissage machine
- Orientée données

Machine learning

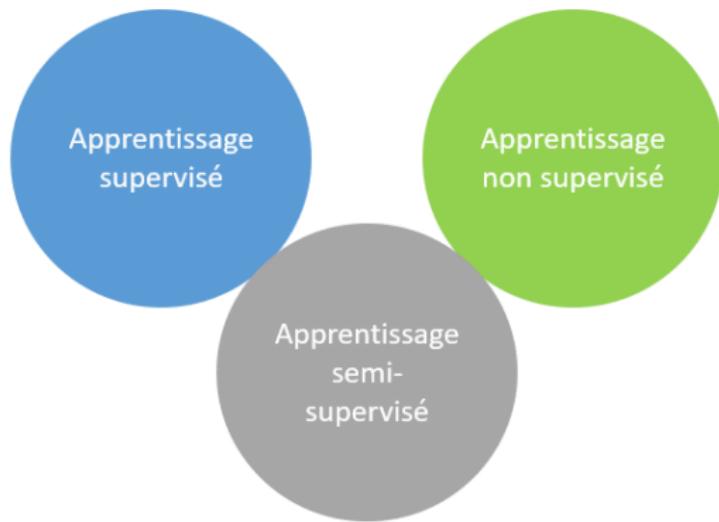
Définition de l'apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui consiste à doter les machines de la capacité d'apprendre à partir de données sans que celles-ci ne soient explicitement programmées pour exécuter des tâches spécifiques.

Le machine learning 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.

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.

Classification

Exemple de classification : credit scoring

Âge	Revenu annuel (k€)	Historique de crédit	Nombre de cartes de crédit	Niveau d'éducation	Propriétaire immobilier (Oui/Non)	Label (y)
30	50	Bon	2	Licence	Oui	Accepté
45	80	Moyen	3	Master	Oui	Accepté
22	20	Mauvais	1	Bac	Non	Refusé
35	60	Bon	4	Licence	Oui	Accepté
40	70	Moyen	2	Bac+2	Non	Refusé

Régression

Exemple de régression : prédition des prix des logements

Surface (m ²)	Nombre de chambres	Distance du centre-ville (km)	Année de construction	Quartier (Score 1-10)	Prix (k€) (y)
80	3	5	2010	8	300
120	4	10	2005	7	450
60	2	2	2020	9	200
150	5	15	1995	6	600
100	3	8	2015	7	400

L'apprentissage non supervisé

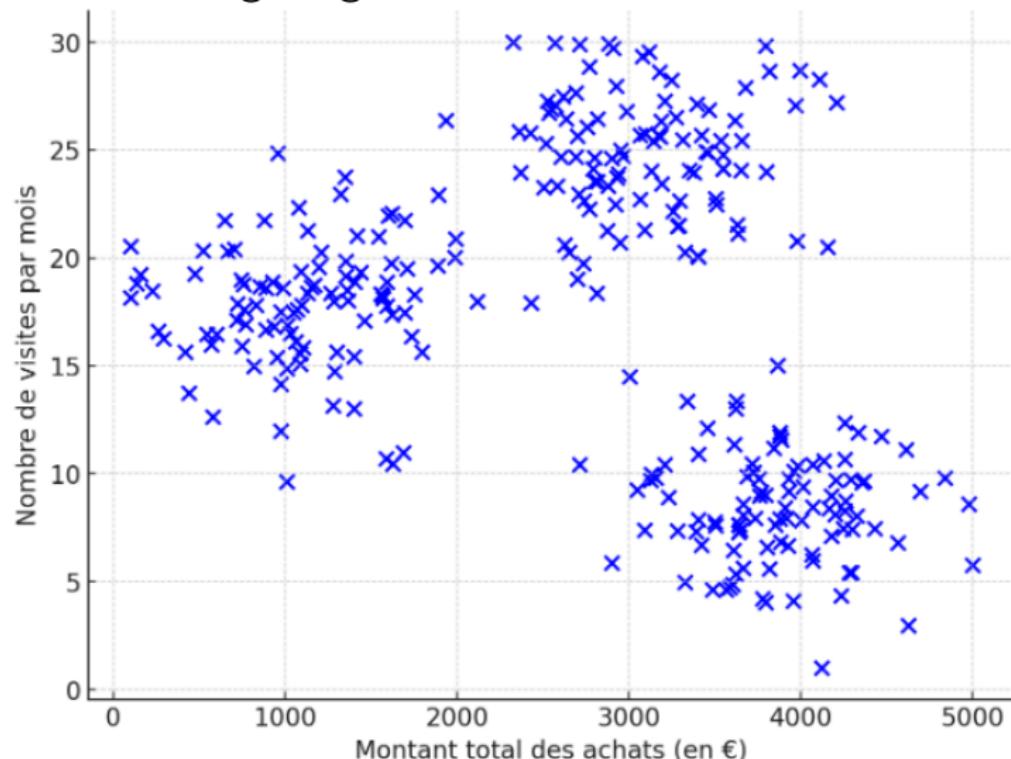
L'apprentissage non supervisé se réfère à l'utilisation de modèles d'apprentissage automatique pour identifier des patterns et des structures dans des données qui ne sont pas étiquetées.

Principales typologies de l'apprentissage non supervisé :

- **Clustering** : Regroupement de points de données similaires ensemble.
Exemple : segmentation de marché, regroupement social.
- **Détection d'anomalies** : Détecter des observations dont les caractéristiques sont inhabituelles par rapport à la majorité.

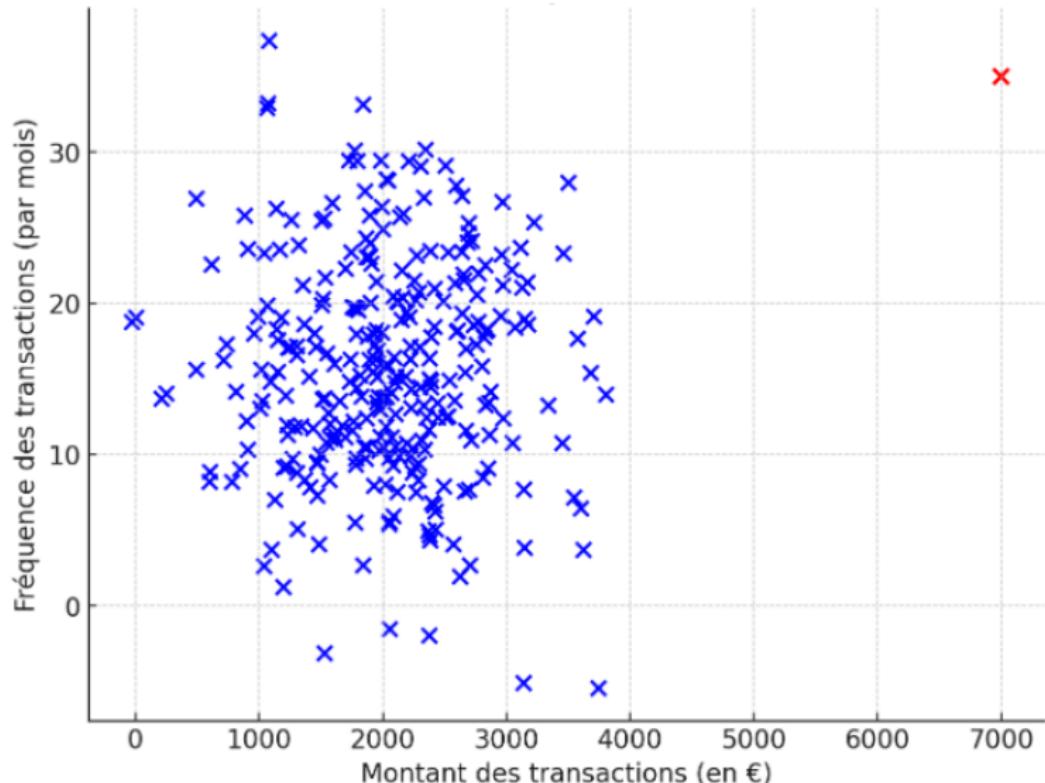
Clustering

Exemple de clustering : segmentation clients



Détection d'anomalies

Exemple de détection d'anomalies : fraude bancaire



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 :

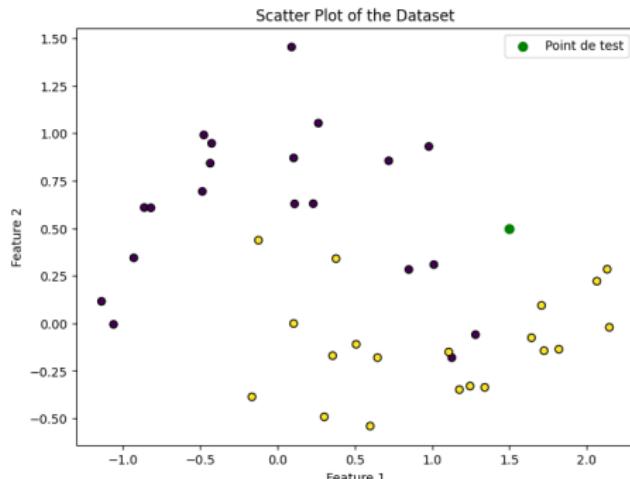
- Développement de systèmes de recommandation plus performants.
- Traitement de langage naturel et analyse de sentiment lorsque les annotations complètes ne sont pas disponibles.

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

Apprentissage supervisé

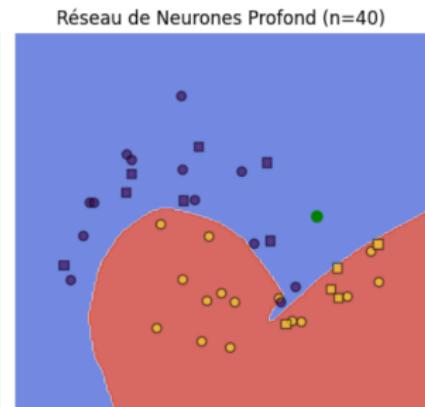
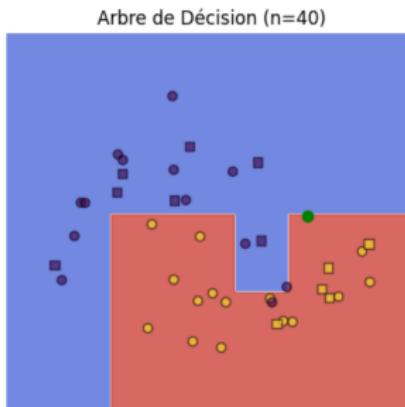
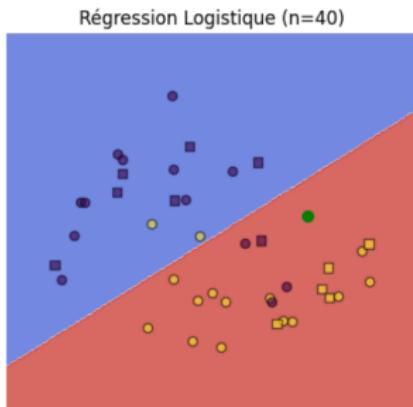
L'apprentissage supervisé comme un problème d'induction

- **Définition** : L'apprentissage supervisé consiste à apprendre une fonction f qui mappe les entrées X aux sorties y , à partir d'un ensemble d'exemples d'entraînement (X, y) .
- **Induction** : Le modèle induit une règle générale à partir de données particulières, dans le but de généraliser à de nouvelles instances.
- **Problème de généralisation** : Comment garantir que le modèle apprend une règle qui s'applique à de nouvelles données ?



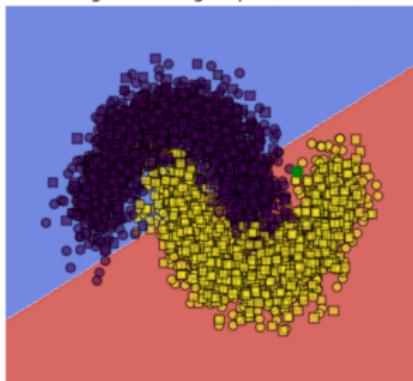
Une indétermination intrinsèque pour le choix du modèle

Il y a une infinité de manières d'induire un modèle à partir d'un échantillon de données d'entraînement.

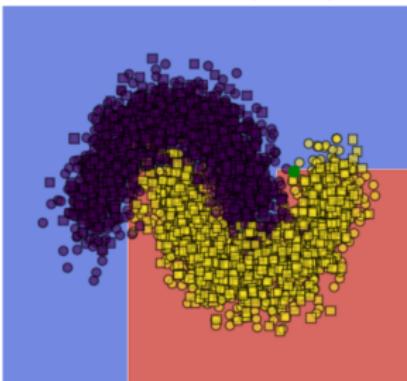


Sous-apprentissage et surapprentissage sur les données d'entraînement

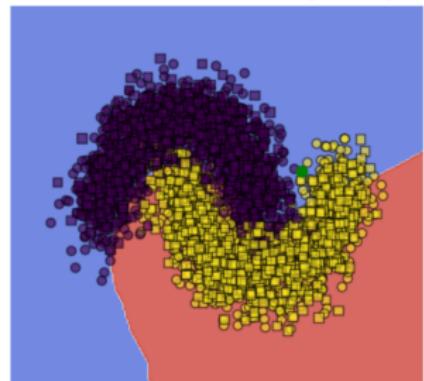
Régression Logistique (n=3000)



Arbre de Décision (n=3000)



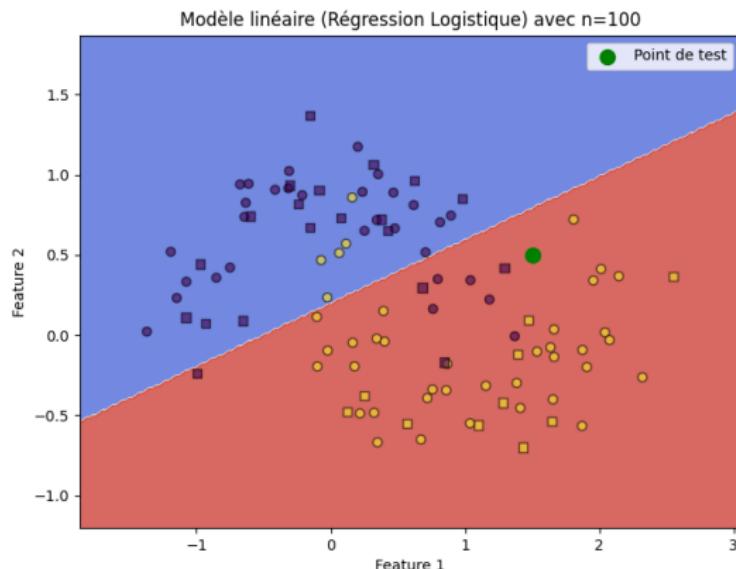
Réseau de Neurones Profond (n=3000)



Sous-apprentissage

Definition

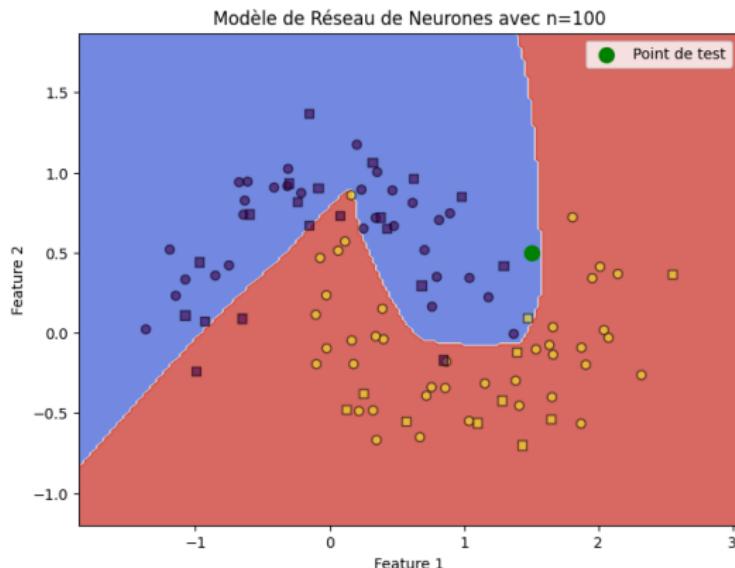
On dit qu'un modèle de machine learning est en régime de sous-apprentissage (underfitting) lorsqu'il n'arrive pas à capturer la complexité (l'information) présente dans le jeu de données d'entraînement.



Sur-apprentissage

Definition

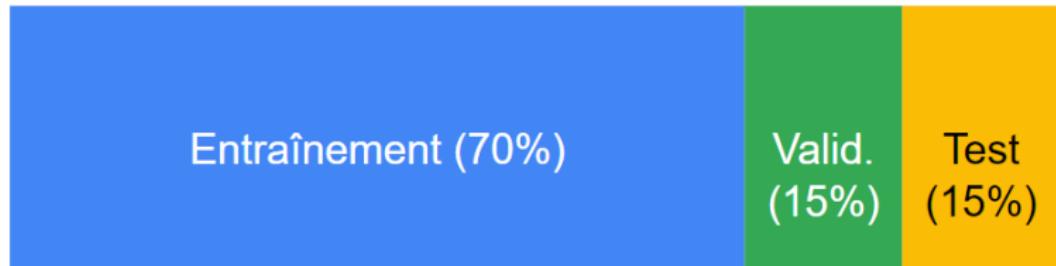
On dit qu'un modèle de machine learning est en régime de sur-apprentissage (overfitting) lorsqu'il n'arrive pas à généraliser à des données non encore observées, i.e. lorsqu'il est trop adapté aux données d'entraînement.



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.



Exemple : sélection de modèle

Modèle	Précision Entraînement	Précision Validation
Régression Logistique	0.90	0.88
Arbre de Décision	0.95	0.92
Réseau de Neurones Profond	1	0.90

Table: Comparaison des précisions des modèles sur les jeux d'entraînement et de validation

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 2/2

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.

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

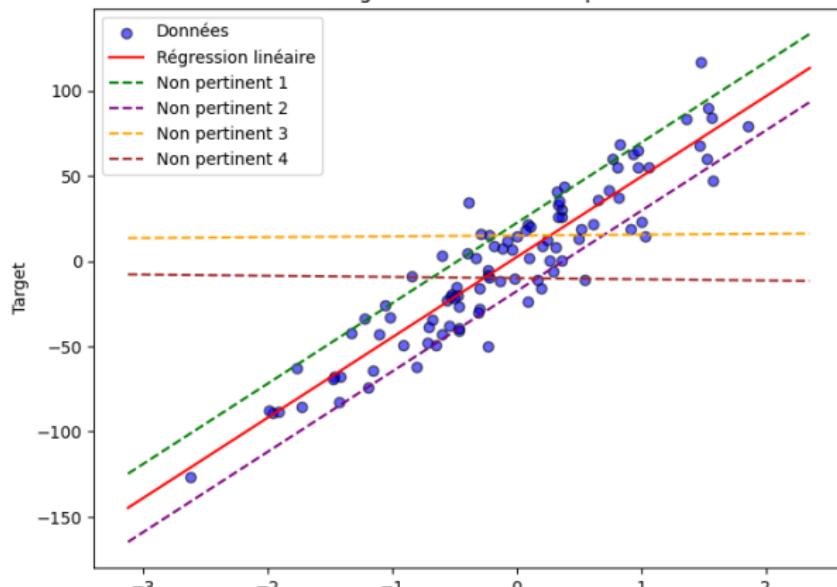
Modèles classiques de machine learning

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.

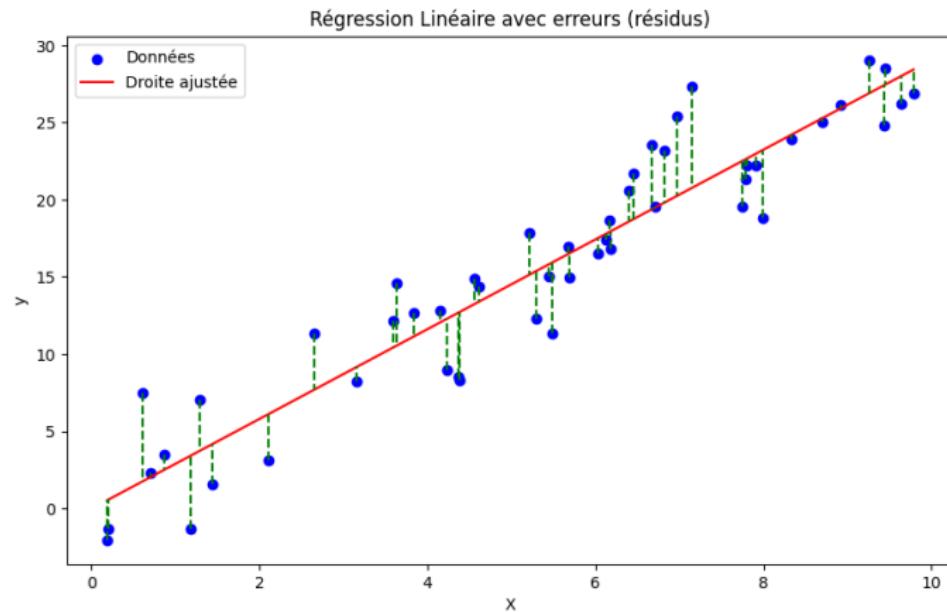
$$\hat{y} = \beta_0 + \beta_1 x$$

Illustration de la régression linéaire avec plusieurs modèles



Minimisation du risque empirique 1/2

L'erreur de prédiction pour la i ième observation est : $e_i = y_i - \hat{y}_i$. où $\hat{y}_i = \beta_0 + \beta_1 x_i$.



Minimisation du risque empirique 2/2

Déterminer un modèle de régression linéaire simple revient à minimiser les erreurs de prédiction (risque empirique) :

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

Autrement dit, chercher les coefficients β_j qui minimisent le risque empirique :

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

Les expressions des β sont obtenues en résolvant les équations normales :

$$\beta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

où \bar{x} et \bar{y} sont les moyennes des x_i et y_i , respectivement.

Exemple : prédire le prix d'un logement en fonction de la surface 1/2

Soit un dataset de 100 points où x représente la surface (en m²) et y le prix des logements (en €). Les 4 premières lignes sont :

x (Surface m ²)	y (Prix en €)
40	120 000
65	200 000
80	250 000
55	160 000
:	:

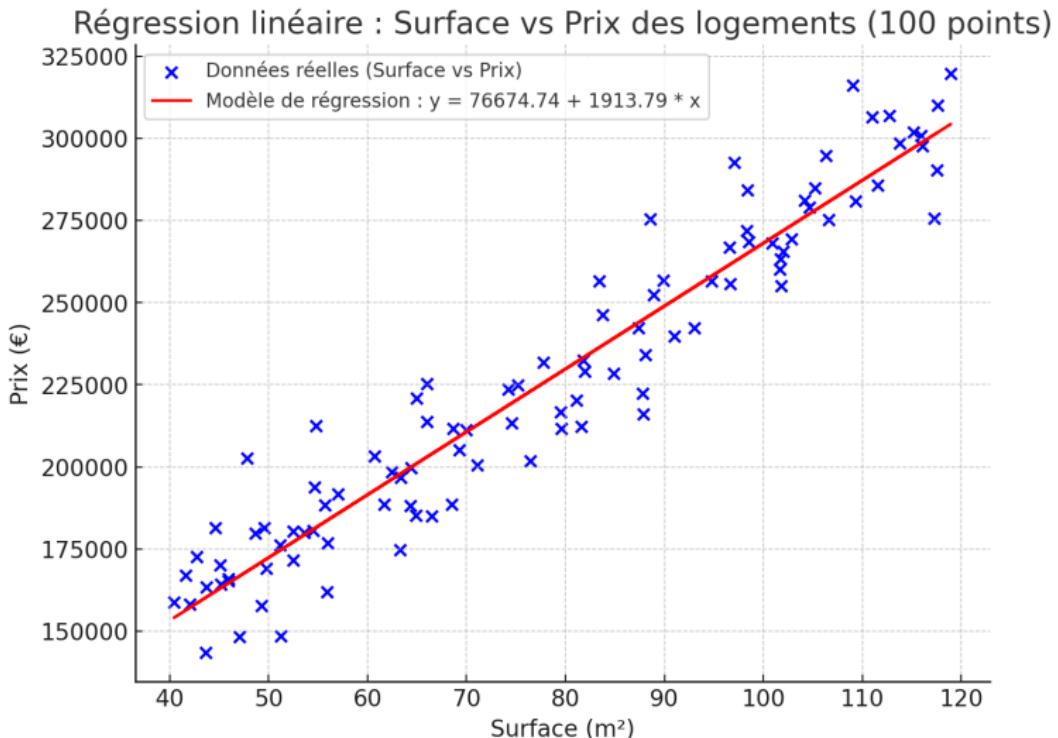
Les coefficients β_0 et β_1 sont calculés à partir des formules suivantes :

$$\beta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{1234567}{56789} = 2173.15$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x} = 200000 - 2173.15 \times 60 = 69410.5$$

L'équation de la droite de régression obtenue est donc :

Exemple : prédire le prix d'un logement en fonction de la surface 2/2



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 :

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

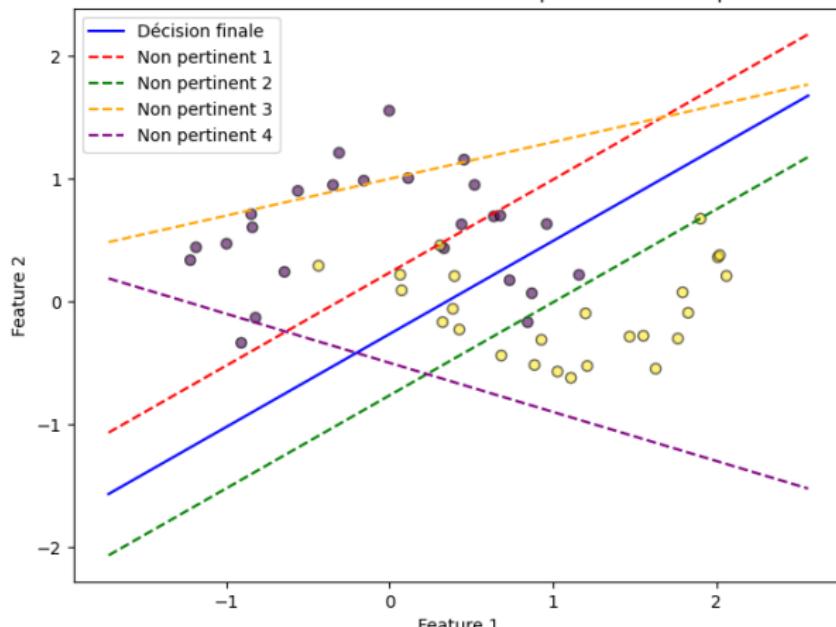
Les coefficients $\beta_0, \beta_1, \dots, \beta_p$ sont déterminés par la méthode des moindres carrés :

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

Principe de la régression logistique

- **Définition :** La régression logistique est un algorithme de classification linéaire utilisé pour prédire la probabilité qu'une observation appartienne à une classe donnée.

Illustration de la surface de décision avec plusieurs droites possibles



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 :

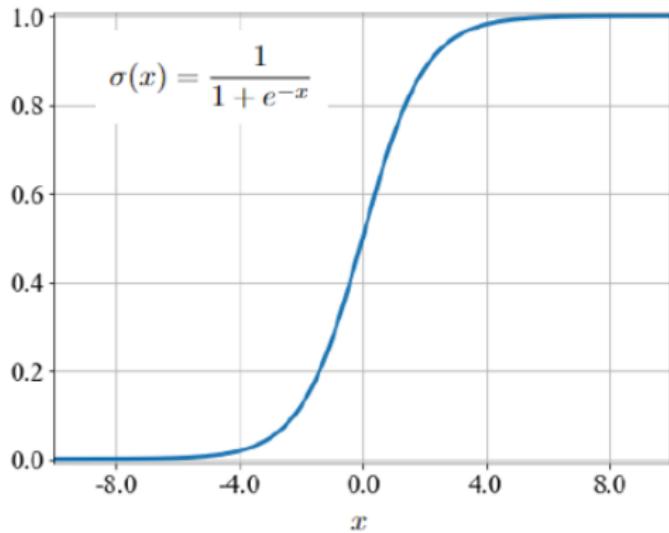
- **Variable dépendante** : On cherche la probabilité que la variable dépendante (y) appartienne à une classe (0 ou 1, vrai ou faux, succès ou échec). Autrement dit, on cherche à modéliser $P(y = 1)$ en fonction des variables dépendantes (explicatives) x .
- **Odds ratio** : Plus concrètement, on cherche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

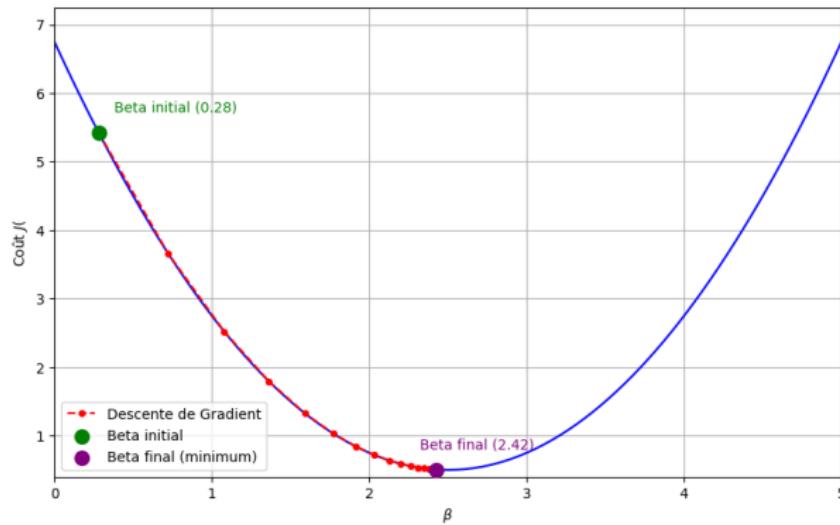
$$p(\mathbf{x}) = \frac{1}{1 + e^{-(\beta^T \mathbf{x})}}$$



Calcul des coefficients de la régression logistique

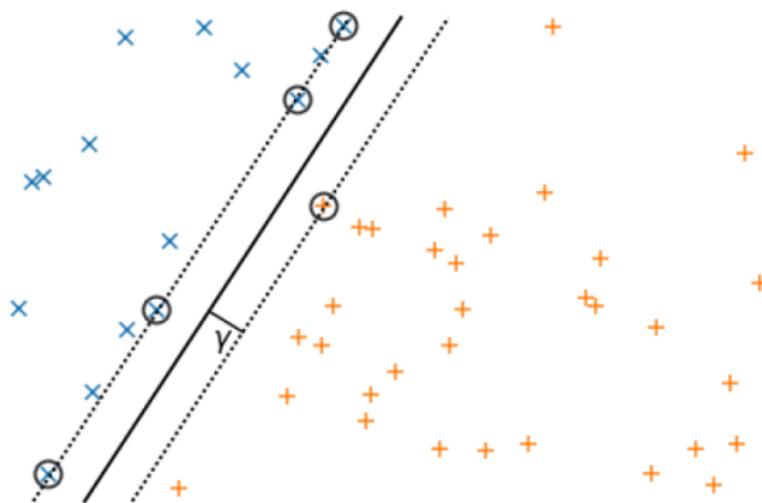
Les coefficients de la régression logistique peuvent être calculés en minimisant le risque empirique par rapport à une fonction de coût sous forme d'entropie croisée :

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



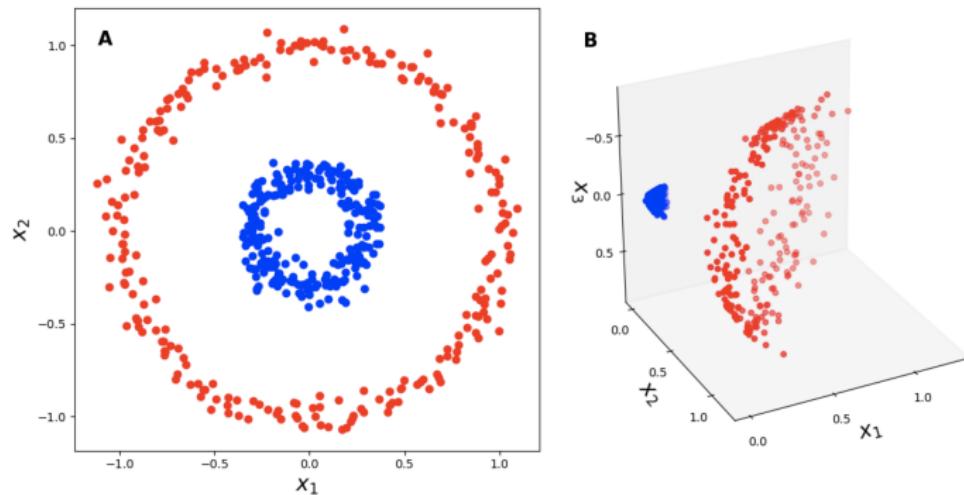
Machines à vecteurs de support (SVM)

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



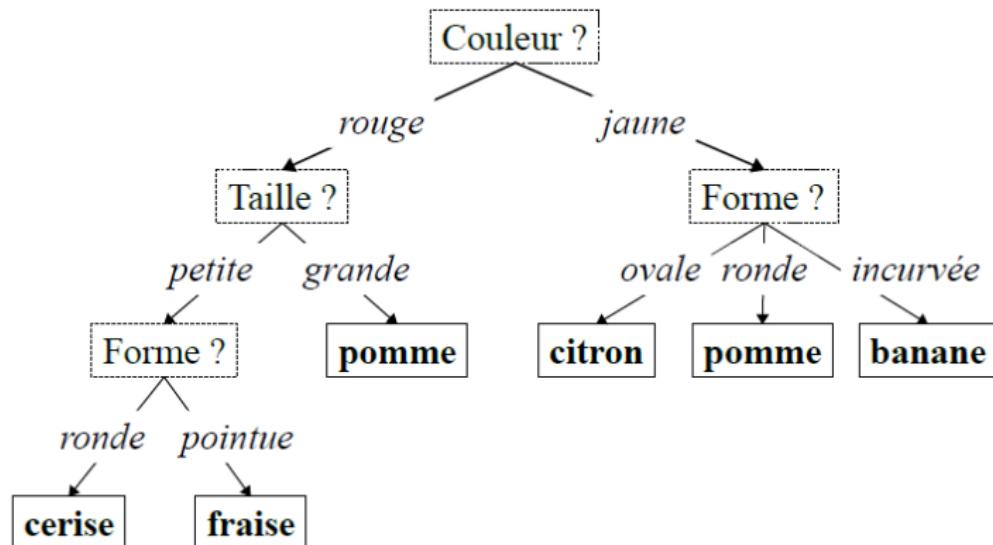
SVM à noyau

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.



Arbres de décision

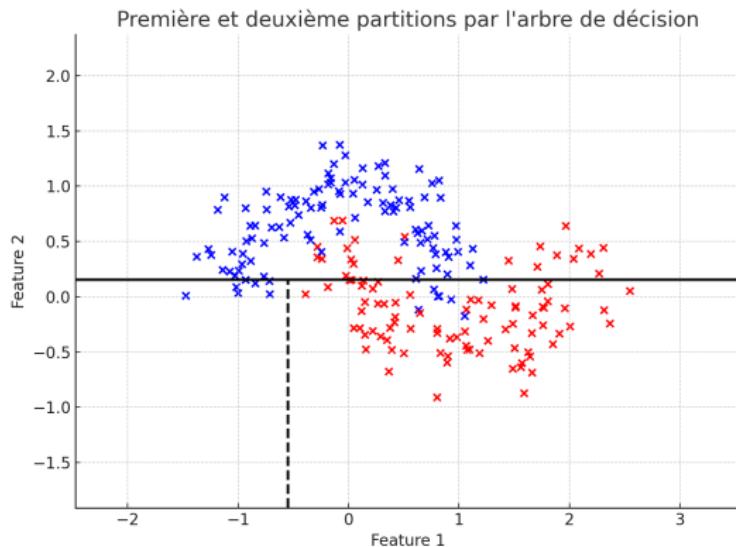
Les arbres de décisions sont des modèles dont le processus de décision est hiérarchique et prend la forme d'un arbre.



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 .



Critères d'optimisation

Le critère d'optimisation dépend de la tâche en question.

- **Classification**

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

- **Régression**

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

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 partinente 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).

Régularisation de modèle

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 *bêta* peuvent alors être calculés en minimisant la *RSS*.

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

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 :

$$X = UDV^T$$

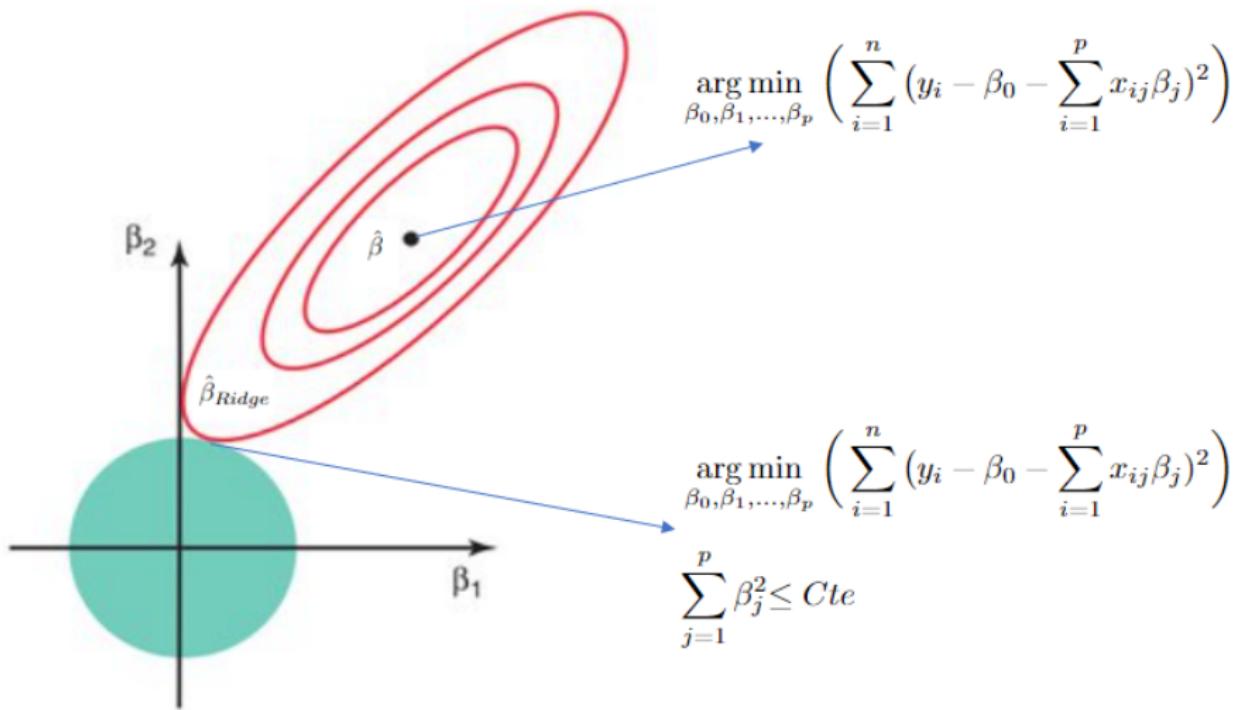
où U et V sont des matrices orthogonales de dimensions respectives $n \times p$ et $p \times p$.

On peut montrer que :

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

On peut noter que le paramètre de régularisation λ 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 dimensionnalité du problème pour avoir un modèle avec peu de variable (plus simple et plus explicable).

La régression Lasso est formalisée de la manière suivante :

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

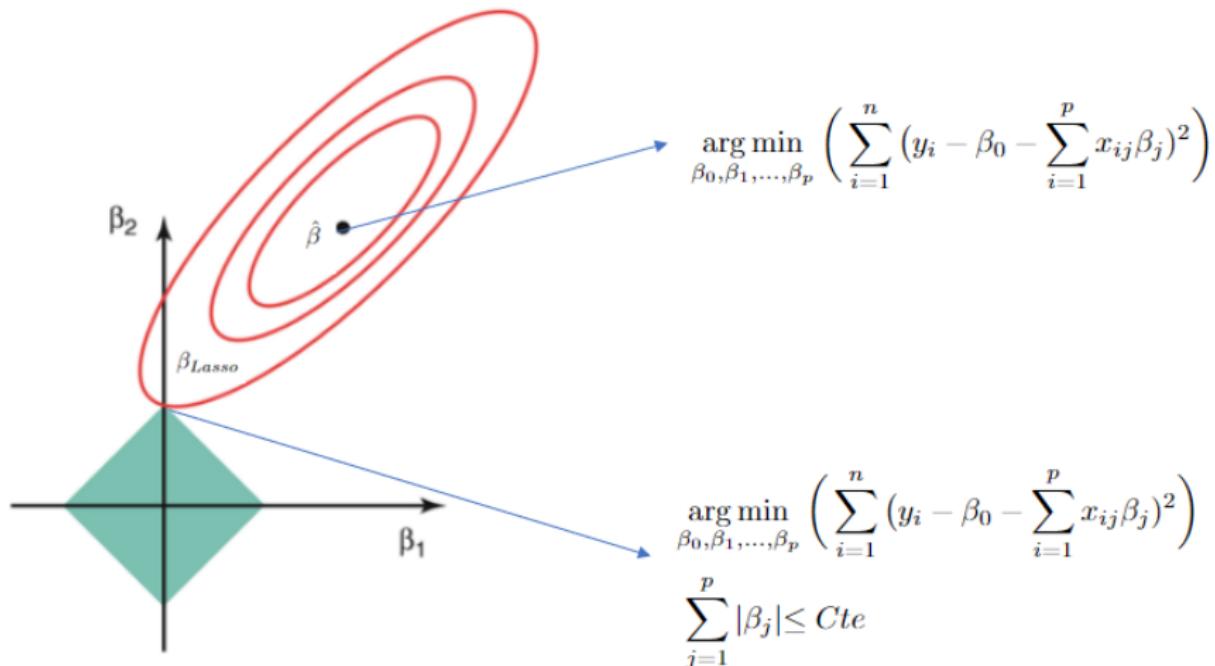
sous la contrainte :

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

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

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

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 $[-1, 1]$:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- **Standardisation** redimensionne les données pour qu'elles aient une moyenne de 0 et un écart type de 1 :

$$X_{std} = \frac{X - \mu}{\sigma}$$

où μ est la moyenne et σ l'écart type.

Algorithmes nécessitant la normalisation ou la standardisation

Certains algorithmes de Machine Learning exigent que les données soient normalisées ou standardisées pour assurer leur bon fonctionnement :

- k -Means Clustering
- Analyse en Composantes Principales (PCA)
- Régression Ridge, Lasso, ElasticNet
- Machines à vecteurs de support (SVM)
- Réseaux de neurones

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

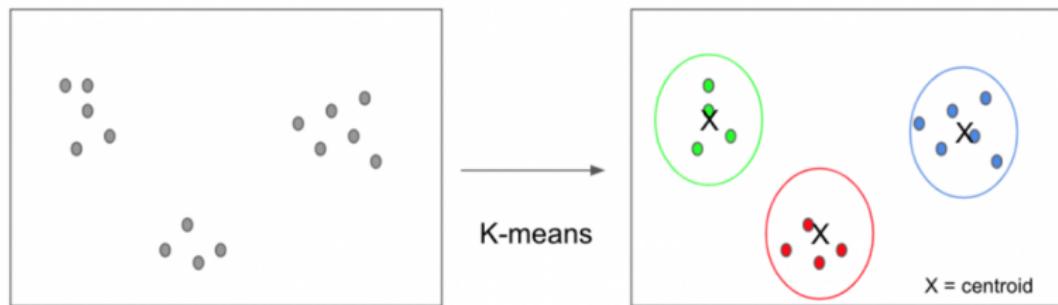
Dans l'apprentissage non supervisé, on considère n observations sans labels. On s'intéresse fondamentalement à la probabilité jointe de ces observations.

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

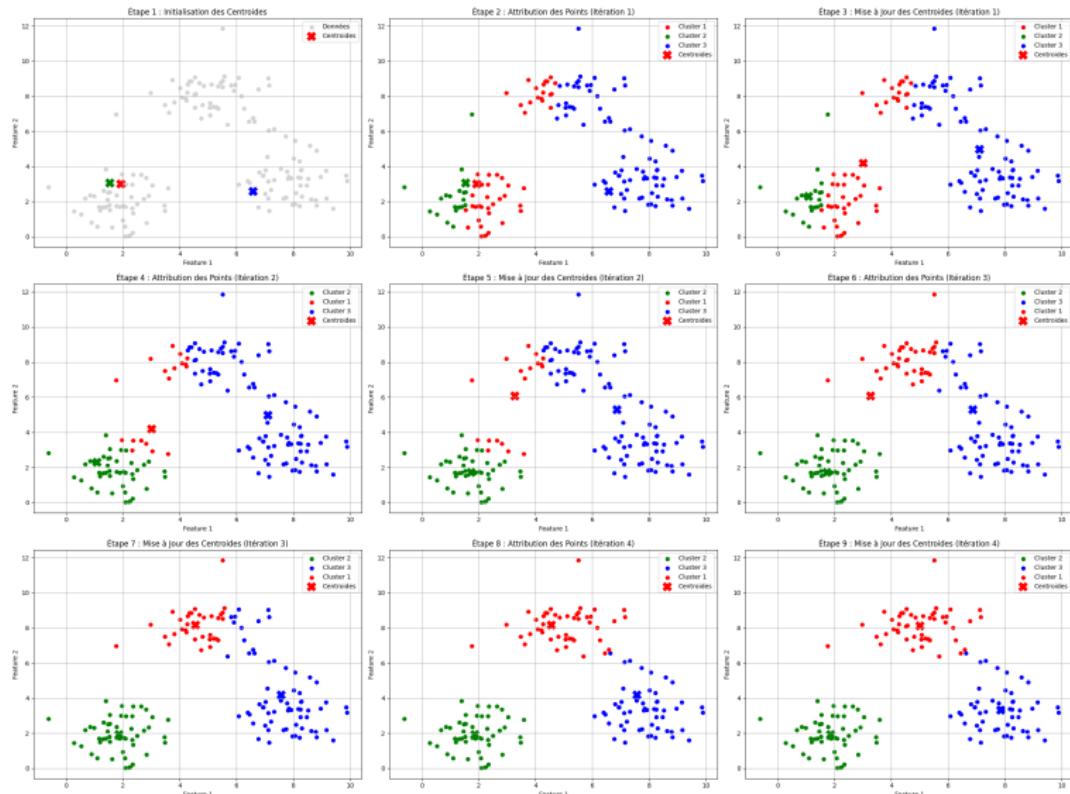
- **Clustering (partitionnement)** : cela consiste à partitionner les n observations en K groupes pertinents (généralement le critère de pertinence a une signification d'un point de vue métier).
- **Réduction de dimension** : il s'agit de trouver une représentation des données originelles dans un nouvel 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 explicables, etc.).
- **Détection d'anomalie** : il s'agit de détecter des observations qui présentent un profil (des features) inhabituel par rapport au profil moyen de la majorité des observations.

K-means

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.



Algorithme de Lloyd

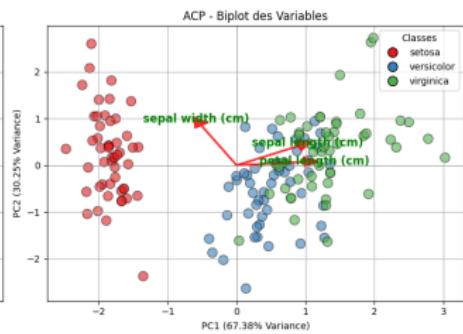
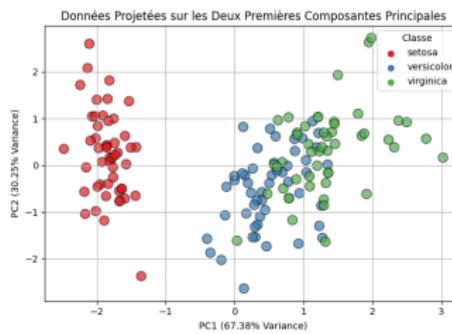
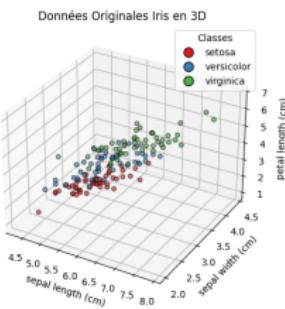


Remarques

- 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 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 $C = \frac{1}{n-1} X^T X$.

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 λ_k .

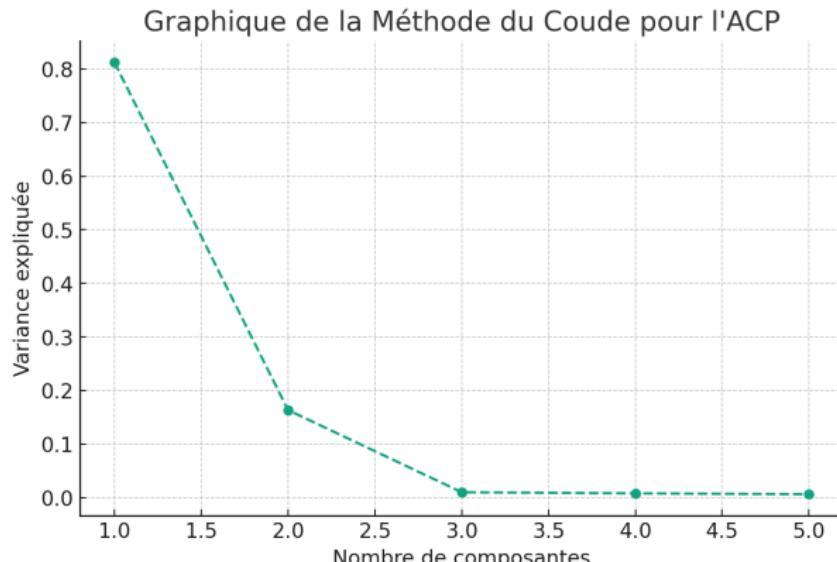
La contribution de chaque composante principale à la variance totale est donnée par :

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

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.

On utilise généralement la méthode du coude (Scree plot). Il s'agit d'un graphique montrant la proportion de la variance expliquée en fonction du nombre de composantes.



Isolation Forest : Principe

L'Isolation Forest est une technique de détection d'anomalies basée sur l'isolement des observations. Son efficacité repose sur l'hypothèse que les anomalies sont "faciles à isoler" par rapport aux observations normales.

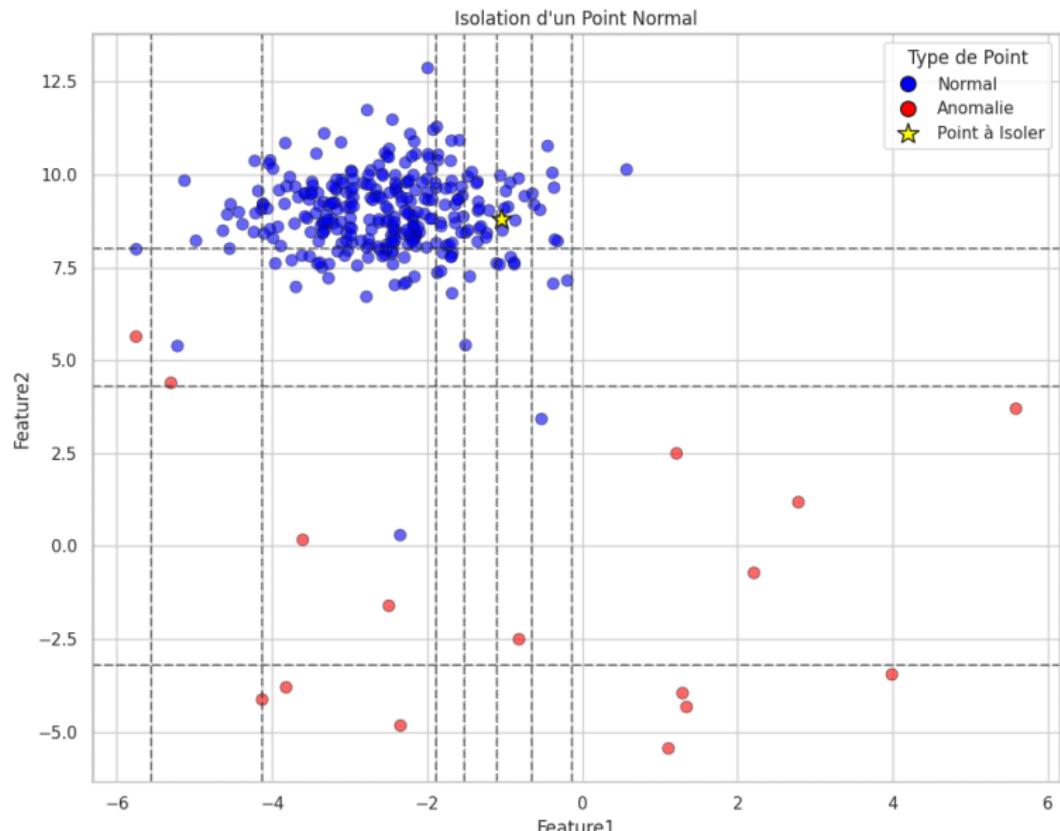
- Fonctionne en construisant des arbres d'isolement à partir de sous-ensembles de données.
- Isoler une observation signifie la séparer des autres par des divisions aléatoires de l'espace des caractéristiques.
- Moins de divisions sont nécessaires pour isoler une anomalie, ce qui constitue le fondement du score d'anomalie.

Les arbres d'isolement sont construits de manière récursive :

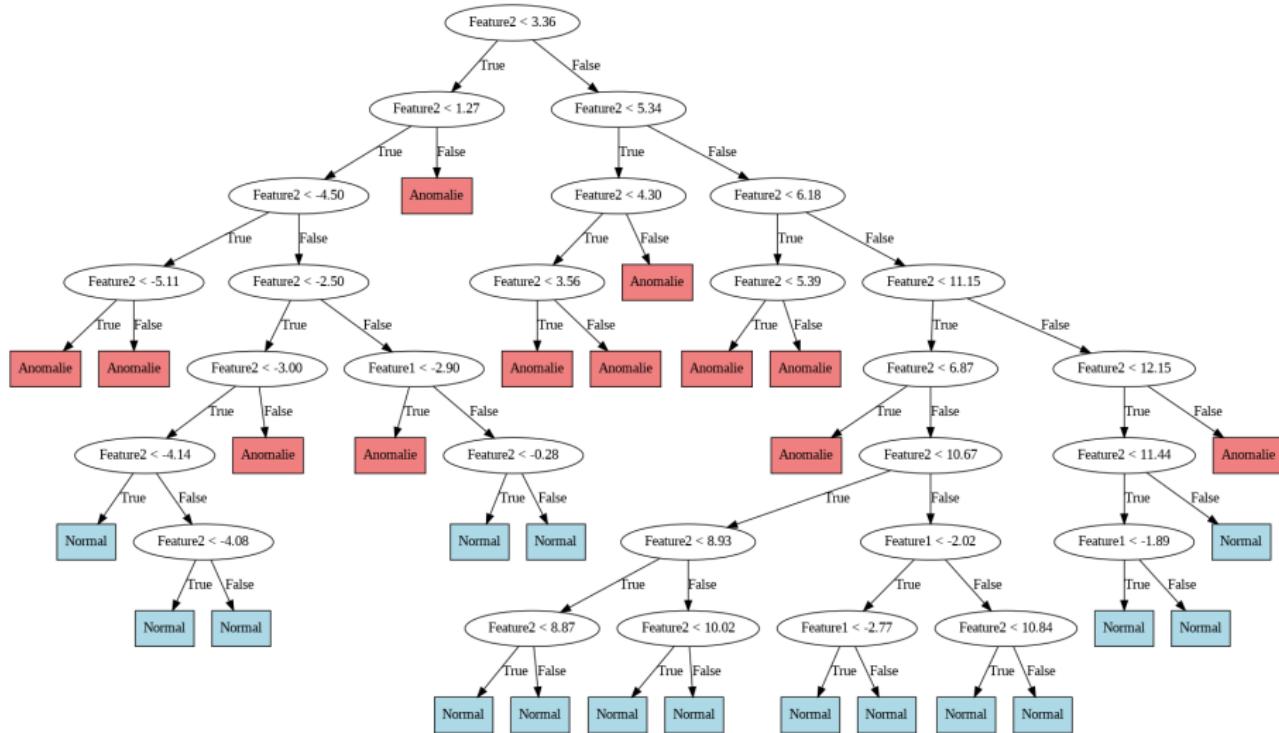
- ➊ Sélection aléatoire d'un sous-ensemble de données.
- ➋ Choix aléatoire d'une caractéristique et d'une valeur de seuil pour diviser le sous-ensemble.
- ➌ Répétition des divisions jusqu'à l'isolement des observations ou atteinte d'une limite de profondeur prédéfinie.

Chaque arbre est ainsi unique, offrant une perspective différente sur les données.

Isolation tree : fonctionnement 1/2



Isolation tree : fonctionnement 2/2



Deep learning

Introduction aux réseaux de neurones

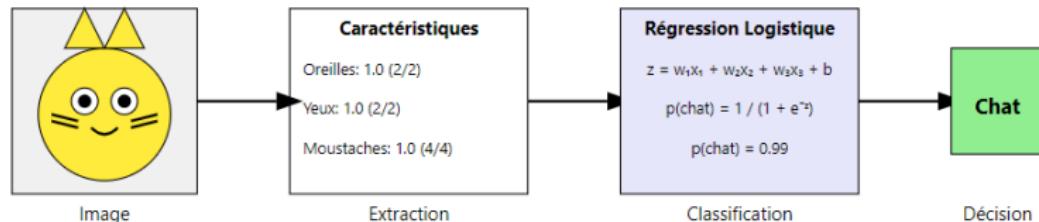
Définition : Les réseaux de neurones sont des modèles computationnels inspirés par le fonctionnement des neurones dans le cerveau humain. Ils sont capables d'apprendre des tâches complexes en modélisant des relations non linéaires entre les entrées et les sorties.

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.
- **Modélisation en grande dimension** : Les modèles de deep learning sont particulièrement adaptés pour les données en grande dimension (images, texte, etc.).
- **Flexibilité** : Applicables à un large éventail de tâches et de typologies de données (images, langage naturel, données graphiques, etc.).

Extraction de features

Extraction de features en machine learning "classique"



Extraction de features en deep learning

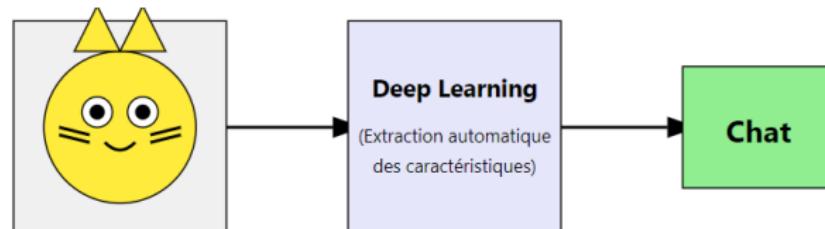
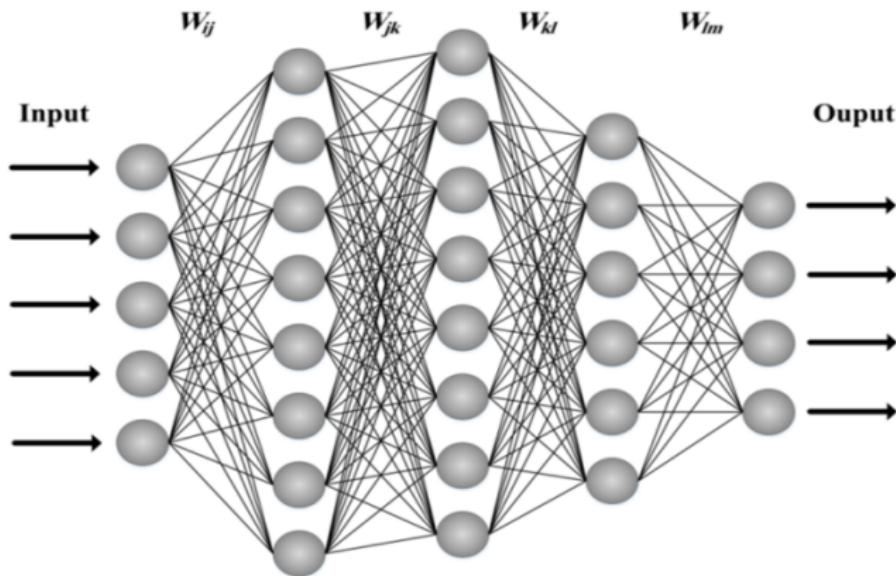
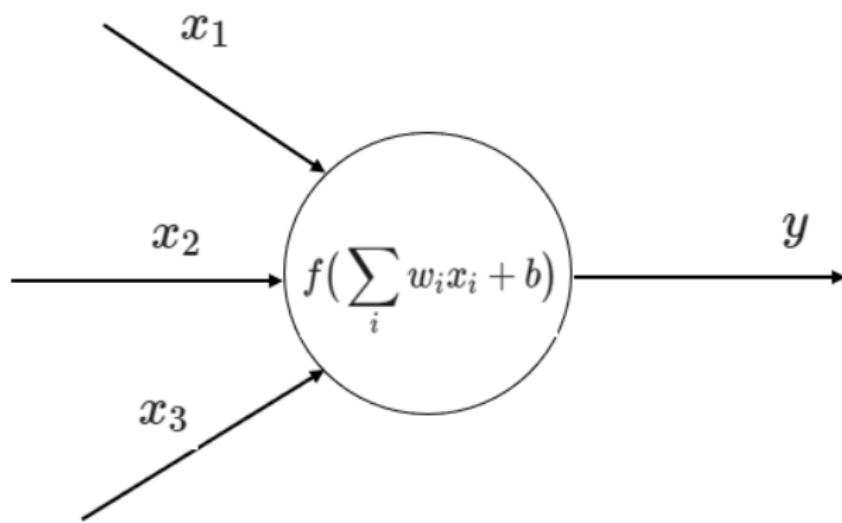


Illustration d'un réseau de neurones classique



Neurone aritificial



Fonctions d'activation

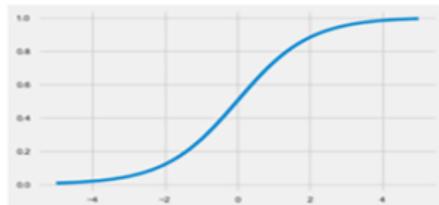
Les fonctions d'activation permettent aux modèles d'apprendre des relations plus complexes en capturant les non-linéarités dans les données.

- **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.

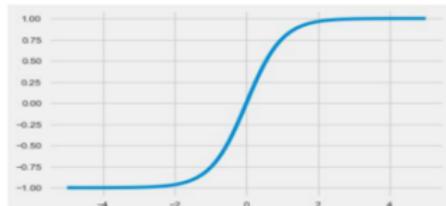
$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Illustration des fonctions d'activation

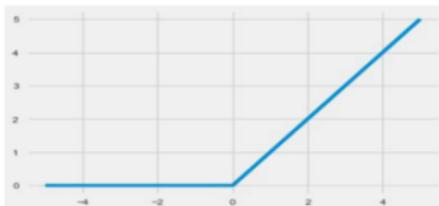
Sigmoïde



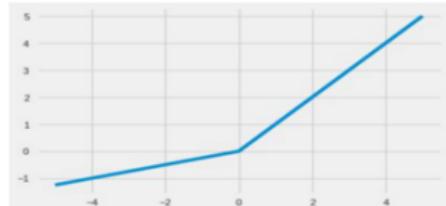
Tanh



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) :

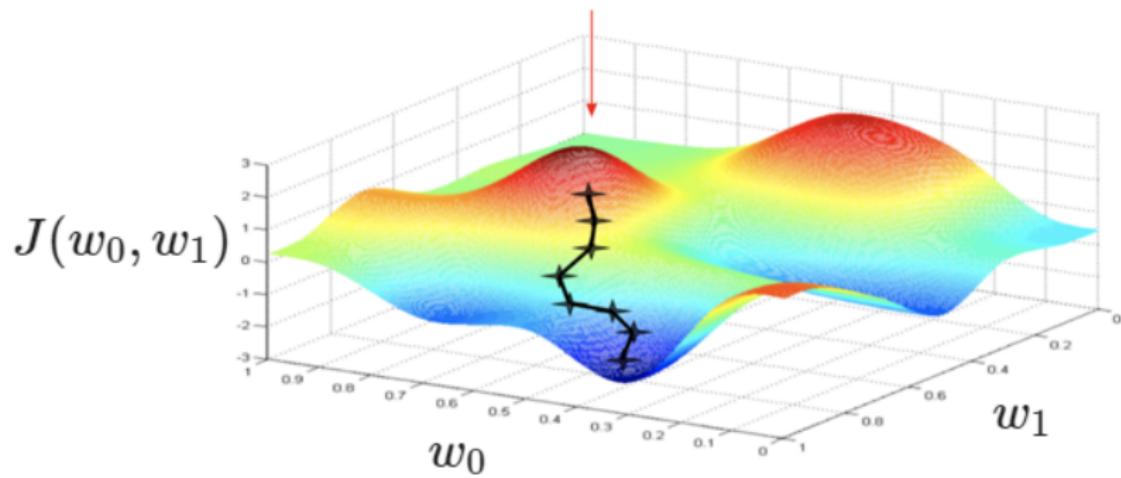
- Méthode d'optimisation utilisée pour mettre à jour les poids du réseau de manière itérative.
- À chaque itération, un sous-ensemble (batch) de données est utilisé pour calculer le gradient de la fonction de coût.
- Les poids sont mis à jour dans la direction opposée du gradient pour réduire l'erreur.

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

où :

- w_{old} et w_{new} sont les valeurs des poids avant et après la mise à jour,
- η 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



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 $n - 1$:

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

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.

Traitement automatique du langage (NLP)

Introduction aux Embeddings

• Qu'est-ce qu'un Embedding ?

- Un embedding est une représentation vectorielle d'un mot qui capture le contexte du mot dans un document, des relations sémantiques et syntaxiques avec d'autres mots.
- Ils transforment des mots en vecteurs de nombres pour que les algorithmes de machine learning puissent les traiter efficacement.

• Pourquoi sont-ils importants ?

- Les embeddings permettent de capturer non seulement l'identité d'un mot mais aussi ses aspects sémantiques et contextuels.
- Ils facilitent des tâches telles que la classification de texte, la traduction automatique, et la détection des sentiments.

• Évolution des embeddings

- Historiquement, les mots étaient représentés comme des indices ou des vecteurs one-hot, où chaque mot est indépendant des autres.
- Les embeddings modernes, tels que Word2Vec et GloVe, représentent les mots dans des espaces vectoriels continus où les mots de sens similaires sont proches les uns des autres.

Différence entre One-hot Encoding et Embeddings

• One-hot Encoding

- Chaque mot est représenté par un vecteur avec un '1' dans la position qui lui est propre et des '0' partout ailleurs. Ce type de représentation est simple mais très inefficace en termes d'espace et ne capture pas les relations entre les mots.
- Exemple : pour un vocabulaire de dimension 100 000 :

Véhicule = [0, 0, 1, 0, 0, 0, 0, 0, ..., 0, 0]

Voiture = [0, 0, 0, 0, 0, 1, 0, 0, 0, ..., 0, 0]

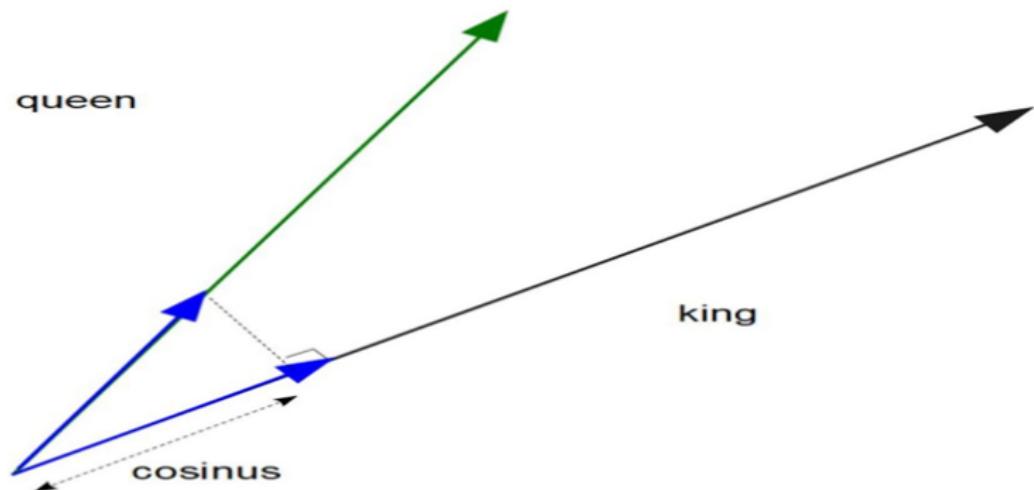
Cette représentation ne permet pas de prendre en compte la dimension sémantique

• Embeddings

- Les embeddings représentent les mots comme des vecteurs denses de nombres flottants (généralement entre 50 et 300 dimensions).
- Cette représentation est beaucoup plus riche et peut capturer des relations complexes entre les mots, comme la similarité sémantique.

Similarité sémantique

Nous sommes intéressés par des représentations de mots qui capturent la distance sémantique, sous forme de produit scalaire par exemple (ou distance cosinus).



Représentations distribuées : Principes

“You shall know a word by the company it keeps”

— J.R. Firth (1957)

Les mots sont similaires s'ils apparaissent fréquemment dans le même contexte.

- Il conduit son *véhicule* pour rentrer à la maison.
- Il conduit sa *voiture* pour rentrer chez lui.

Construction d'une représentation distribuée

Considérons le corpus suivant à titre d'exemple :

cnn in crop analysis

cnn and svm are widely used.

linear_regression performed along with svm

linear_regression for crop and farm

svm being used for farm monitoring

Do cnn, svm and linear_regression appear in the same context?

Le vocabulaire est alors :

[cnn, in, crop, analysis, and, svm, are, widely, used,
linear_regression, performed, along, with, for, farm, being,
monitoring, do, appear, the, same, context]

$$\dim(\text{vocabulaire}) = 22$$

Similarité sémantique

La matrice de co-occurrence représente la fréquence à laquelle les mots apparaissent ensemble deux à deux.

cnn in crop ...

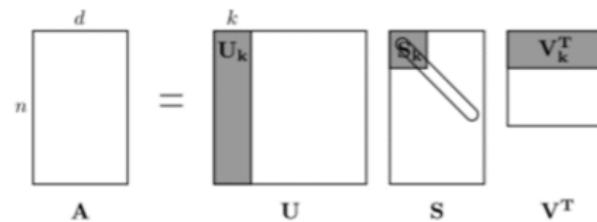
```
cnn [[0, 2, 1, 1, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],  
in [2, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],  
crop [1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
... [1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[2, 1, 0, 0, 1, 0, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
[1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[1, 0, 0, 0, 1, 2, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0],  
[1, 1, 1, 0, 1, 2, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1],  
[0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
[0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0],  
[0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0],  
[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],  
[1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1],  
[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0]]]
```

Réduction de la dimension

La décomposition en valeurs singulières est une technique permettant de réduire la dimension des données (similaire à l'ACP).

Étant donné une matrice $A \in \mathbb{R}^{n \times d}$

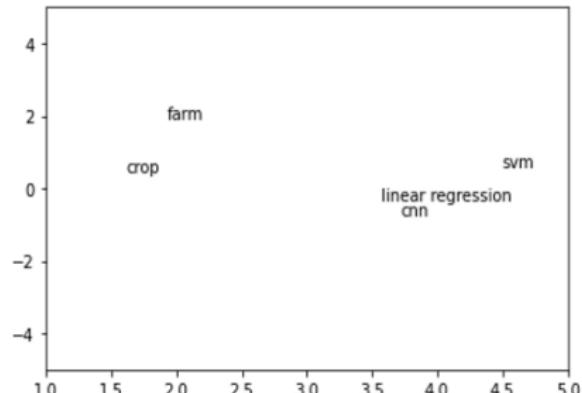
$$A = UDV^T \quad \text{où} \quad U \in \mathbb{R}^{n \times r}, D \in \mathbb{R}^{r \times r}, V \in \mathbb{R}^{d \times r}.$$



U est la matrice contenant les représentations (vecteurs de mots)

Visualisation des représentations distribuées

```
cnn :[ [ 3.72113518, -0.73233585],  
ln [ 2.7940387 , -1.40864784],  
crop [ 1.61178082, 0.44786021],  
... [ 0.77784994, -0.31230389],  
[ 2.12245048, 1.29571556],  
[ 4.49293777, 0.58090417],  
[ 1.33312748, 0.66758743],  
[ 1.33312748, 0.66758743],  
[ 2.25882809, 1.80738544],  
[ 3.56593605, -0.31006976],  
[ 0.95394179, 0.079159 ],  
[ 0.95394179, 0.079159 ],  
[ 0.95394179, 0.079159 ],  
[ 0.95394179, 0.079159 ],  
[ 1.92921553, 1.94783497],  
[ 1.92921553, 1.94783497],  
[ 1.12301312, 1.42126096],  
[ 1.12301312, 1.42126096],  
[ 2.26025282, -1.31571175],  
[ 2.26025282, -1.31571175],  
[ 2.26025282, -1.31571175],  
[ 2.26025282, -1.31571175],  
[ 2.26025282, -1.31571175]]
```



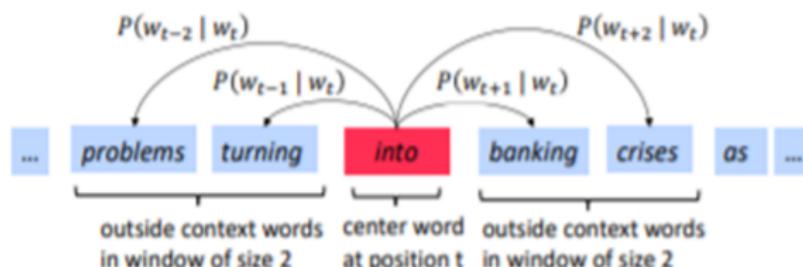
Word2Vec : Principles

- **Principes de base :**

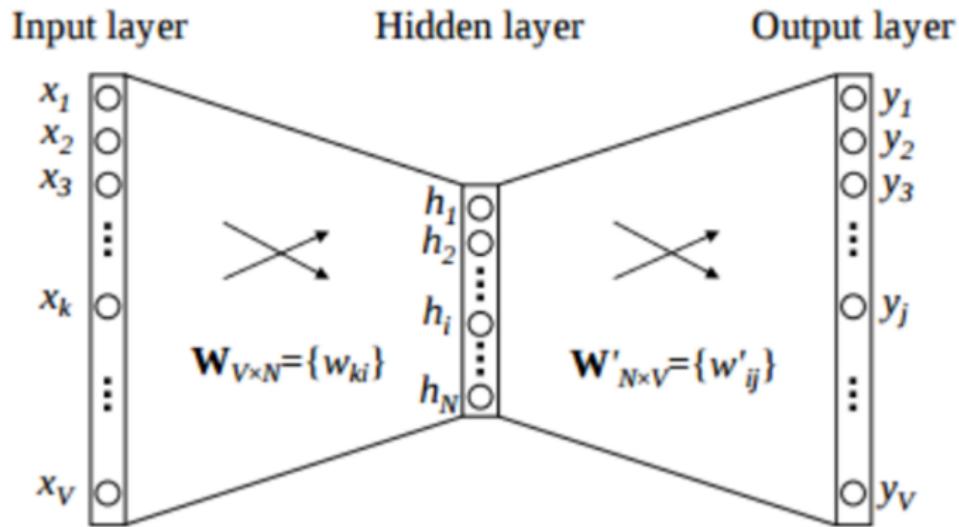
- Word2Vec est basé sur l'hypothèse distributionnelle : les mots qui apparaissent dans des contextes similaires ont des significations similaires.
 - Utilise un réseau de neurones peu profond pour apprendre des embeddings de mots à partir de grands corpus.

- Deux architectures principales :

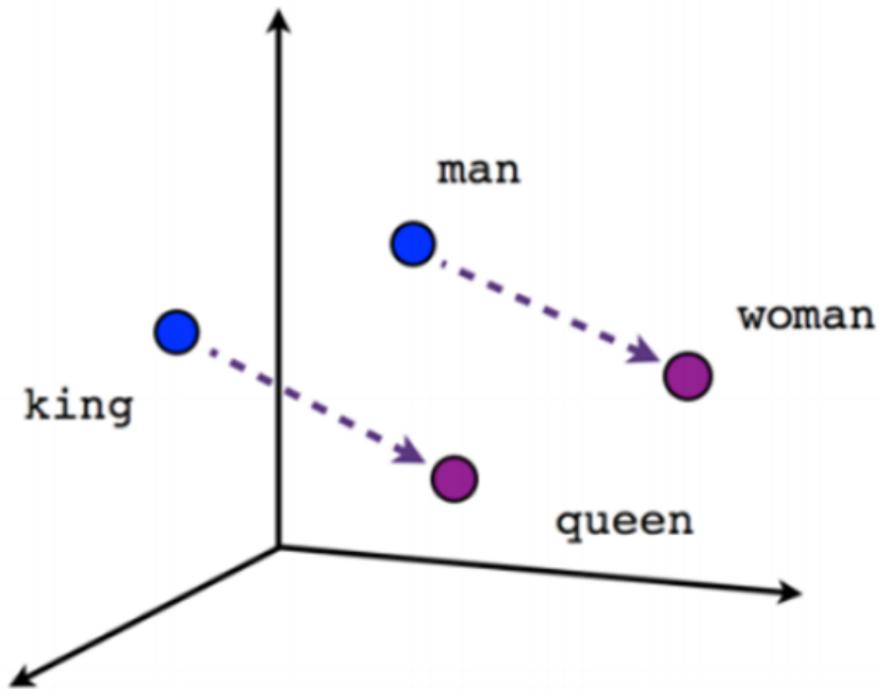
- ① *Continuous Bag of Words (CBOW)* : Prédit le mot cible à partir du contexte.
 - ② *Skip-Gram* : Prédit le contexte à partir du mot cible.



Architecture du modèle CBOW



Représentations Word2Vec



Applications pratiques de Word2Vec

- **Analogie de mots :**

- Word2Vec est célèbre pour capturer des relations complexes, comme "homme est à femme ce que roi est à reine".
- Permet de résoudre des analogies en utilisant des opérations arithmétiques simples sur les vecteurs de mots.

- **Clustering sémantique :**

- Les embeddings peuvent être utilisés pour regrouper des mots sémantiquement similaires, facilitant l'analyse de textes volumineux.

- **Amélioration des systèmes de recommandation :**

- Les vecteurs de mots de Word2Vec peuvent être utilisés pour améliorer la précision des recommandations en comprenant mieux les préférences des utilisateurs.

Autres représentations distribuées

Il existe d'autres représentations distribuées :

- Glove (2014) : proposé par une équipe de Stanford, il combine à la fois les techniques modernes de deep learning et les techniques statistiques (co-occurrences entre les mots). Il permet d'avoir des représentations des mots plus globales que Word2Vec.
- Fasttext (à partir de 2016) : la librairie a été créée par Facebook. Le modèle est entraîné sur des subwords (n-grams de caractères). Il est ainsi plus efficace pour traiter les mots inconnus (out of vocabulary).
- Représentations contextuelles (Transformers), etc.

Réseaux de neurones récurrents

Perspective historique des RNN

Les Réseaux de Neurones Récurrents (RNN) ont évolué au fil du temps, marquant des étapes importantes dans le domaine du machine learning et du NLP.

Développements clés :

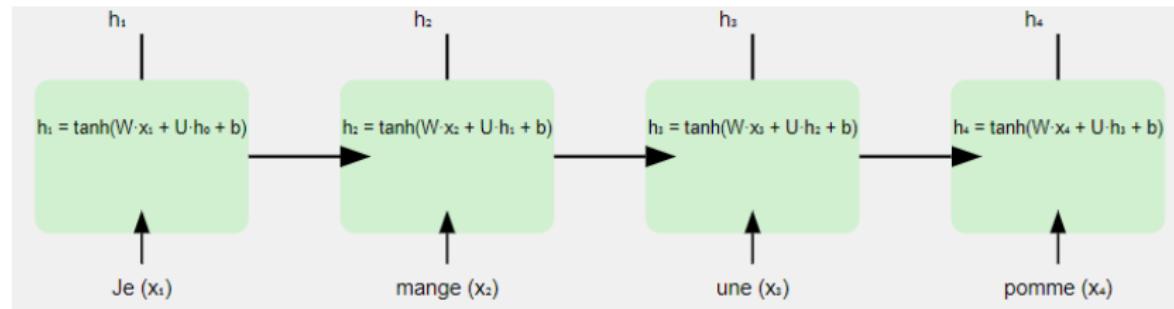
- **1980s : Naissance des RNN** - Introduction des concepts de base des RNN par John Hopfield et David Rumelhart, posant les fondations des réseaux à mémoire.
- **1990s : Popularisation des RNN** - Jordan et Elman développent des architectures permettant une meilleure gestion des séquences temporelles et des dépendances.
- **Début des années 2000** - Identification des problèmes de disparition et d'explosion du gradient, limitant l'efficacité des RNN sur de longues séquences.
- **1997 : Avènement des LSTM** - Introduction des LSTM par Hochreiter et Schmidhuber, offrant une solution aux problèmes de mémoire à long terme.
- **2010s : GRU** - Les GRU simplifient la structure des LSTM. Les RNN sont de plus en plus utilisés dans des applications complexes comme la traduction automatique et la reconnaissance vocale.

Introduction aux RNN

Les Réseaux de Neurones Récursifs (RNN) sont une classe de réseaux de neurones artificiels spécialement conçus pour traiter des séquences de données, tels que des séries temporelles ou des séquences textuelles.

Caractéristiques:

- **Mémoire à court terme** : Les RNN ont la capacité de se "souvenir" d'informations passées grâce à leurs connexions récurrentes.
 - **Traitements de séquences** : Ils sont particulièrement adaptés pour des tâches où les données sont séquentielles et où le contexte est important (ex: langage naturel, musique).



RNN avec Long Short-Term Memory (LSTM)

Les LSTM sont une variante des RNN conçus pour mieux capturer les dépendances à long terme. Ils introduisent des 'cellules mémoire' avec des portes de régulation :

- Porte d'oubli : contrôle la quantité d'informations à retenir de l'état précédent.
- Porte d'entrée : contrôle la quantité d'informations à ajouter de l'entrée actuelle.
- Porte de sortie : détermine la quantité d'informations à transmettre à l'état suivant.

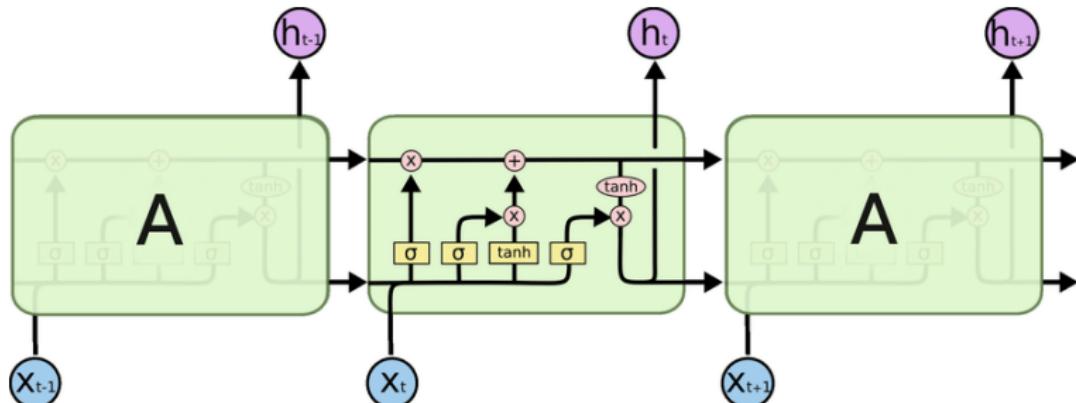
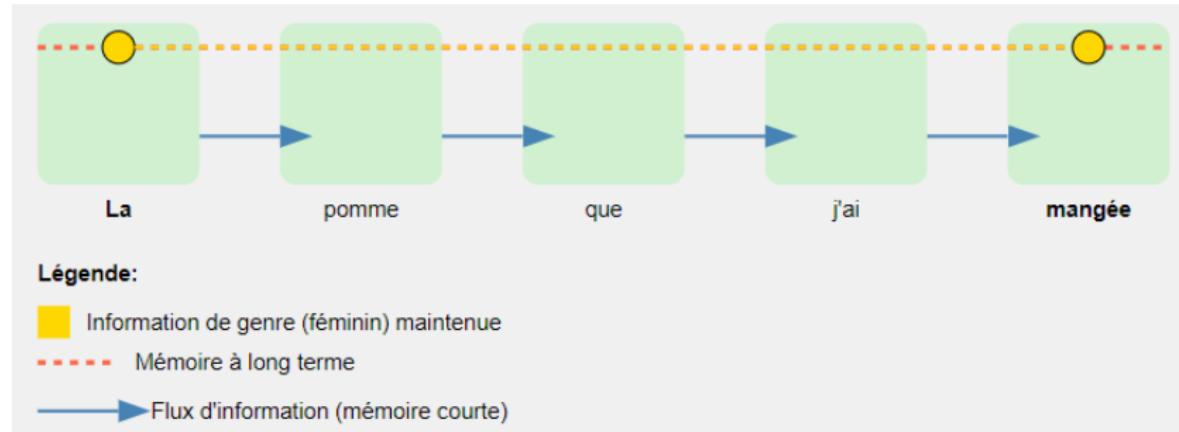


Illustration de la mémoire longue dans les LSTM

- Les LSTM excellent dans la gestion des dépendances à long terme
- Exemple : "La pomme que j'ai mangée"
- Mémoire à long terme :
 - Stocke l'information sur l'article "La" (indiquant le féminin)
 - Maintient cette information de genre à travers la séquence
- Permet l'accord correct du participe passé à distance

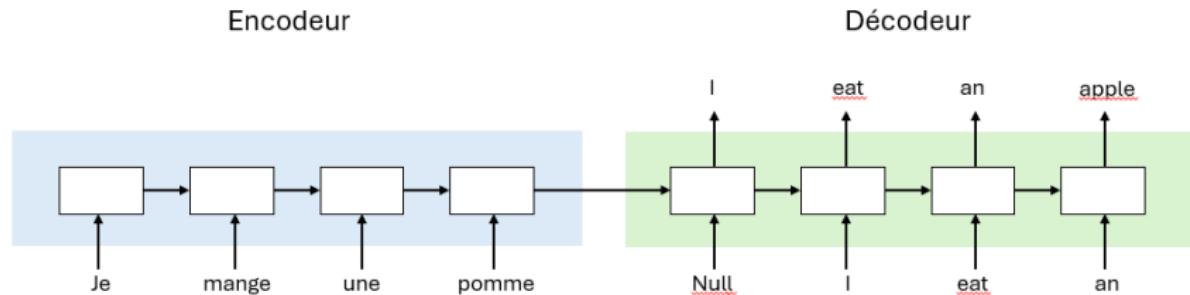


Modèles Seq-to-Seq

Les modèles de séquence à séquence (seq-to-seq) sont une classe de modèles en apprentissage profond conçus pour transformer une séquence d'entrée en une séquence de sortie.

L'architecture seq-to-seq comprend deux composants principaux :

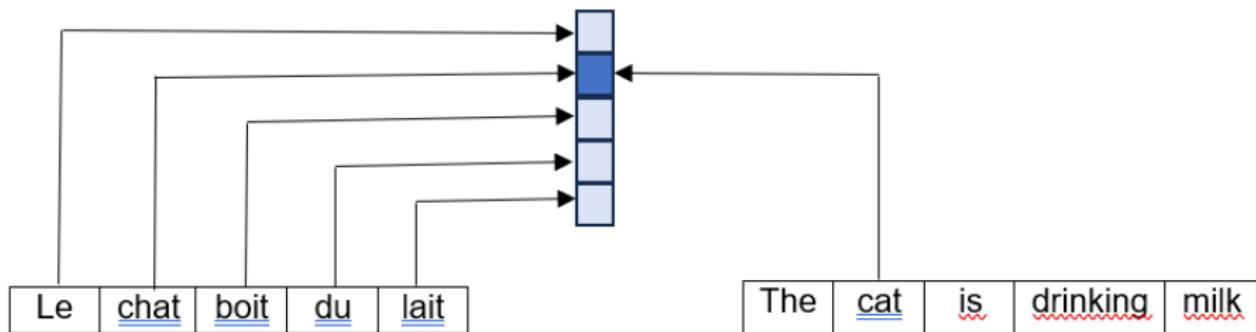
- **Encodeur** : Un réseau de neurones qui lit et encode la séquence d'entrée en un vecteur de contexte (une représentation dense de la séquence).
- **Décodeur** : Un autre réseau de neurones qui lit le vecteur de contexte pour générer la séquence de sortie, un élément à la fois.



Modèles Seq2Seq avec mécanisme d'Attention

Le mécanisme d'attention est une amélioration clé apportée aux modèles seq-to-seq traditionnels, permettant au modèle de se "concentrer" sur différentes parties de la séquence d'entrée lors de la génération de la séquence de sortie.

- **Objectif** : Surmonter les limitations des encodeurs seq-to-seq qui compressent toute l'information d'une séquence d'entrée dans un vecteur de contexte fixe.
- **Avantage** : Améliore la capacité du modèle à gérer de longues séquences d'entrée, en rendant le processus de génération de la séquence de sortie plus dynamique et contextuellement informé.



Transformers et LLMs

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

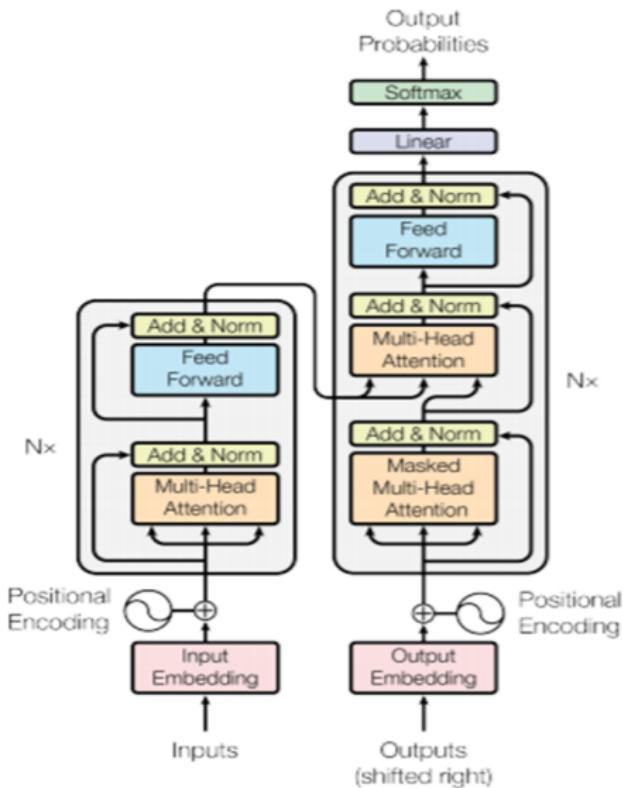
Lukasz Kaiser*
Google Brain
lukasz.kaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

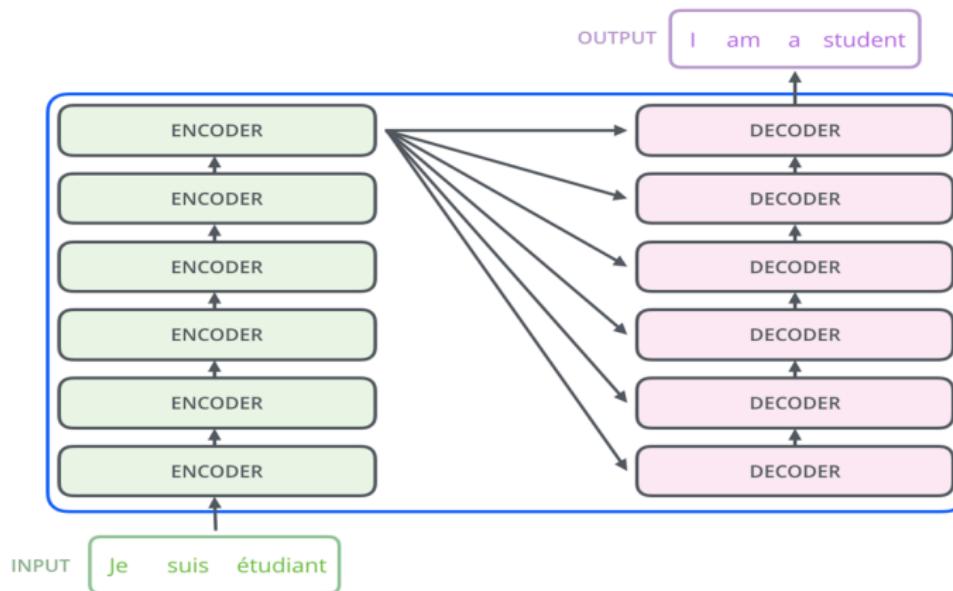
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to

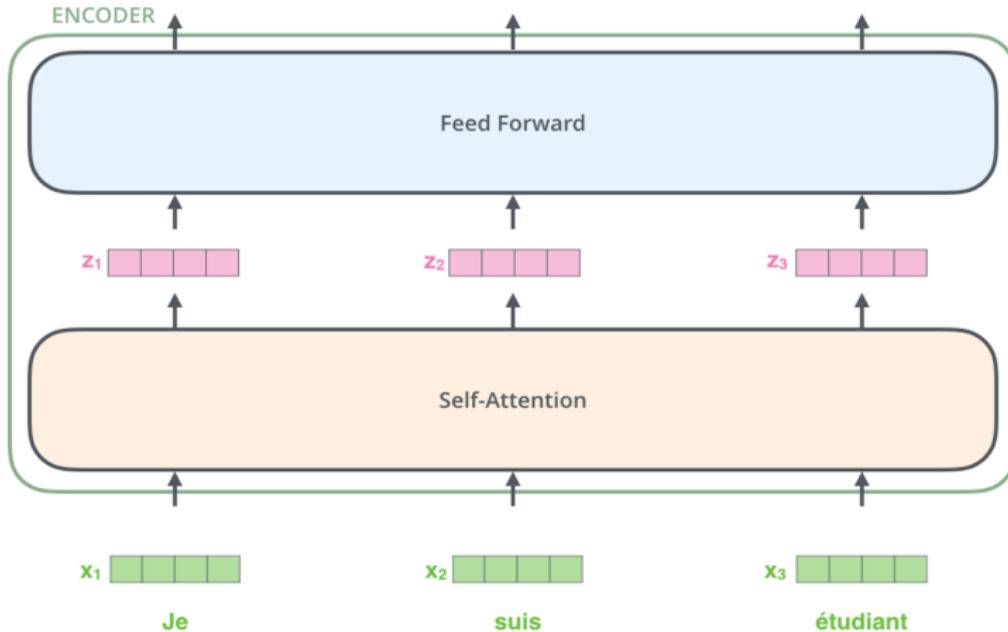
Transformer originel



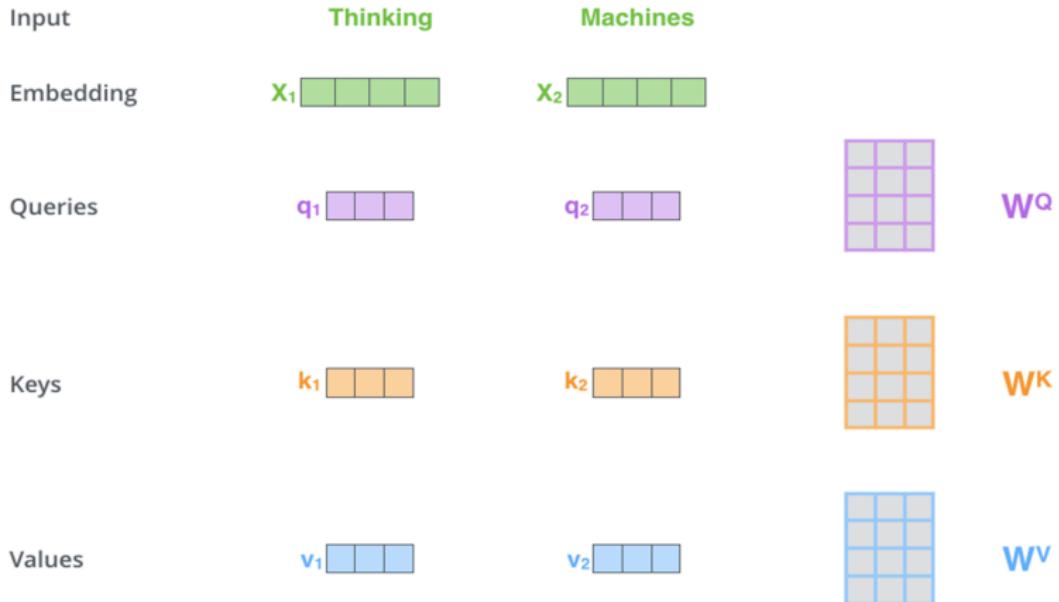
Mécanisme de cross-attention



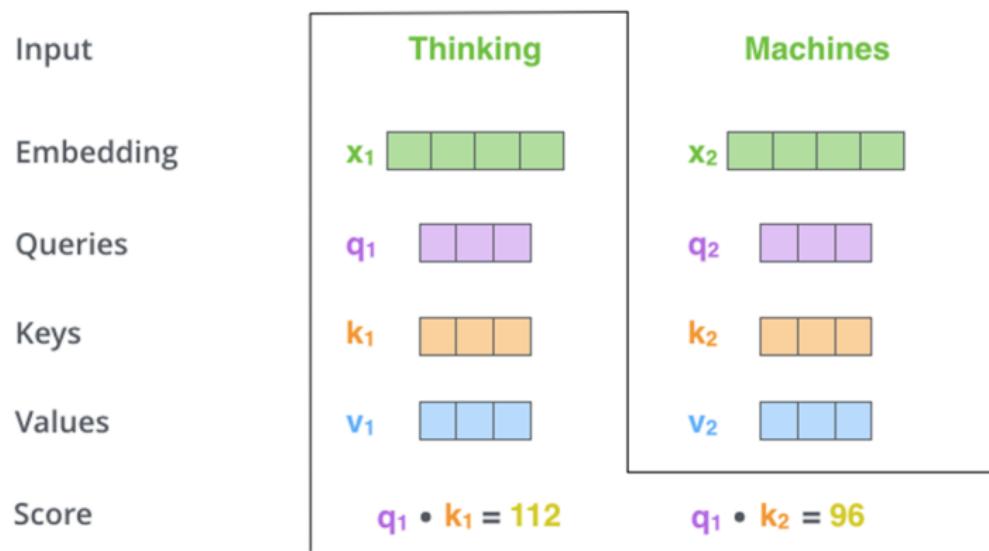
Mécanisme de self-attention



Fonctionnement du mécanisme de self-attention



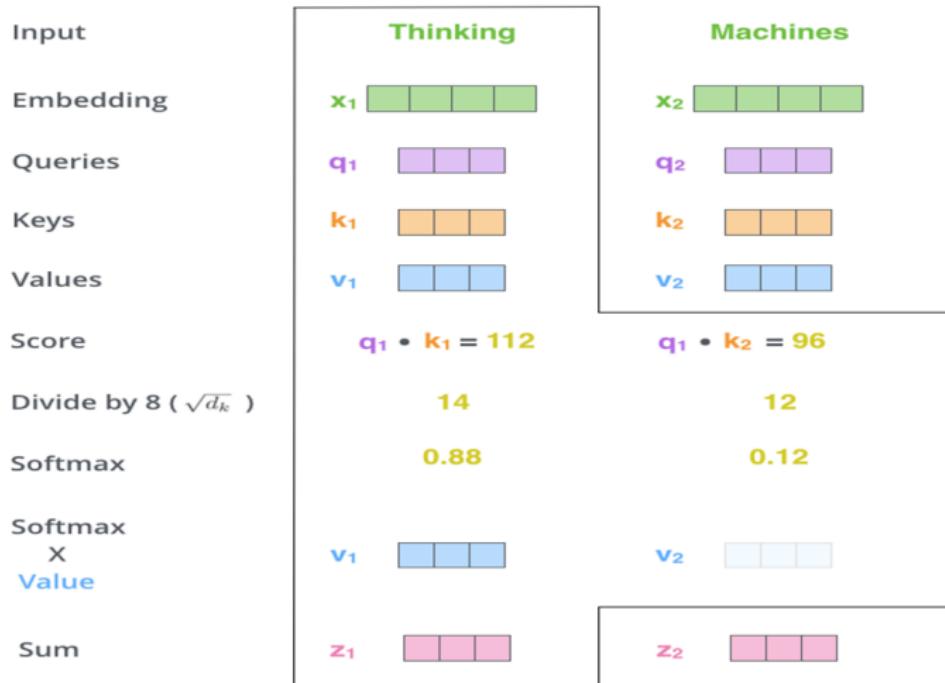
Calcul des scores de self-attention



Calcul des scores de self-attention

Input	Thinking		Machines	
Embedding	x_1	[4 green boxes]	x_2	[4 green boxes]
Queries	q_1	[3 purple boxes]	q_2	[3 purple boxes]
Keys	k_1	[3 orange boxes]	k_2	[3 orange boxes]
Values	v_1	[3 blue boxes]	v_2	[3 blue boxes]
Score	$q_1 \bullet k_1 = 112$		$q_1 \bullet k_2 = 96$	
Divide by 8 ($\sqrt{d_k}$)	14		12	
Softmax	0.88		0.12	

Calcul de la nouvelle représentation



Performances du tranformer originel

Entraîné sur WMT 2014 English-German dataset, comprenant près de 4.5 millions de phrases sentence, le WMT 2014 English-French dataset, comprenant près de 36 millions de phrases.

- 8 NVIDIA P10 GPUs
- **Base** : 12 heures
- **Large** : 3.5 jours

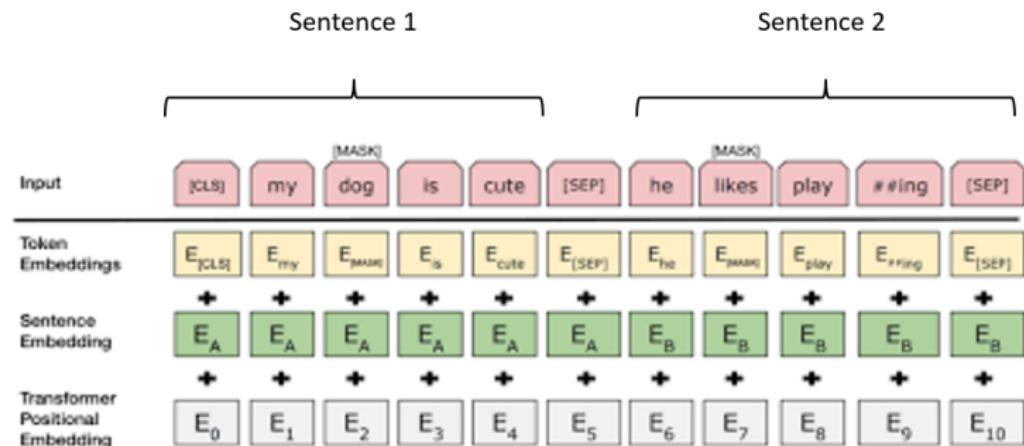
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.8	$2.3 \cdot 10^{19}$	

BERT

BERT (Bidirectional Encoder Representations from Transformers) représente une avancée majeure dans le NLP, introduisant une approche novatrice pour la modélisation de langage. Il utilise la bidirectionnalité pour comprendre le contexte des mots, permettant une compréhension plus fine du langage.

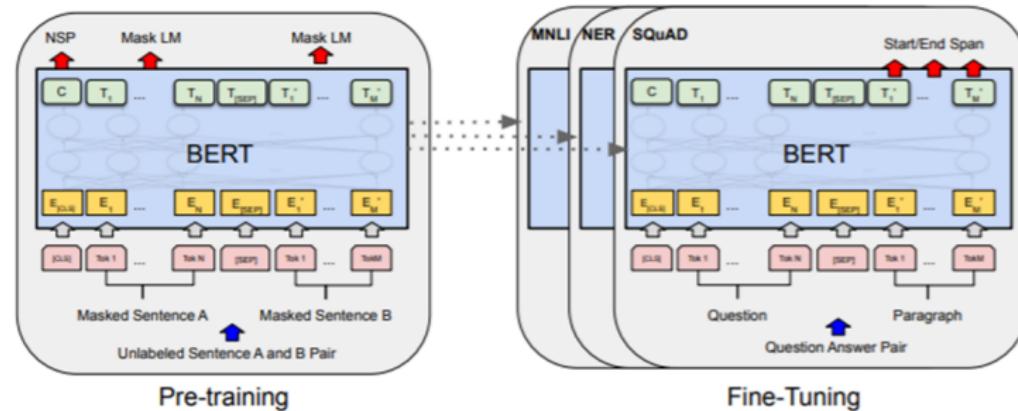
Pré-entraînement



Fine-tuning de BERT pour des tâches spécifiques

Après pré-entraînement, BERT est affiné pour des tâches spécifiques:

- Ajout d'une couche de sortie spécifique à la tâche (classification, NER, QA).
 - Fine-tuning de tous les paramètres du modèle pré-entraîné sur le corpus de la tâche.



Performances de BERT

BERT a été pré-entraîné sur le BookCorpus (800 millions de mots) et English Wikipedia (2,500 millions de mots).

Deux modèles :

- **BERT Base** : 12 couches avec 110 millions de paramètres.
- **BERT Large** : 24 couches avec 340 millions de paramètres.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Introduction aux modèles autorégressifs - GPT-3

GPT-3 (Generative Pre-trained Transformer 3) est le modèle de langue développé par OpenAI, représentant la troisième génération de la série GPT. Avec 175 milliards de paramètres, GPT-3 pousse les limites de la génération de texte et de la compréhension du langage naturel.

- **Capacités générales** : GPT-3 excelle dans une variété de tâches de TALN sans fine-tuning spécifique, grâce à sa puissance de modélisation du langage.
- **Architecture autorégressive** : Utilise un modèle Transformer pour prédire le mot suivant dans un texte, apprenant des patterns complexes sur des données massives.
- **Approche few-shot learning** : Capable d'adapter ses réponses à des tâches spécifiques avec peu ou pas d'exemples d'entraînement.
- **Impact sur IA et le NLP** : GPT-3 a significativement avancé les capacités des systèmes d'IA dans la compréhension et la génération de langage naturel, ouvrant de nouvelles voies pour l'application de l'intelligence artificielle.

Architecture de GPT-3

GPT-3 repose sur une architecture Transformer améliorée, optimisée pour traiter efficacement de grandes quantités d'informations et générer des réponses cohérentes et contextuellement pertinentes.

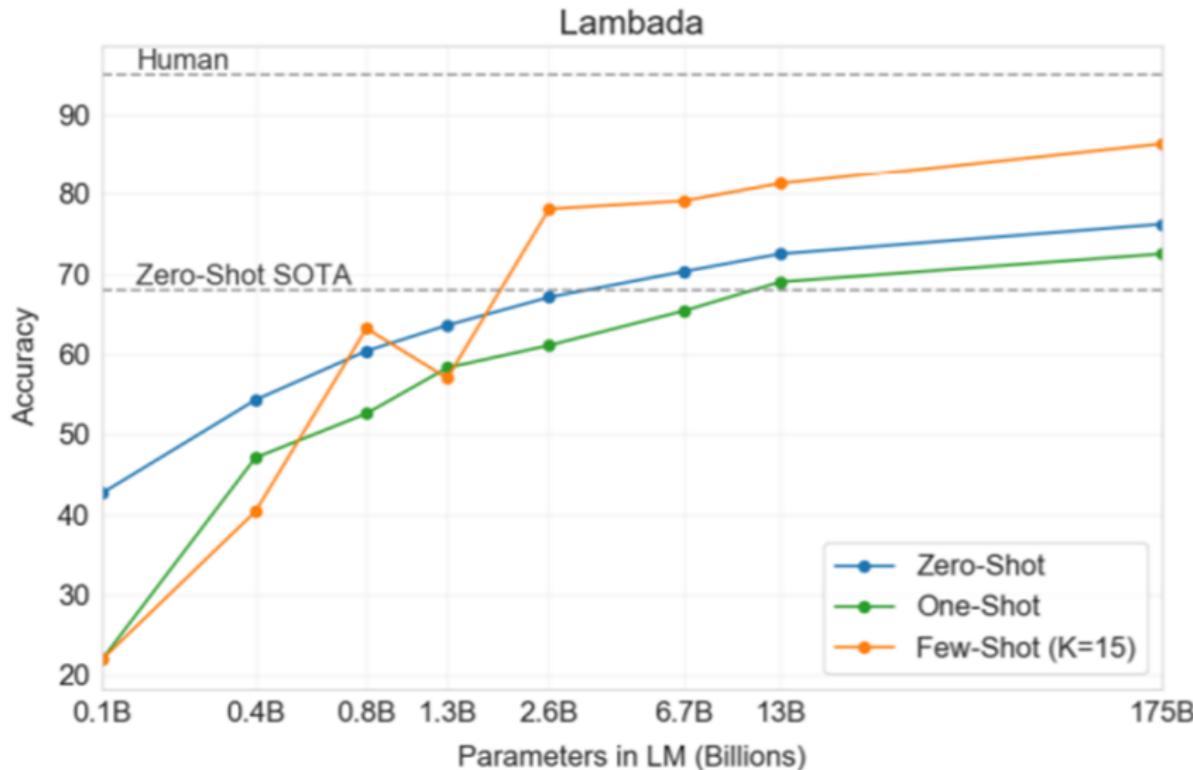
- **Taille du modèle :** Avec 175 milliards de paramètres, GPT-3 est l'un des modèles de langue les plus grands et les plus complexes à ce jour.
- **Mécanisme d'attention :** L'attention multi-têtes permet au modèle de pondérer différemment les parties d'un input, améliorant la compréhension du contexte.
- **Optimisation et entraînement :** Techniques d'optimisation avancées pour gérer la taille du modèle et l'efficacité de l'entraînement.

Few-Shot Learning avec GPT-3

L'une des innovations les plus remarquables de GPT-3 réside dans sa capacité à effectuer du few-shot learning, permettant au modèle de comprendre et d'exécuter des tâches spécifiques avec très peu d'exemples.

- **Définition :** Le few-shot learning désigne la capacité d'un modèle à apprendre une nouvelle tâche à partir d'un très petit nombre d'exemples d'entraînement, souvent seulement quelques-uns.
- **Mécanisme dans GPT-3 :**
 - GPT-3 utilise des prompts contenant quelques exemples de la tâche désirée pour guider le modèle sur ce qui est attendu, avant de présenter la question ou la tâche à résoudre.
 - Le modèle généralise ensuite à partir de ces exemples pour générer des réponses ou des solutions aux problèmes posés, montrant une compréhension étonnante de la tâche avec un minimum de guidance.
- **Exemples d'application :**
 - Classification de texte, génération de résumés, réponse à des questions spécifiques, et plus, avec seulement quelques exemples pour chaque tâche.
- **Impact :** Cette capacité émergente réduit considérablement le besoin de vastes ensembles de données d'entraînement spécifiques à la tâche, ouvrant la voie à des applications plus flexibles et accessibles de modèles de langue.

Performances de GPT-3 en few-shot learning



ChatGPT

ChatGPT, développé par OpenAI, est un modèle de langage basé sur l'architecture GPT (Generative Pre-trained Transformer) optimisé pour comprendre et générer des dialogues naturels. L'entraînement de ChatGPT se décline selon les étapes suivantes :

- ① Pré-entraînement sur un corpus volumineux :** Comme GPT-3, ChatGPT est d'abord pré-entraîné sur un vaste ensemble de données textuelles, englobant un large éventail de la littérature disponible sur Internet, pour apprendre une compréhension générale du langage.
- ② Fine-tuning supervisé :** Ensuite, ChatGPT est affiné sur des dialogues spécifiques pour améliorer ses compétences conversationnelles. Cette étape utilise des paires question-réponse et des conversations pour enseigner au modèle des structures de dialogue et des réponses contextuellement appropriées.
- ③ Reinforcement Learning from Human Feedback (RLHF):** Utilisation de techniques de renforcement pour ajuster les réponses du modèle basées sur les préférences et les corrections fournies par des évaluateurs humains, raffinant davantage la pertinence et la naturalité des réponses.

LLMs propriétaires vs Open Source

• Modèles propriétaires

- ChatGPT, Claude, Gemini...
- Poids non disponibles au public.
- Performances élevées, accès limité par API payante
- Protection intellectuelle stricte (code non accessible)
- Support technique assuré, documentation avancée
- Dépendance aux fournisseurs et coûts potentiellement élevés

• Modèles open source

- Llama, Mistral, Gemma...
- Poids disponibles au public
- Transparence du code, flexibilité d'adaptation
- Performances parfois inférieures, mais amélioration continue
- Gratuit, mais coût de l'infrastructure à gérer
- Exposition aux failles potentielles de sécurité, support communautaire

Conclusion : Choisir entre contrôle, flexibilité et performances optimales.

Déploiement des modèles de machine learning

Pipline d'un projet de machine learning



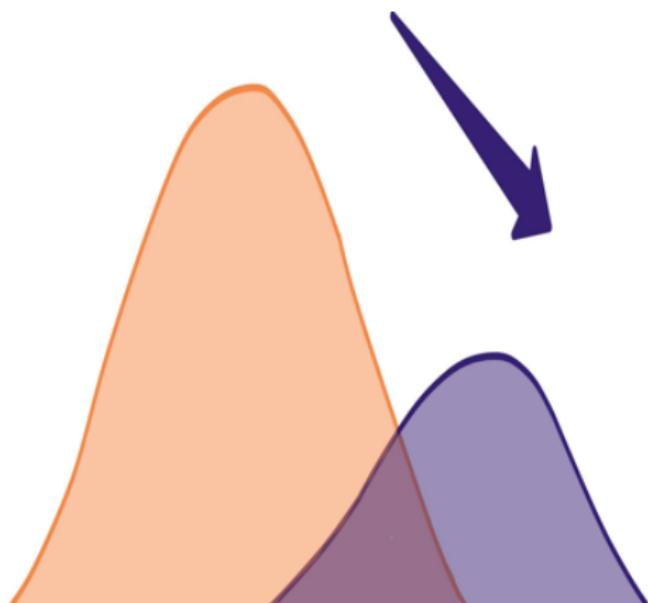
Mais ce processus est en réalité très itératif.

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 :

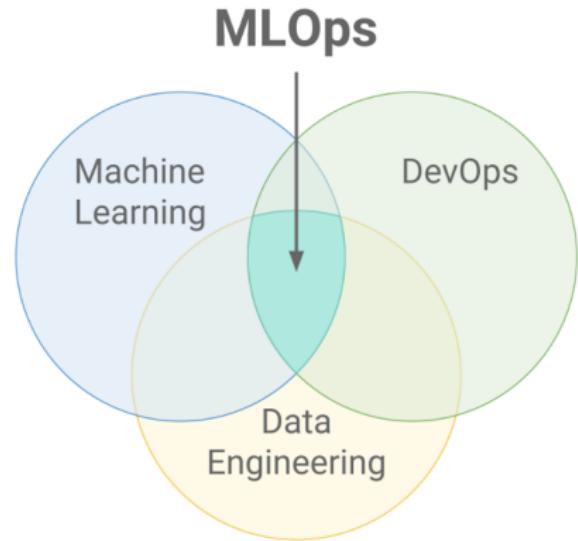
- Différence entre les données de test et les données en production.
 - Changement de l'environnement (avant et après le Covid par exemple).
 - Changement dans le comportement des utilisateurs ou clients.
 - Effets de l'algorithme lui-même sur les utilisateurs.
 - Etc.



MLOps

Objectif du MLOps :

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



Principes du MLOps

- Automatisation du cycle de vie des modèles.
- Intégration continue et déploiement continu (CI/CD).
- Collaboration entre les équipes ML (data scientists) et les équipes opérations (Data engineer, ML engineers, Développeurs).
- Monitoring des modèles et des données en production.
- Versioning et traçabilité du code et des données.
- Optimisation et gestion des ressources, notamment sur le Cloud.

Outils

Versioning



Virtualisation



CI/CD



APIs



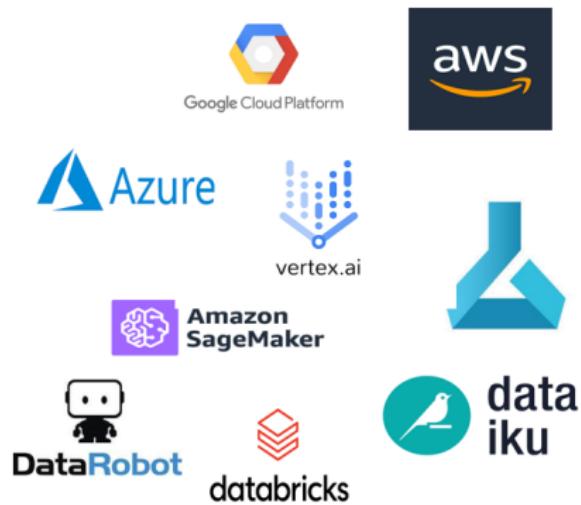
Cloud



Plateformes end-to-end

Les solutions de ML sont généralement déployées dans un environnement Cloud.

- AWS, GCP, Azure, etc.
- SageMaker, Vertex AI, Azure ML, etc.
- Dataiku, databricks, DataRobot, etc.



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.

Enjeux éthiques de l'IA

- **Enjeux sociétaux** : production massive de deepfakes, environnement peuplé par des artefacts, etc.
- **Alignement** : les valeurs encodées dans les modèles génératifs sont aujourd'hui déterminées par une poignée de personnes dans la Silicon Valley.
- **Transhumanisme** : à mesure que les modèles deviennent très puissants — voire plus puissants que l'intelligence humaine, la tentation de l'augmentation se fera plus pressante.
- **Enjeux ontologiques** : si toutes les tâches et facultés dont on pensait jusque-là qu'elles étaient le propre de l'Homme sont accessibles à l'intelligence artificielle, alors quelle est finalement l'essence de l'humain ?

Merci de votre attention

redha.moulla@axia-conseil.com