

Intelligence artificielle pour product owner

Redha Moulla

Septembre - octobre 2025

Plan de la formation

- Qu'est-ce que l'intelligence artificielle ?
- Machine learning
- Deep learning
- Vision par ordinateur
- Traitement automatique du langage
- Intelligence artificielle générative
- Industrialisation des modèles de machine learning
- Enjeux éthiques de l'IA

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 ?

Le mythe des servantes d'or

“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 proposed by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon. The proposal required 17 pages plus a title page. Copies of the reprints are bound in a volume of the *Proceedings of the Conference on Cognition and Computer*, published by Stanford University. The first 5 papers state the proposal and give the general goals of the study and interests of the four who proposed the study. In the interest of brevity, this article reprints the first 2 pages of the proposal, which contain the philosophical statements of the proposal.

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

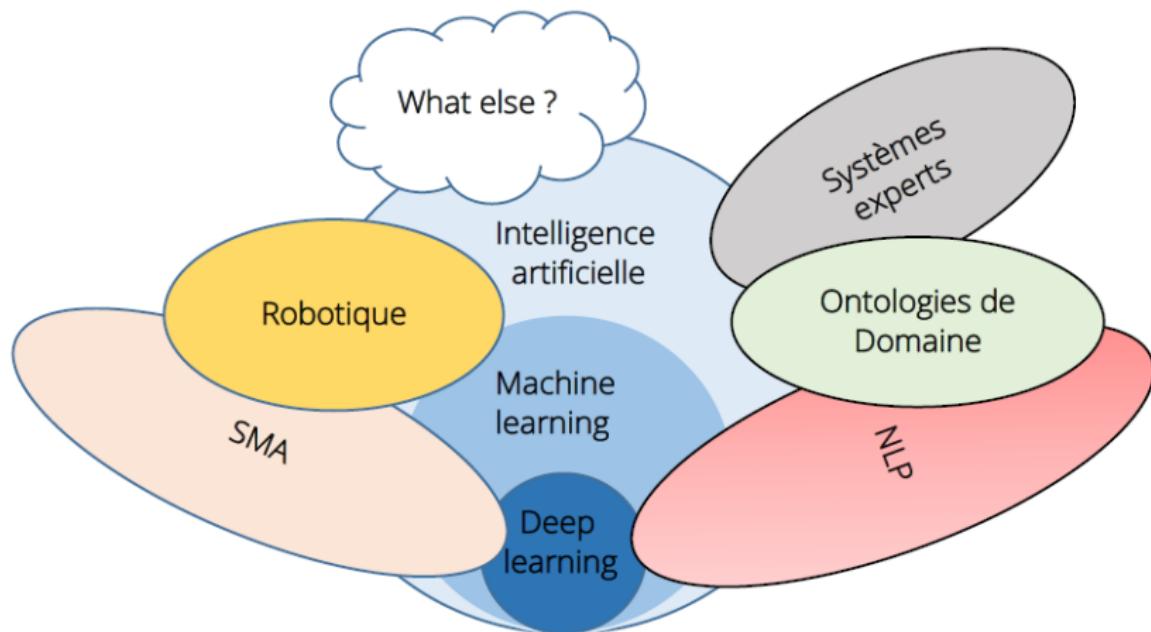
Le Test de Turing, développé par Alan Turing en 1950, est une tentative de mesurer l'intelligence d'une machine, plus précisément de la faculté d'une machine à penser. Cette dernière n'étant pas si évidente à mesurer, le test substitue finalement à la faculté de penser celle de traiter le langage naturel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

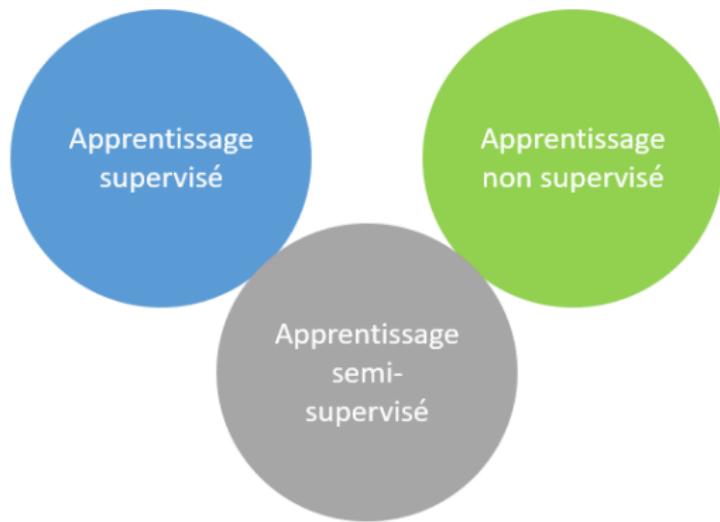
Définition de l'apprentissage automatique

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

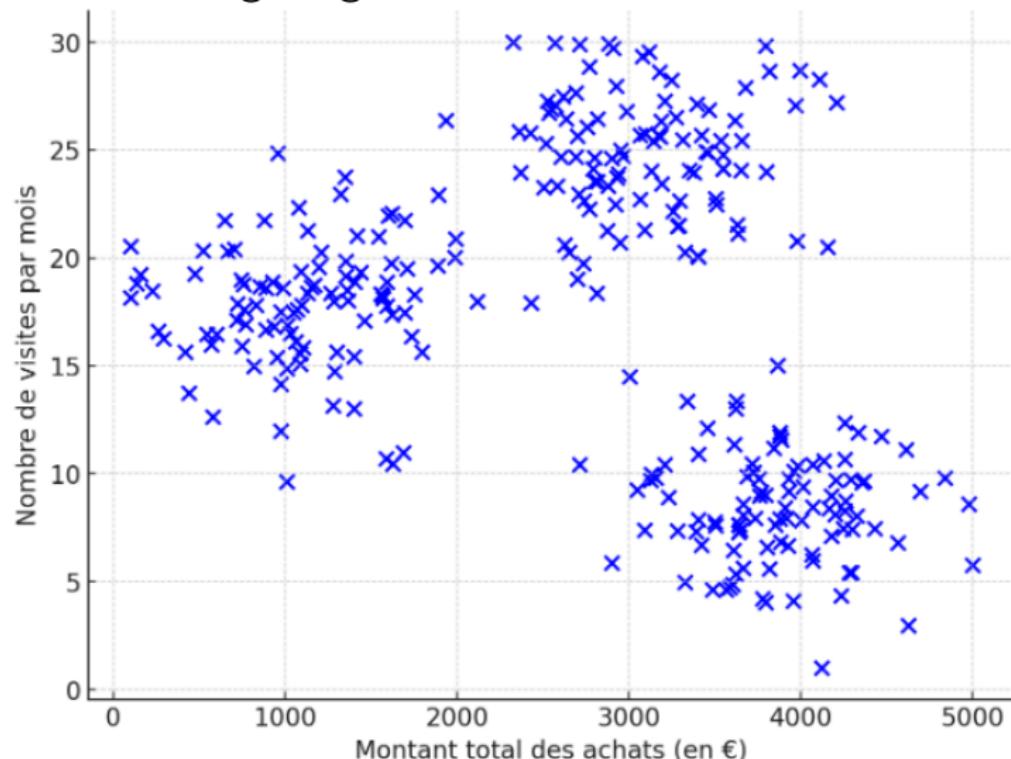
L'apprentissage non supervisé se réfère à l'utilisation de modèles d'apprentissage automatique pour identifier des patterns et des structures dans des données qui ne sont pas étiquetées.

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.

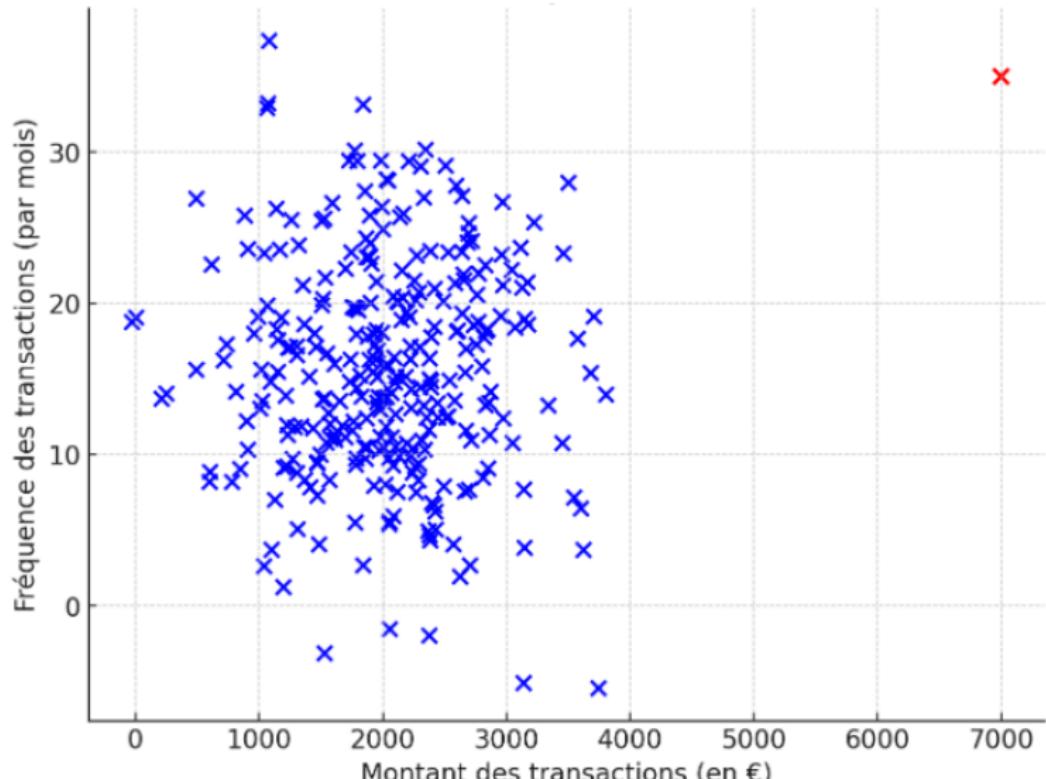
Clustering

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 :

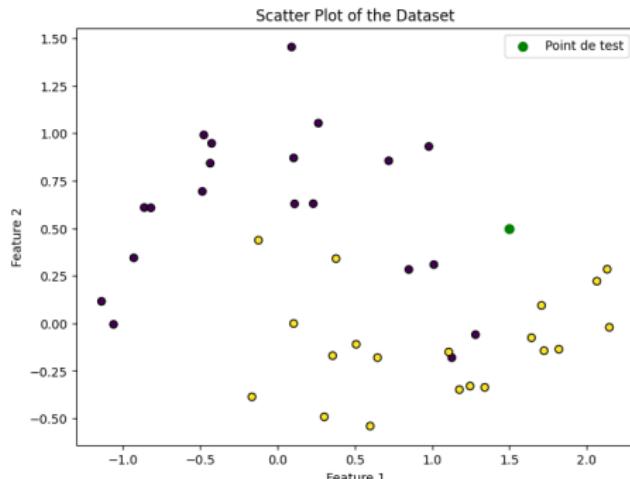
- Développement de systèmes de recommandation plus performants.
- Traitement de langage naturel et analyse de sentiment lorsque les annotations complètes ne sont pas disponibles.

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

Apprentissage supervisé

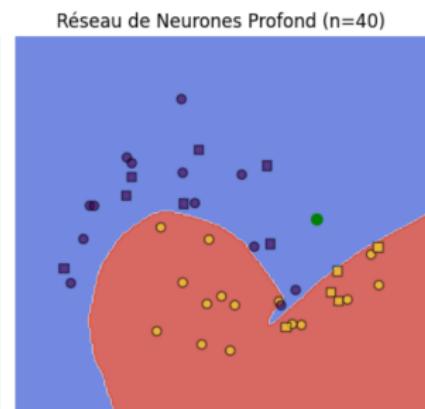
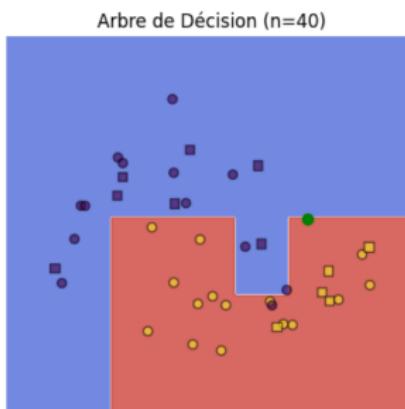
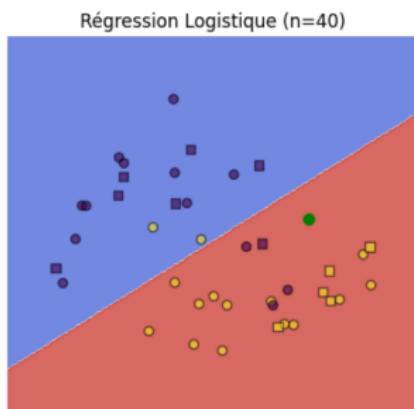
L'apprentissage supervisé comme un problème d'induction

- **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 ?



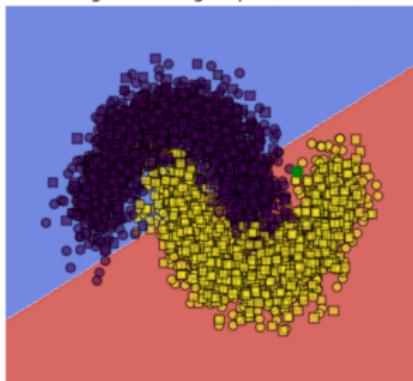
Une indétermination intrinsèque pour le choix du modèle

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

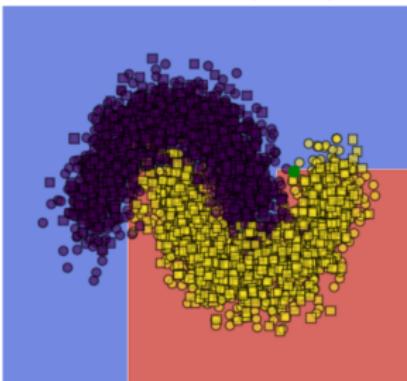


Sous-apprentissage et surapprentissage sur les données d'entraînement

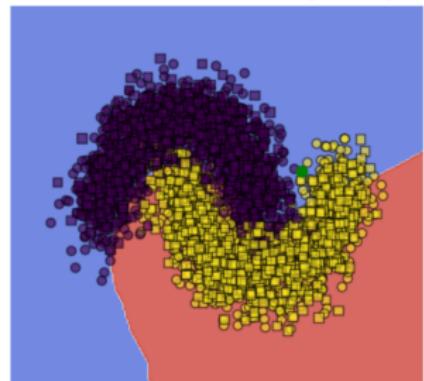
Régression Logistique (n=3000)



Arbre de Décision (n=3000)



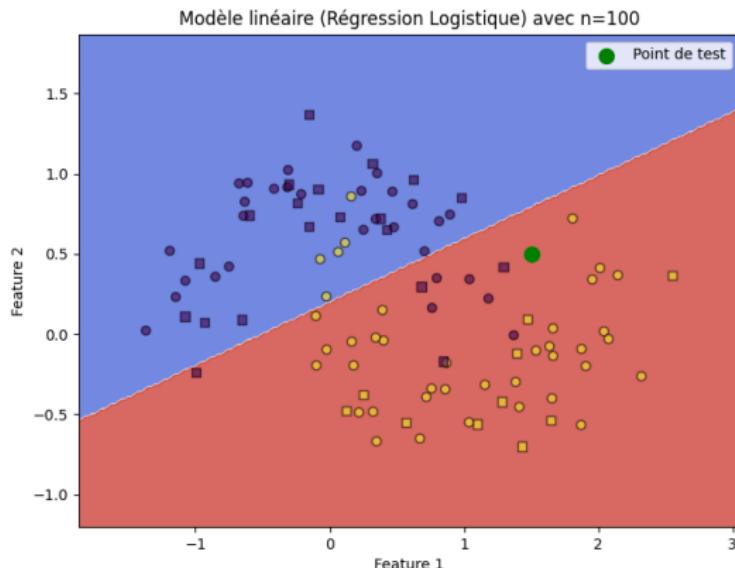
Réseau de Neurones Profond (n=3000)



Sous-apprentissage

Definition

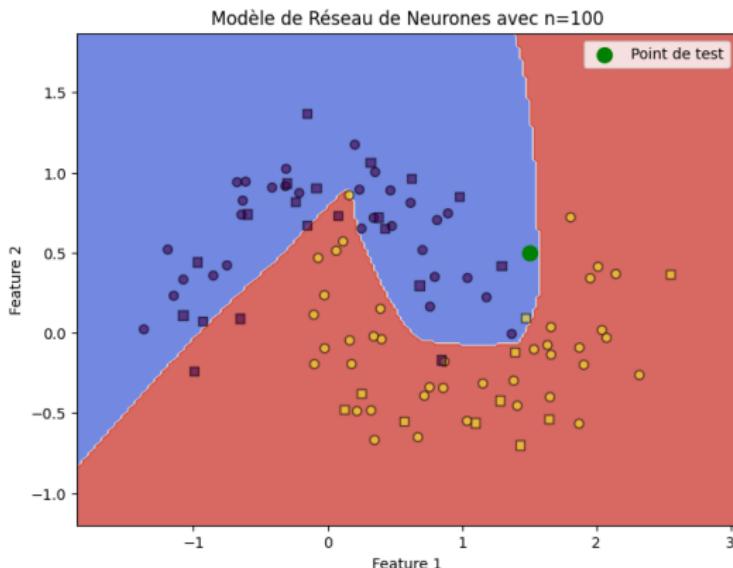
On dit qu'un modèle de machine learning est en régime de sous-apprentissage (underfitting) lorsqu'il n'arrive pas à capturer la complexité (l'information) présente dans le jeu de données d'entraînement.



Sur-apprentissage

Definition

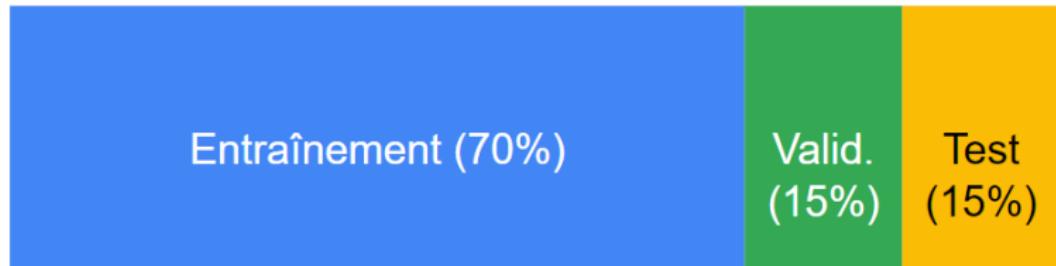
On dit qu'un modèle de machine learning est en régime de sur-apprentissage (overfitting) lorsqu'il n'arrive pas à généraliser à des données non encore observées, i.e. lorsqu'il est trop adapté aux données d'entraînement.



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.

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

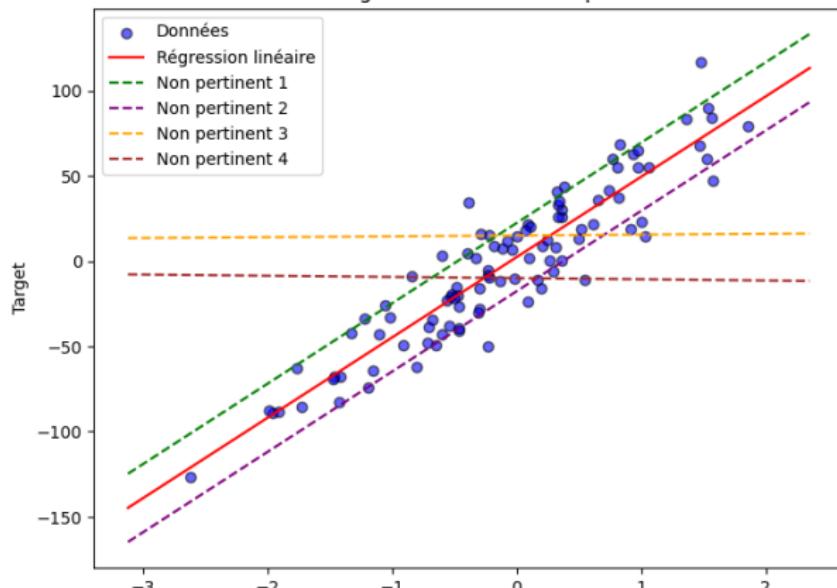
Modèles classiques de machine learning

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.

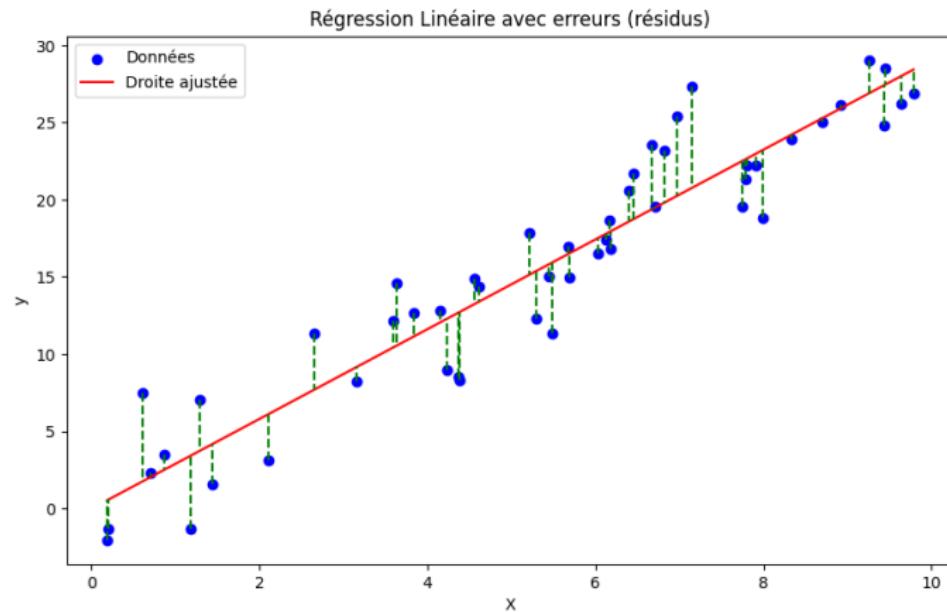
$$\hat{y} = \beta_0 + \beta_1 x$$

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
:	:

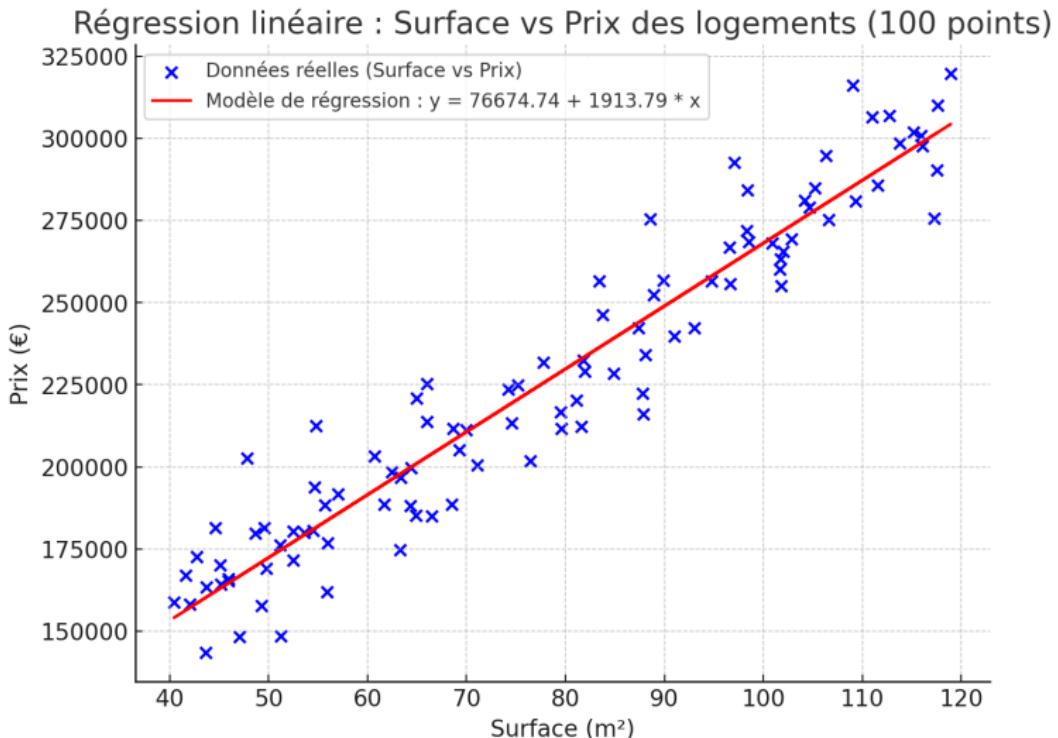
Les coefficients β_0 et β_1 sont calculés à partir des formules suivantes :

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

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

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 :

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

Choix de modèle

Il y a plusieurs manières d'évaluer la pertinence d'un modèle de régression linéaire. Le coefficient 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'abosrvations et k le nombre de variables.

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

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 :

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

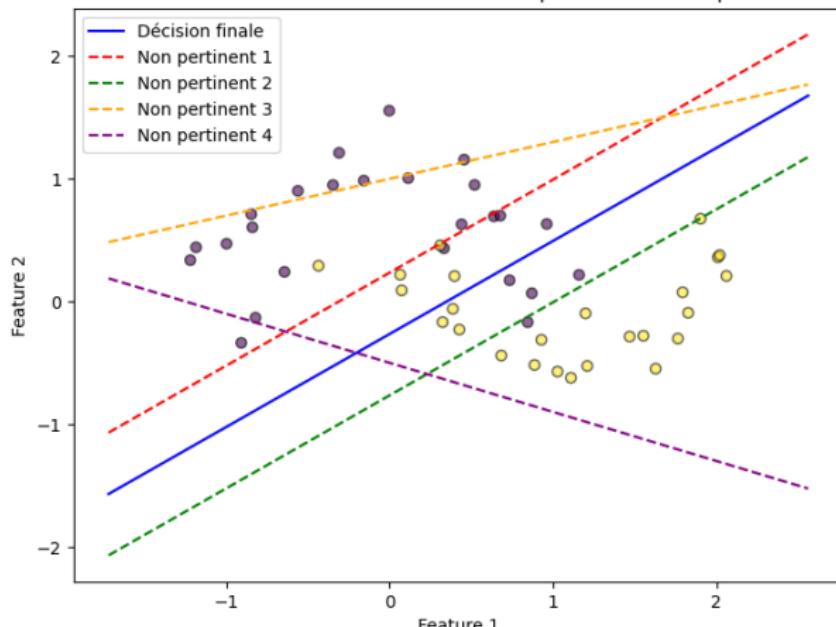
Les coefficients β_{ij} peuvent être de la même manière, avec la méthode des moindres carrés.

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).

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 :

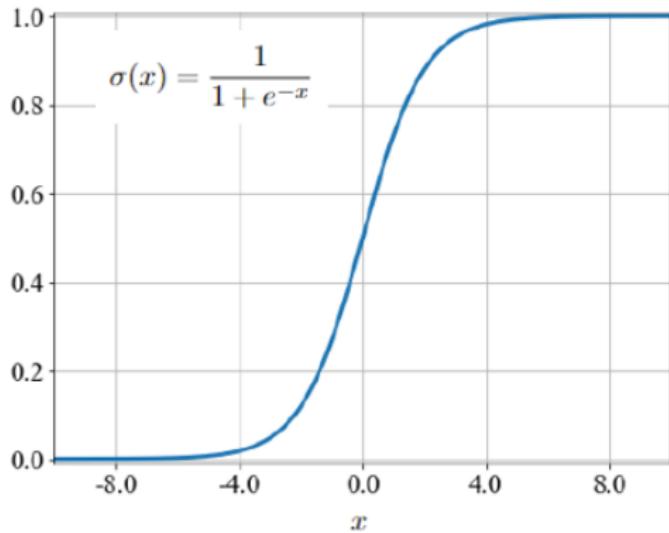
- **Variable dépendante** : On cherche la probabilité que la variable dépendante (y) appartienne à une classe (0 ou 1, vrai ou faux, succès ou échec). Autrement dit, on cherche à modéliser $P(y = 1)$ en fonction des variables dépendantes (explicatives) x .
- **Odds ratio** : Plus concrètement, on cherche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

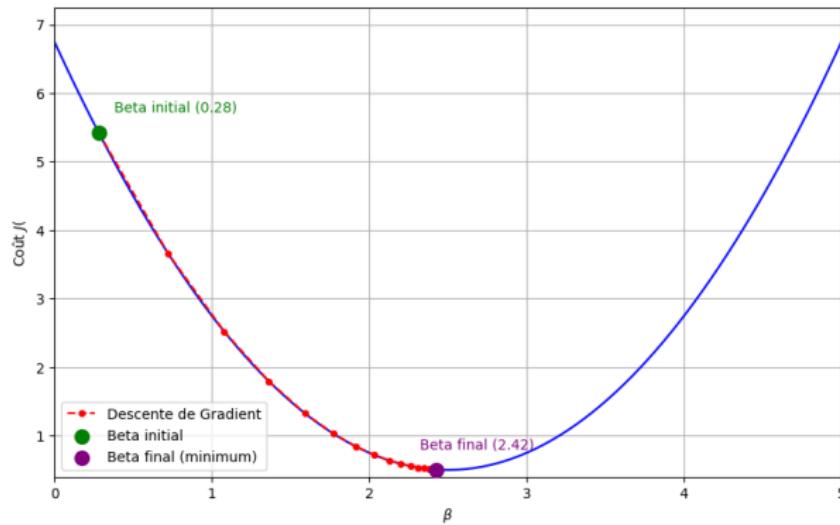
$$p(\mathbf{x}) = \frac{1}{1 + e^{-(\beta^T \mathbf{x})}}$$



Calcul des coefficients de la régression logistique

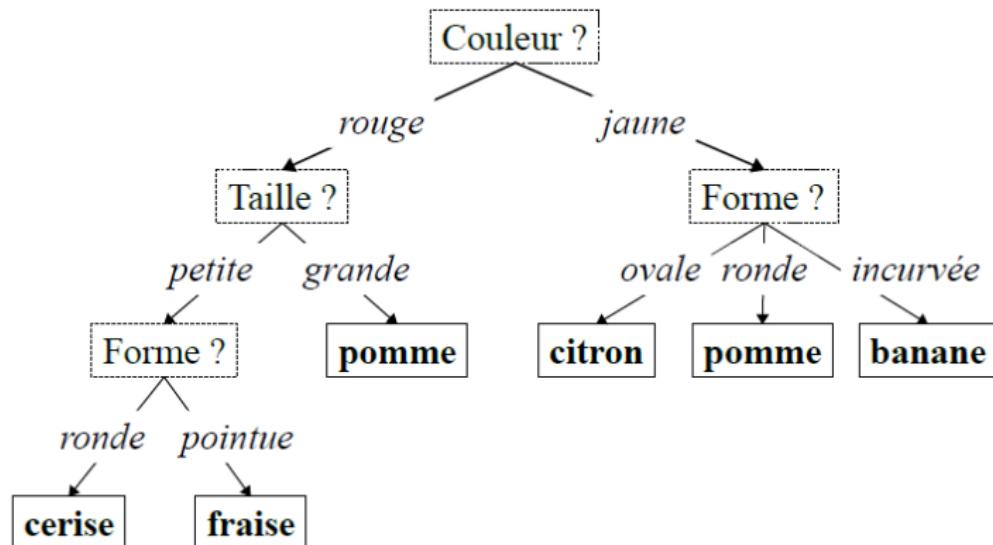
Les coefficients de la régression logistique peuvent être calculés en minimisant le risque empirique par rapport à une fonction de coût sous forme d'entropie croisée :

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



Arbres de décision

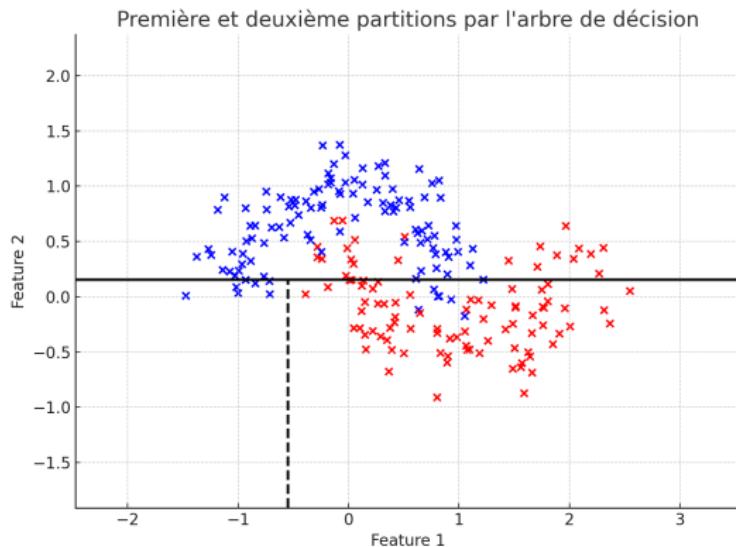
Les arbres de décisions sont des modèles dont le processus de décision est hiérarchique et prend la forme d'un arbre.



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).

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 :

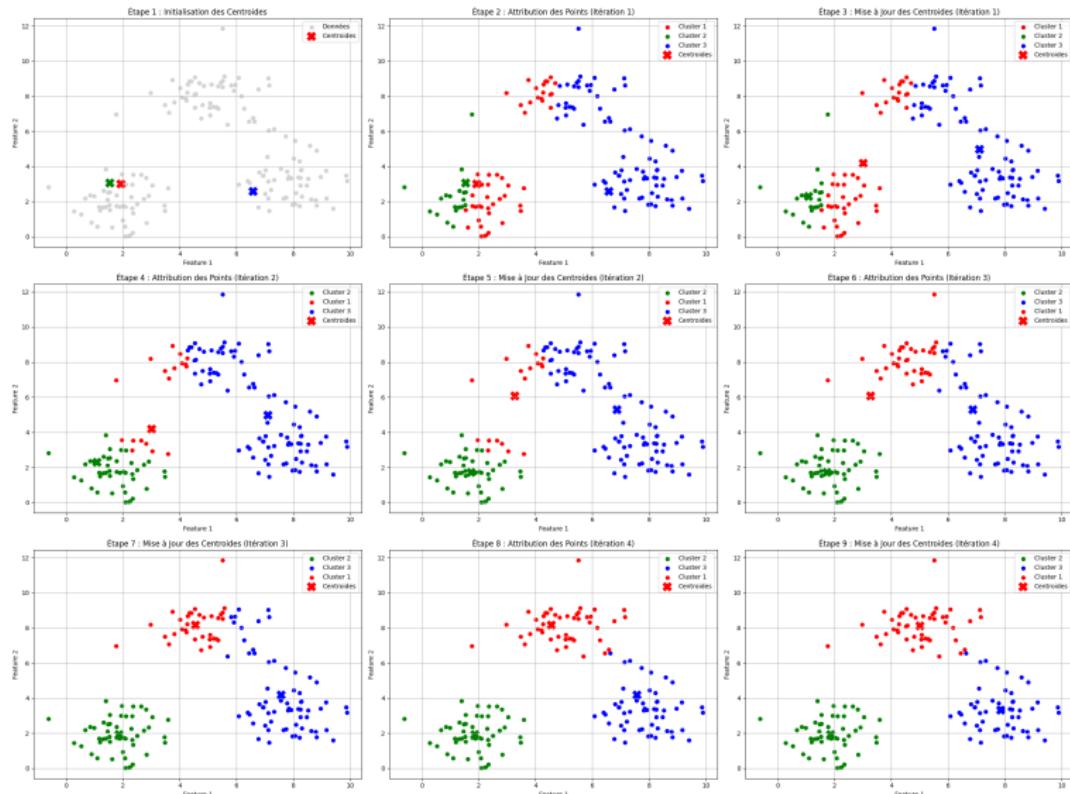
- **Clustering (partitionnement)** : cela consiste à partitionner les n observations en K groupes pertinents (généralement le critère de pertinence a une signification d'un point de vue métier).
- **Détection d'anomalie** : il s'agit de détecter des observations qui présentent un profil (des features) inhabituels 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

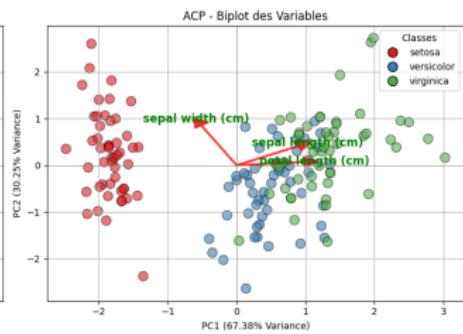
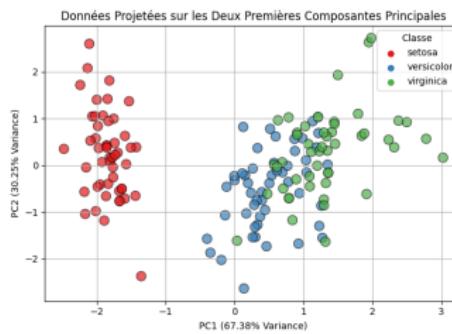
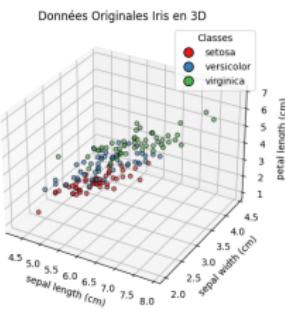


Remarques

- 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 .

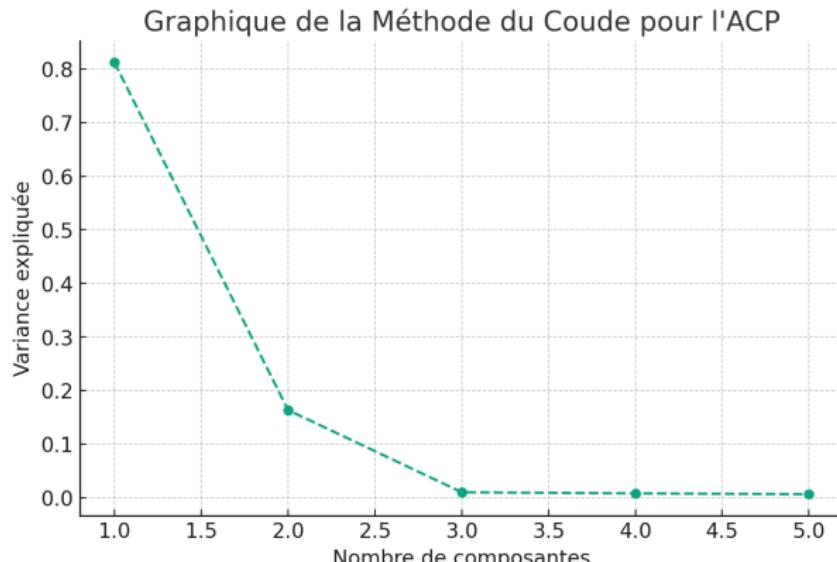
La contribution de chaque composante principale à la variance totale est donnée par :

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

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.

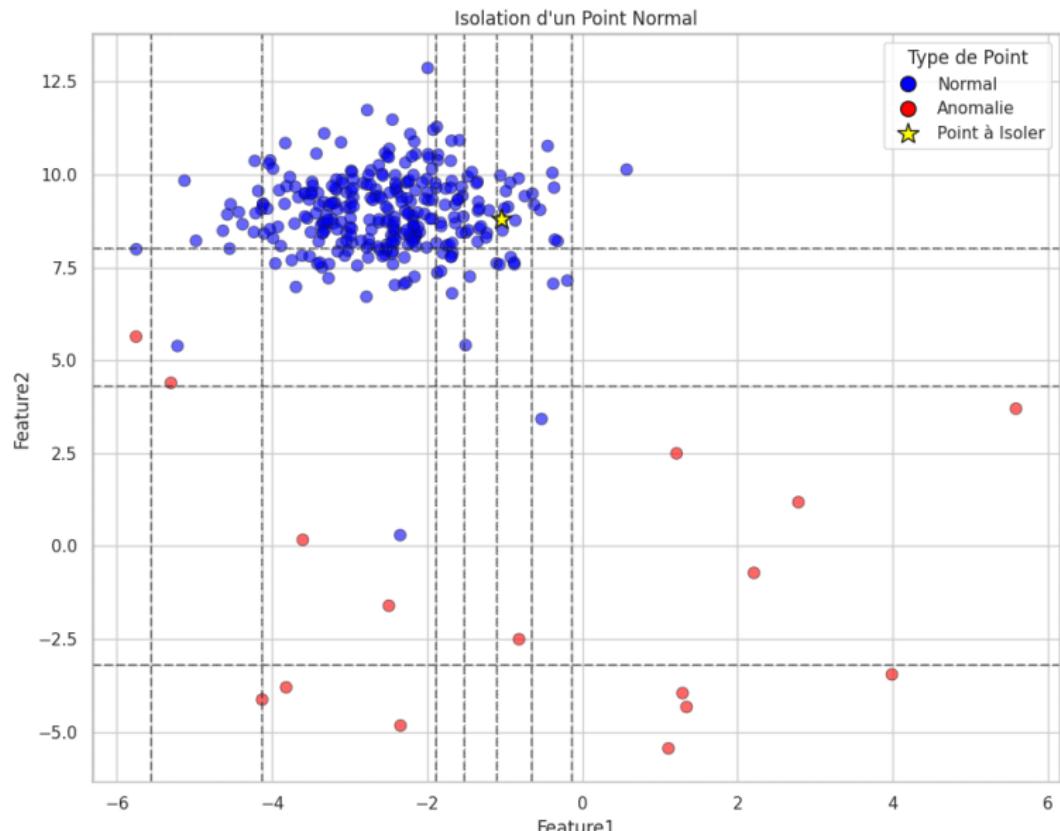
- Fonctionne en construisant des arbres d'isolement à partir de sous-ensembles de données.
- Isoler une observation signifie la séparer des autres par des divisions aléatoires de l'espace des caractéristiques.
- Moins de divisions sont nécessaires pour isoler une anomalie, ce qui constitue le fondement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

Les arbres d'isolement sont construits de manière récursive :

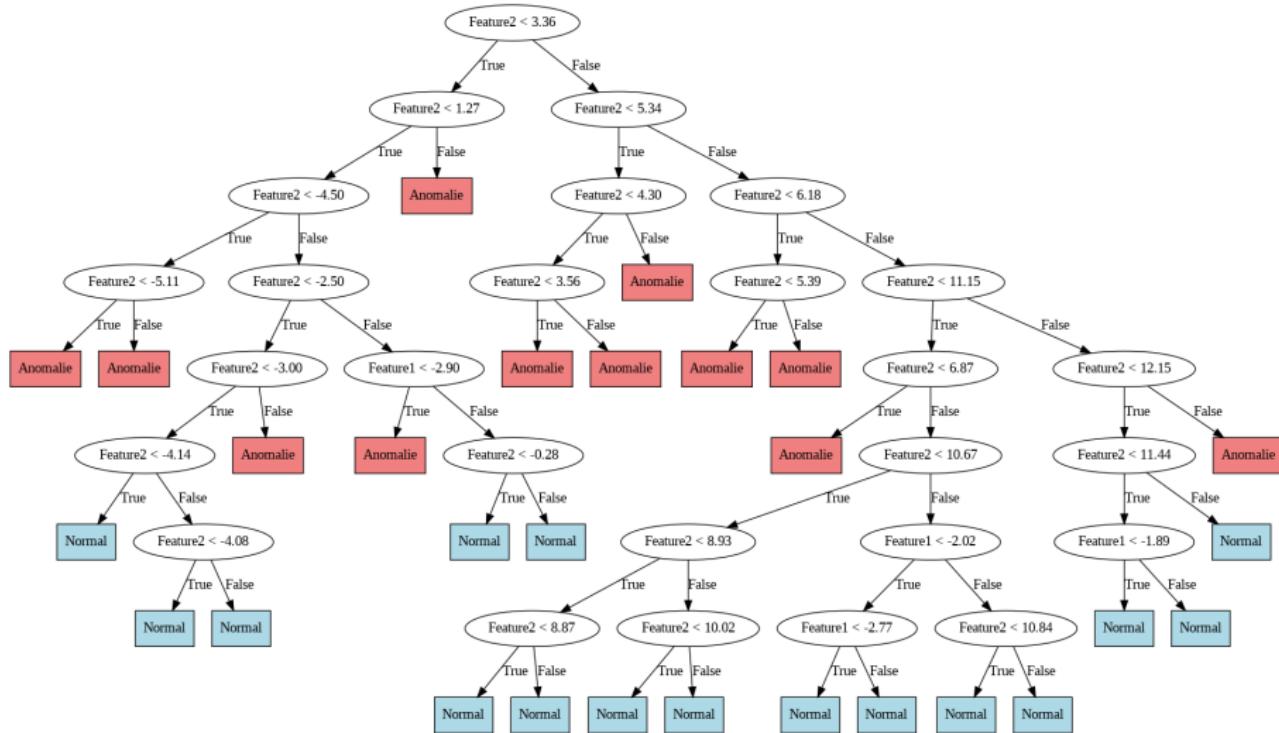
- ➊ Sélection aléatoire d'un sous-ensemble de données.
- ➋ Choix aléatoire d'une caractéristique et d'une valeur de seuil pour diviser le sous-ensemble.
- ➌ Répétition des divisions jusqu'à l'isolement des observations ou atteinte d'une limite de profondeur prédéfinie.

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

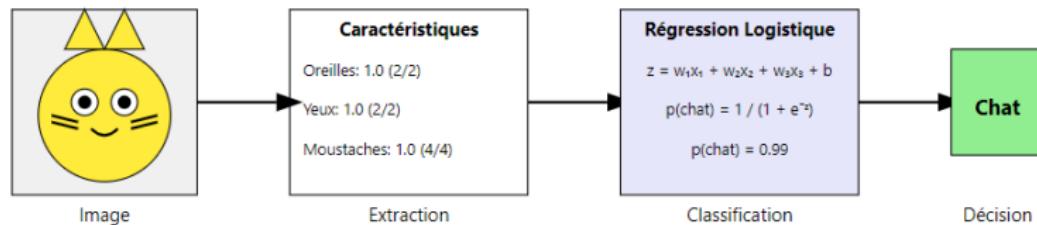
Définition : Les réseaux de neurones sont des modèles computationnels inspirés par le fonctionnement des neurones dans le cerveau humain. Ils sont capables d'apprendre des tâches complexes en modélisant des relations non linéaires entre les entrées et les sorties.

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

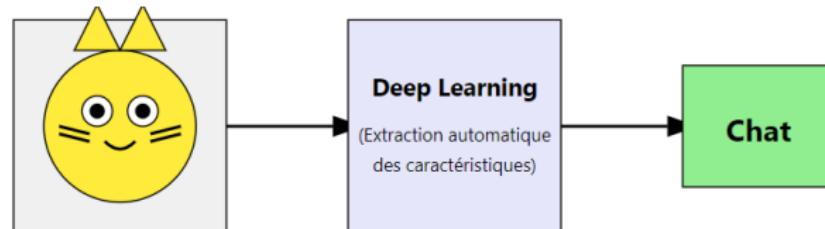
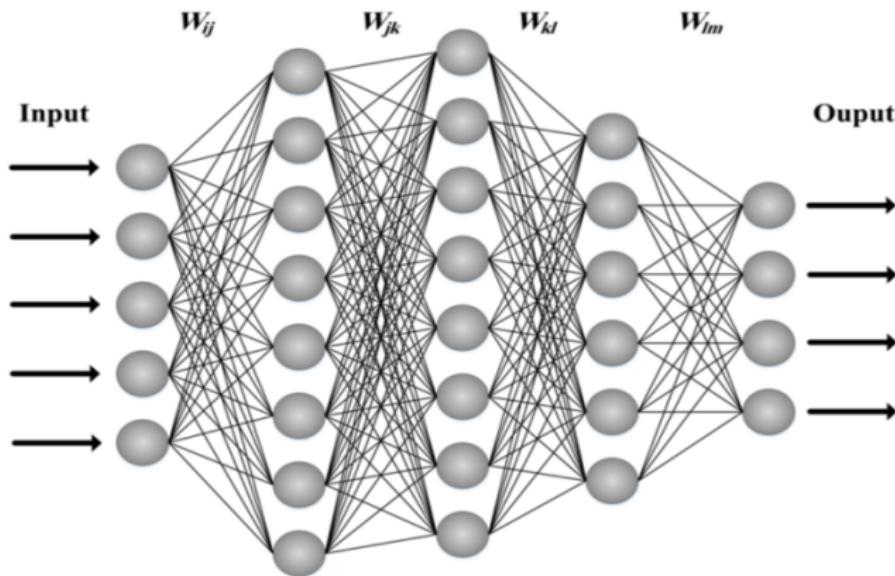
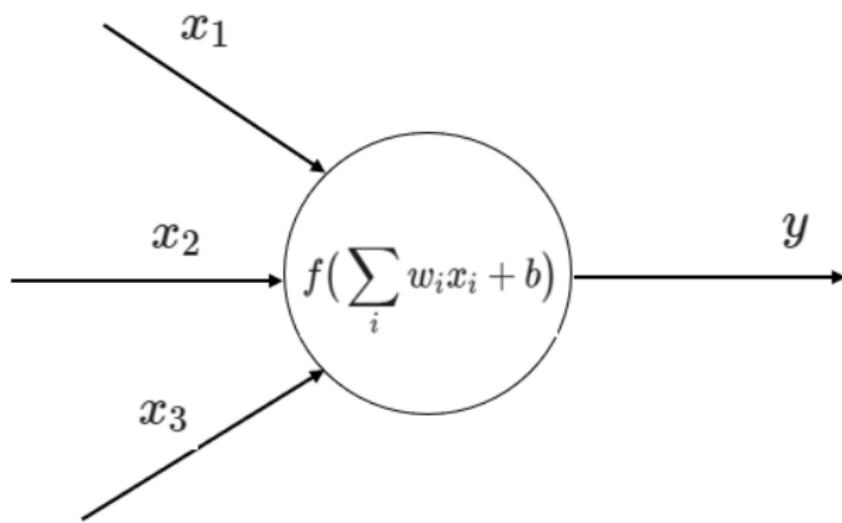


Illustration d'un réseau de neurones classique



Neurone aritificial



Fonctions d'activation

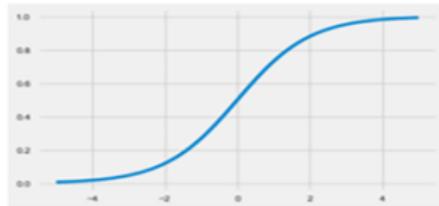
Les fonctions d'activation permettent aux modèles d'apprendre des relations plus complexes en capturant les non-linéarités dans les données.

- **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.

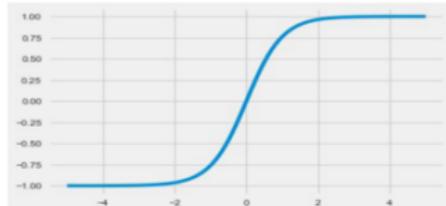
$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Illustration des fonctions d'activation

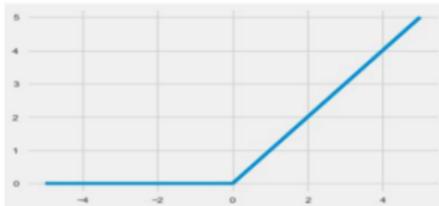
Sigmoïde



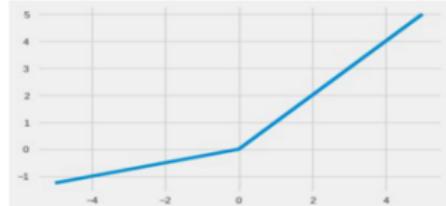
Tanh



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) :

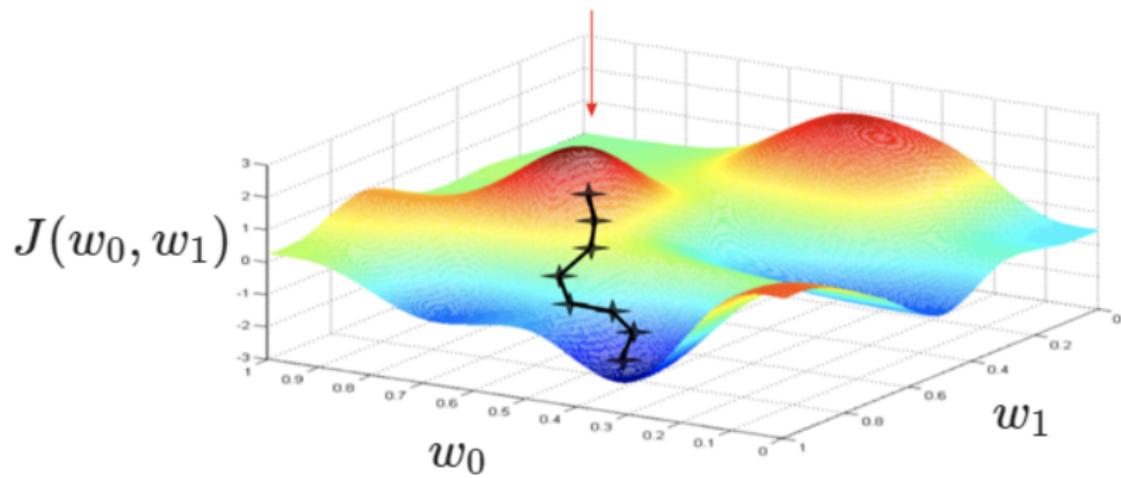
- Méthode d'optimisation utilisée pour mettre à jour les poids du réseau de manière itérative.
- À chaque itération, un sous-ensemble (batch) de données est utilisé pour calculer le gradient de la fonction de coût.
- Les poids sont mis à jour dans la direction opposée du gradient pour réduire l'erreur.

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

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 $n - 1$:

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

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.

Computer vision

Introduction à la vision par ordinateur

La vision par ordinateur est une discipline de l'informatique qui permet aux ordinateurs de déduire des informations à partir d'images numériques, de vidéos et d'autres entrées visuelles similaires. Elle simule la capacité humaine de voir et de comprendre le contenu visuel.

- **Objectifs :**

- Interpréter et comprendre le contenu visuel de manière automatisée.
- Extraire et analyser des informations des données visuelles.

- **Applications :**

- Classification d'images.
- Détection d'objets.
- Segmentation d'images.
- Réduction de la dimensionnalité et débruitage.
- Génération d'images.

Introduction aux réseaux de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs (CNN) sont une catégorie de réseaux de neurones profonds, spécialement efficaces pour l'analyse d'images. Ils sont largement utilisés pour des tâches de vision par ordinateur telles que la classification d'images, la reconnaissance de motifs, et la détection d'objets.

- **Caractéristiques clés :**

- Utilisation de convolutions pour extraire des caractéristiques de l'image.
- Capacité à capturer des hiérarchies de caractéristiques spatiales et temporelles.

- **Importance :**

- Efficacité dans le traitement des données visuelles.
- Réduction de la dimensionnalité, tout en préservant les caractéristiques essentielles.

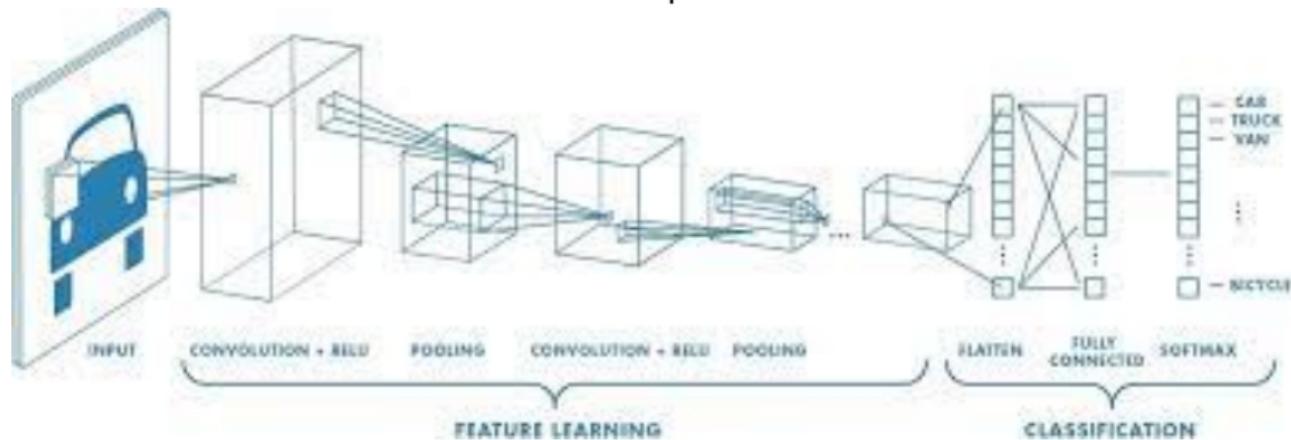
Architecture basique d'un CNN

Les CNNs sont structurés en plusieurs couches qui transforment progressivement l'entrée (image) pour aboutir à une sortie (décision).

- ① **Couche de convolution** : Applique des filtres pour extraire des caractéristiques de bas niveau comme les bords et les textures.
- ② **Couche de pooling (ou Sous-échantillonnage)** : Réduit la dimensionnalité de chaque carte de caractéristiques tout en préservant les informations les plus importantes.
- ③ **Couche de normalisation** : Normalise les activations de la couche précédente, souvent pour améliorer la convergence.
- ④ **Couches entièrement Connectées (Fully Connected)** : Utilisées en fin de réseau pour classer les images en fonction des caractéristiques extraites.

Architecture d'un CNN

Le fonctionnement d'un CNN peut être visualisé comme un enchaînement de transformations où chaque couche traite et transforme les informations provenant de la couche précédente.



Fonctionnement des filtres dans un CNN 1/2

Qu'est-ce qu'un filtre ? Un filtre (ou noyau) est une matrice de poids utilisée pour extraire des caractéristiques locales d'une image (bords, textures, motifs). Il est appliqué à l'image via une **opération de convolution**.

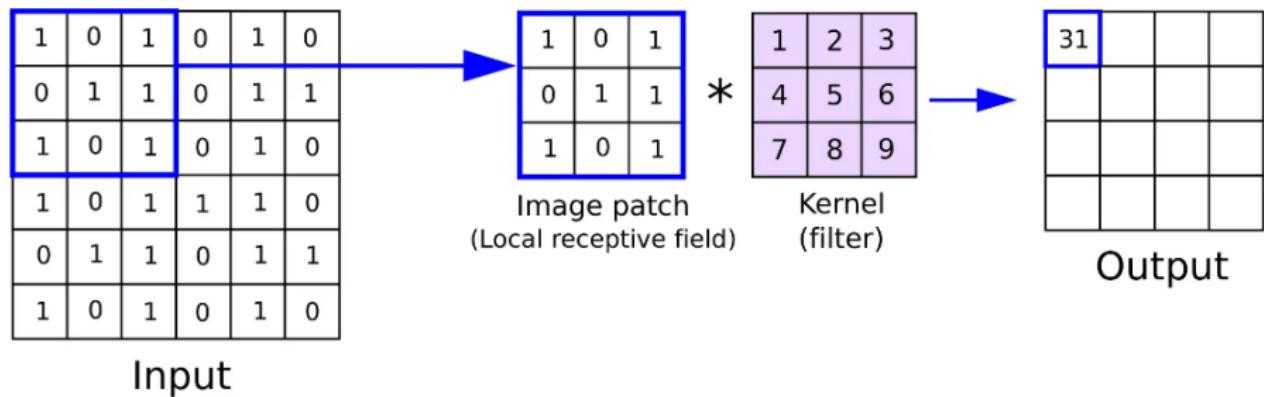
Processus de convolution :

- ① Un filtre de taille $k \times k$ glisse sur l'image.
- ② À chaque position, il effectue un produit scalaire avec la région correspondante de l'image.
- ③ La somme des produits donne une valeur qui constitue un pixel dans la **feature map**.
- ④ Plusieurs filtres sont utilisés pour extraire différentes caractéristiques.

Hyperparamètres influents :

- **Taille du filtre** ($k \times k$) : typiquement 3×3 ou 5×5
- **Stride** : pas de déplacement du filtre (1, 2... pixels)
- **Padding** : ajout de bordures pour contrôler la taille de sortie

Fonctionnement des filtres d'un CNN 2/2



Pourquoi utiliser plusieurs filtres dans un CNN ?

Principe Dans un CNN, chaque couche de convolution applique plusieurs filtres simultanément. Chaque filtre apprend à détecter un motif spécifique (**bords, textures, formes, objets**).

Pourquoi plusieurs filtres ?

- Capturer différentes caractéristiques d'une image (ex. bords horizontaux, diagonaux, textures).
- Obtenir une représentation plus riche et discriminante.
- Augmenter la profondeur de l'apprentissage sans ajouter trop de paramètres.

Comment cela fonctionne ?

- Chaque filtre produit une **feature map** en appliquant une convolution sur l'image d'entrée.
- Toutes les feature maps sont ensuite empilées pour former la sortie de la couche.
- À chaque nouvelle couche, les filtres combinent les informations des couches précédentes.

Fonctionnement du pooling dans les CNN

Pourquoi le pooling ? Le pooling est une opération qui réduit la dimensionnalité des feature maps tout en conservant les informations importantes. Il permet de :

- Réduire le nombre de paramètres et de calculs
- Rendre le réseau plus robuste aux translations et variations locales
- Prévenir le sur-apprentissage (overfitting)

Types de pooling :

- **Max pooling** : conserve la valeur maximale dans chaque région du filtre.
- **Average pooling** : calcule la moyenne des valeurs dans chaque région.

Hyperparamètres influents :

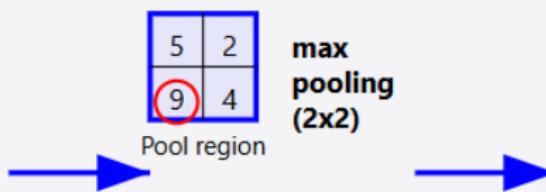
- **Taille du filtre** ($k \times k$) : ex. 2×2 ou 3×3
- **Stride** : contrôle le déplacement du filtre (souvent égal à la taille du filtre)

Impact sur l'architecture CNN : - Réduit la taille des feature maps → moins de mémoire et de calculs - Maintient les caractéristiques les plus importantes - Supprime des détails moins pertinents, favorisant l'invariance spatiale

Fonctionnement du pooling

5	2	7	1	8	3
9	4	6	2	7	5
3	8	2	9	4	1
7	6	5	8	3	2
1	5	9	4	7	6
4	2	8	3	6	1

Input



9	7	8
8	9	4
5	9	7

Output

La normalisation dans les CNN

Pourquoi la normalisation ? La normalisation permet de stabiliser l'entraînement des réseaux de neurones et d'accélérer la convergence en contrôlant la distribution des activations. Elle aide à :

- Réduire le décalage de covariances interne (internal covariate shift).
- Accélérer l'apprentissage en permettant des taux d'apprentissage plus élevés.
- Améliorer la généralisation et réduire l'overfitting.

Types de normalisation :

- **Batch Normalization (BatchNorm)** Normalise les activations par mini-batch pour assurer une distribution stable.
- **Layer Normalization (LayerNorm)** Appliquée sur chaque exemple indépendamment en normalisant sur les neurones d'une couche.
- **Instance Normalization (InstanceNorm)** Similaire à LayerNorm mais appliquée indépendamment sur chaque canal d'une image.
- **Group Normalization (GroupNorm)** Divise les canaux en groupes et applique une normalisation à chaque groupe.

Batch Normalization - Formule : Pour une activation x_i dans un mini-batch :

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Les couches fully connected dans les CNN

Qu'est-ce qu'une couche fully connected ? Une couche fully connected (FC) est une couche où chaque neurone est connecté à tous les neurones de la couche précédente. Elle permet de transformer les caractéristiques extraites en une sortie interprétable (ex. classification).

Fonctionnement :

- Prend en entrée un vecteur aplati (**flatten**) issu des dernières feature maps d'un CNN.
- Applique une transformation linéaire :

$$y = Wx + b$$

où W est la matrice des poids et b le biais.

- Applique une **fonction d'activation** (ex. ReLU, softmax).

Pourquoi utiliser des couches fully connected ?

- Convertir les features extraites en classes ou valeurs de sortie.
- Capturer des combinaisons complexes des caractéristiques apprises.

Inconvénients :

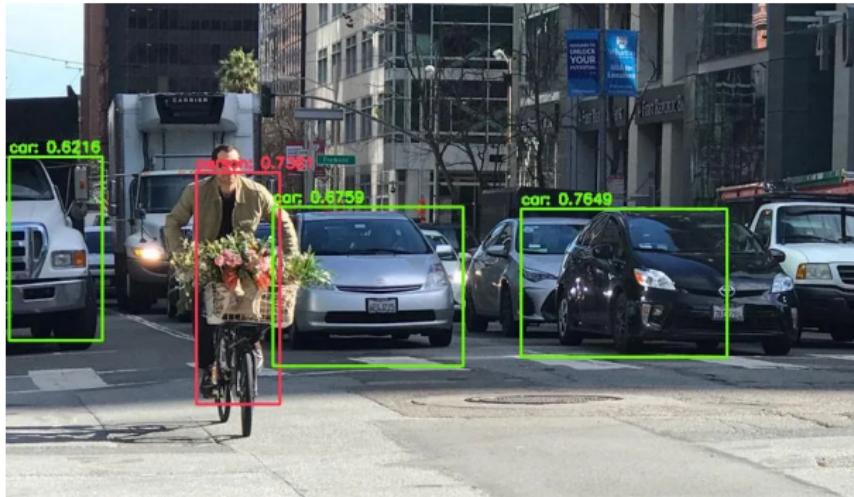
- Nombre élevé de paramètres → risque d'overfitting.
- Perte d'informations spatiales importantes.

Optimisations : Utilisation de dropout pour éviter le sur-apprentissage

Introduction à YOLO (You Only Look Once)

YOLO est un système de détection d'objets révolutionnaire car il traite l'image entière lors d'une seule évaluation de réseau, ce qui lui permet de détecter les objets en temps réel.

- **Principe :** YOLO divise l'image en une grille et prédit les boîtes englobantes et les probabilités de classe pour chaque grille simultanément.
- **Avantages :** Extrêmement rapide par rapport à d'autres détecteurs qui proposent des régions puis classifient (comme R-CNN).



Entraînement de YOLO pour la détection d'objets

L'entraînement de YOLO est un processus crucial pour assurer une détection efficace et précise. Il comprend plusieurs étapes et paramètres importants à configurer.

- **Préparation des données :**

- Collecte d'un ensemble de données d'images annotées avec des boîtes englobantes autour des objets.
- Normalisation des images et des coordonnées des boîtes pour les adapter à l'entrée du réseau.

- **Architecture du réseau :**

- Utilisation de Darknet, un réseau de neurones convolutionnel spécialement conçu pour YOLO.
- Configuration de différentes couches pour extraire et apprendre des caractéristiques à différentes échelles.

- **Fonction de perte :**

- Fonction de perte composite qui inclut la perte pour les erreurs de prédiction des classes, des boîtes englobantes et de la confiance des objets.
- Optimisation pour réduire les fausses détections et améliorer la localisation précise des objets.

Avantages et Limitations de YOLO

Bien que YOLO soit extrêmement rapide et efficace, il présente certaines limitations qui sont importantes à comprendre.

- **Avantages :**

- Très rapide, permettant une détection en temps réel.
- Moins de faux positifs sur les fonds d'images.

- **Limitations :**

- Moins précis sur les petits objets par rapport à d'autres modèles comme Faster R-CNN.
- Peut avoir des difficultés avec les objets qui apparaissent groupés.

Introduction à la segmentation d'images avec U-Net

U-Net est une architecture de réseau de neurones convolutifs développée principalement pour la tâche de segmentation d'images médicales. Elle est caractérisée par sa structure en forme de "U", optimisée pour travailler avec moins de données d'entraînement mais avec une grande précision de localisation.

- **Origine :** Initialement conçu pour la segmentation d'images biomédicales.
- **Caractéristique clé :** Utilise un chemin contractant pour capturer le contexte et un chemin expansif symétrique pour permettre une localisation précise.

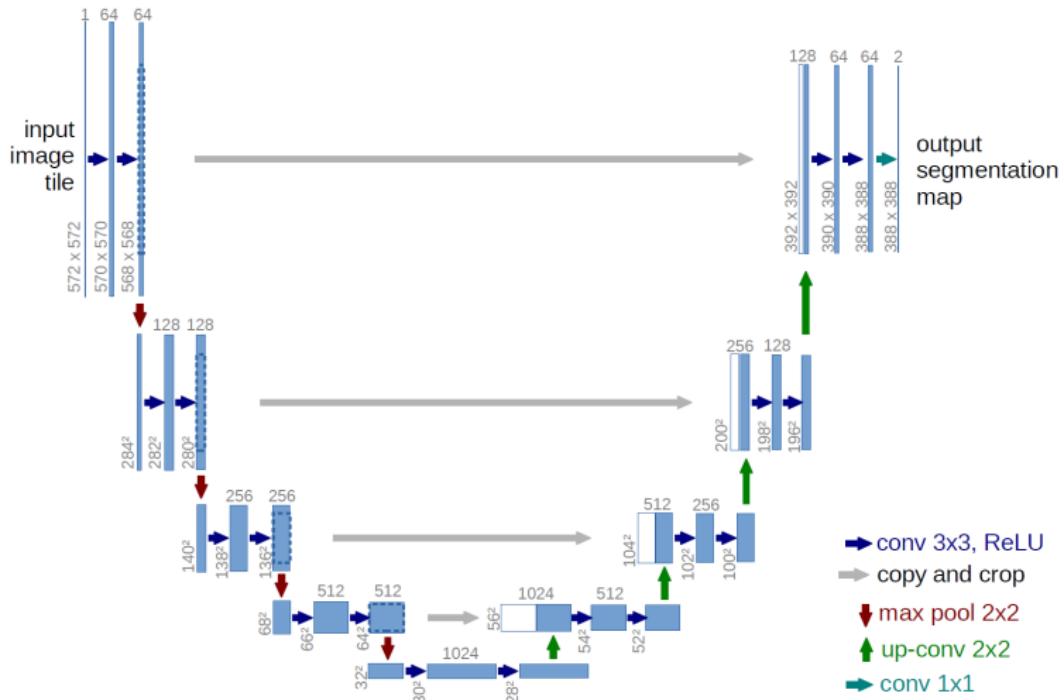
Architecture de U-Net

L'architecture de U-Net comprend deux parties principales: un encodeur (contraction) et un décodeur (expansion).

- **Encodeur :** Composé de couches de convolution suivies de max pooling pour la réduction de dimensionnalité, capturant les caractéristiques pertinentes à différentes échelles.
- **Décodeur :** Comprend des opérations de convolutions transposées pour agrandir les cartes de caractéristiques, et des connexions concaténées depuis l'encodeur pour récupérer les informations spatiales.

Fonctionnement de U-Net

U-Net prédit un masque pour chaque pixel d'une image, indiquant l'appartenance à une classe spécifique, ce qui est particulièrement utile pour les tâches de segmentation fine où chaque pixel importe.



Avantages et limitations de U-Net

U-Net est hautement apprécié pour sa performance en segmentation, mais il présente aussi quelques défis.

- **Avantages :**

- Excellente performance sur les petites quantités de données.
- Haute précision des contours grâce aux connexions de saut qui aident à récupérer des informations contextuelles.

- **Limitations :**

- Principalement conçu pour des images 2D, ce qui peut limiter son application directe à des images 3D sans modifications.
- Nécessite une quantité relativement importante de mémoire et de puissance de calcul en raison de la grande taille des cartes de caractéristiques dans le chemin expansif.

Applications de U-Net en Vision par Ordinateur

Bien que développé pour la segmentation médicale, U-Net a trouvé des applications dans divers autres domaines nécessitant une segmentation précise d'images.

- **Applications :**

- Détection de défauts dans les matériaux industriels.
- Segmentation de routes et de bâtiments dans les images satellitaires.
- Applications en agriculture pour le suivi des cultures.

Introduction aux réseaux de neurones siamois

Les réseaux siamois sont une architecture spéciale de réseaux de neurones conçue pour comparer deux entrées et déterminer leur similitude. Ils sont particulièrement efficaces pour des tâches telles que la vérification d'identité et la comparaison d'images.

- **Principe :** Utilise deux réseaux neuronaux identiques, qui partagent les mêmes poids, pour extraire les caractéristiques de chaque entrée.
- **Utilisation typique :** Comparaison de paires d'images pour déterminer si elles sont similaires ou non, basée sur l'apprentissage par paires ou triplets.

Architecture des Réseaux Siamois

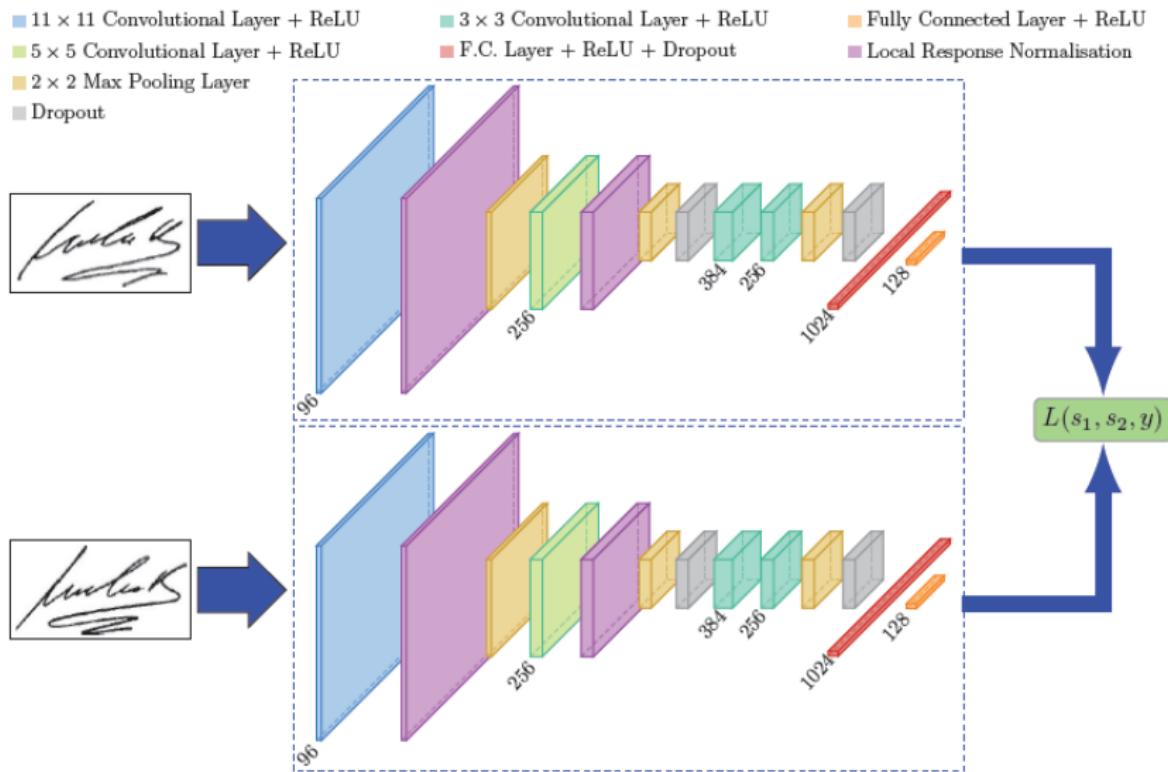
L'architecture siamoise se compose de deux réseaux identiques qui convergent en un point pour comparer leurs caractéristiques. Chaque branche encode une entrée dans un espace de caractéristiques qui permet de mesurer des distances.

- **Composants :**

- Deux réseaux de neurones identiques.
- Une couche qui calcule une distance ou une métrique entre les représentations obtenues.

- **Métrique de distance :** Souvent, la distance euclidienne ou la distance de Manhattan est utilisée pour calculer la similarité entre les représentations.

Architecture des réseaux siamois



Avantages et limitations des réseaux siamois

Les réseaux siamois offrent plusieurs avantages, mais ils ont également des limitations spécifiques à prendre en compte.

- **Avantages :**

- Très efficaces pour apprendre des similitudes ou des différences fines entre deux entrées.
- Peuvent être entraînés avec relativement peu de données grâce à l'apprentissage par comparaison.

- **Limitations :**

- Le choix de la métrique de distance est crucial et peut affecter les performances.
- Peuvent nécessiter un temps de convergence long en raison de la complexité de l'apprentissage des similitudes.

Applications des réseaux siamois

Les réseaux siamois ont une variété d'applications en vision par ordinateur, allant au-delà de la simple comparaison d'images.

- **Applications :**

- Vérification de l'identité dans les systèmes biométriques.
- Reconnaissance de visages même avec des variations de pose ou d'illumination.
- Suivi d'objets en vidéo où les modèles doivent reconnaître un objet au fil du temps.

Introduction aux Generative Adversarial Networks (GANs)

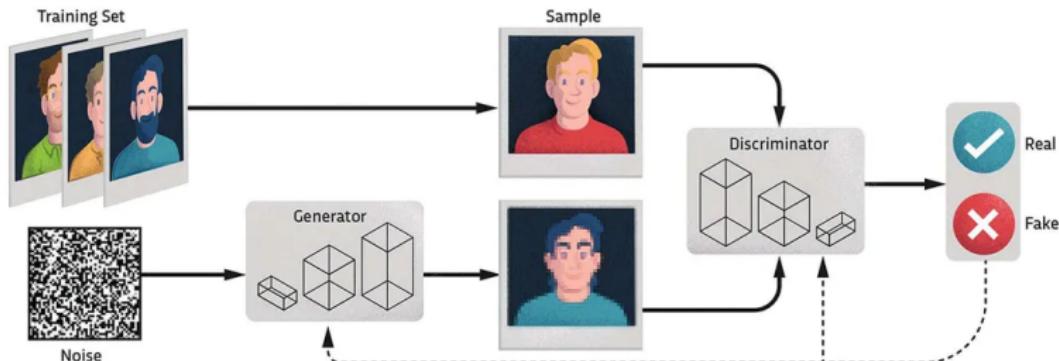
Les Generative Adversarial Networks (GANs) sont une classe de systèmes d'apprentissage automatique où deux réseaux neuronaux se livrent à un jeu compétitif. L'un génère des données (générateur) et l'autre évalue leur authenticité (discriminateur).

- **Fonctionnement** : Le générateur tente de créer des données suffisamment réalistes pour tromper le discriminateur, tandis que le discriminateur apprend à distinguer les fausses données des vraies.
- **Importance** : Les GANs sont particulièrement puissants pour générer des images réalistes et peuvent être utilisés pour améliorer les modèles de reconnaissance d'images.

Architecture des GANs

L'architecture d'un GAN se compose de deux parties principales : le générateur et le discriminateur.

- **Générateur** : Prend une entrée de bruit aléatoire et construit des données (par exemple, des images).
- **Discriminateur** : Reçoit soit les données générées, soit les vraies données et doit déterminer si elles sont authentiques ou non.



Avantages et limitations des GANs

Les GANs offrent de nombreux avantages, mais ils présentent également certaines limitations.

- **Avantages :**

- Capacité à générer des images nouvelles et très réalistes.
- Utiles pour l'augmentation de données et la simulation de scénarios variés.

- **Limitations :**

- Difficiles à entraîner de manière stable.
- Peuvent souffrir de modes collapse où le générateur commence à produire un nombre limité de sorties.

Applications des GANs

Les GANs ont révolutionné plusieurs domaines de la vision par ordinateur grâce à leur capacité à générer des données réalistes.

- **Applications :**

- Création d'art et de designs nouveaux.
- Synthèse d'images pour l'entraînement de modèles de vision.
- Amélioration de la résolution des images (super-resolution).
- Simulation pour la formation en réalité augmentée.

Traitement automatique du langage (NLP)

Tâches classiques en NLP

On retrouve dans le NLP un certain nombre de tâches classiques, qui servent à la fois dans les applications pratiques que dans l'évaluation de l'état de l'art.

- Modélisation du langage (*language modeling*)
- Classification de texte
- Extraction d'informations
- Détection de thématiques (*topic modeling*)
- Résumé de texte
- Question/réponse
- Traduction automatique
- Agent conversationnel (domaine ouvert ou fermé)

Pourquoi le NLP est si difficile ?

Il y a plusieurs facteurs qui rendent le langage naturel difficile à traiter par la machine :

- Le langage naturel est un problème en grande dimension.
- Le langage naturel est **ambigu**.
 - “Acheter un avocat”
- Le langage naturel repose sur le **sens commun**.
 - Il fait froid *en hiver*
 - Le beurre fond dans le four
- Le langage naturel n'est pas basé sur des **règles**.
 - *En bleu adorable fleurit
Le toit de métal du clocher. Alentour
Plane un cri d'hirondelles, autour
S'étend le bleu le plus touchant. Le soleil*
Friedrich Hölderlin

Pré-traitement des données textuelles

Les données textuelles sont non structurées. Les principales étapes de pré-traitement impliquent :

- Segmentation des phrases et **tokenisation**.
- Suppression de **stop words**, de la ponctuation, etc.
- **Vectorisation** : Transformation du texte en vecteurs numériques.
- **Stemming, lemmatisation**, conversion des majuscules, etc.
- **Normalisation** : traitement des accents, des caractères spéciaux, etc.
- **Suppression des entités nommées** : Filtrage des noms de lieux, personnes, organisations, etc.
- Détection du langage, **POS tagging** (part-of-speech), etc.
- Correction orthographique : Pour améliorer la qualité des textes bruts.

Tokenisation

La tokenisation consiste à segmenter une phrase en unités plus petites appelées **tokens**, qui peuvent être des mots, des caractères ou des sous-parties de mots.

Types de tokenisation :

- **Tokenisation par mots** : Chaque mot est un token.
 - *Exemple* : "Il fait beau à Paris." → ["Il", "fait", "beau", "à", "Paris"]
- **Tokenisation par caractères** : Chaque caractère est un token.
 - *Exemple* : "Paris" → ["P", "a", "r", "i", "s"]
- **Tokenisation par sous-mots (subword)** : Découpe les mots en sous-unités fréquentes, utile pour gérer les langues morphologiquement complexes ou les néologismes.
 - *Exemple* : "démocratisation" → ["démocrat", "isation"]

La tokenisation est une étape essentielle du pré-traitement dans le NLP. Elle permet de transformer du texte brut en unités traitables par les algorithmes de machine learning.

Stemming and lemmatisation

- Le **stemming** est l'opération qui consiste à supprimer les suffixes et réduire les mots à leur forme de base.
 - marcher → march
 - marchait → march
 - marcha → march
 - marchassiez → march
 - marché → march
- La **lemmatisation** est l'opération qui consiste à réduire un mot à sa racine.
 - nous → nous
 - sommes → être
 - venus → venir
 - faire → faire
 - les → le
 - marchés → marché

Vectorisation

- **One-Hot encoding:** Étant donné un vocabulaire V , chaque mot est représenté par un vecteur binaire (à 0 ou 1) de dimension V .
 - $V = \{Tom, mange, pomme, chien, ciel\}$, où $\dim(V) = 5$
 - Tom = [1, 0, 0, 0, 0]
 - mange = [0, 1, 0, 0, 0]
 - pomme = [0, 0, 1, 0, 0]
 - etc.
- **N-grams**
 - "machine learning est très utilisé en NLP."
 - 1-gram: {machine, learning, est, très, utilisé, en, NLP}
 - 2-gram: {machine learning, learning est, est utilisé, utilisé en, en NLP}

Reconnaissance des entités nommées (NER)

La reconnaissance d'entités nommées (**NER**) est une tâche de NLP qui consiste à identifier et classifier des entités dans un texte, comme des personnes, des organisations, des lieux, des dates, etc.

Exemples d'entités nommées :

- **Personne** : "Albert Einstein"
- **Organisation** : "Université de Princeton"
- **Lieu** : "Paris"
- **Date** : "15 octobre 2024"

Catégories courantes d'entités nommées :

- **Personnes** : Noms de personnes célèbres ou non.
- **Lieux** : Pays, villes, régions.
- **Organisations** : Entreprises, institutions, ONG.
- **Dates et heures** : Dates précises, années, heures.
- **Monnaie et quantités** : Montants financiers, unités de mesure.

La NER est cruciale pour de nombreuses applications NLP, telles que l'extraction d'informations, la recherche d'informations, et la classification de documents.

Part-of-Speech (POS) Tagging

Le **Part-of-Speech (POS) Tagging** est une tâche de NLP qui consiste à assigner à chaque mot d'une phrase son rôle grammatical (catégorie grammaticale).

Exemples de catégories grammaticales (POS tags) :

- **Nom (N)** : Objet, personne, lieu, idée. *Ex. : chat, Paris*
- **Verbe (V)** : Action, état. *Ex. : mange, est*
- **Adjectif (Adj)** : Caractéristique, qualité. *Ex. : grand, heureux*
- **Adverbe (Adv)** : Modifie un verbe ou une phrase. *Ex. : rapidement, bien*
- **Déterminant (Det)** : Définit un nom. *Ex. : le, une*
- **Préposition (Prep)** : Indique une relation. *Ex. : dans, avec*

Exemple de POS Tagging :

- "Le chat mange une souris."
- **Résultat** : Le/Det chat/N mange/V une/Det souris/N

Le POS Tagging est une étape clé dans de nombreuses applications de NLP, notamment pour l'analyse syntaxique, la traduction automatique et l'extraction d'information.

Techniques statistiques pour le NLP

Introduction aux techniques statistiques pour le NLP

Les techniques statistiques en NLP reposent sur des modèles probabilistes pour analyser et traiter les textes. Elles permettent d'extraire des informations pertinentes en se basant sur les fréquences de mots et les relations entre eux.

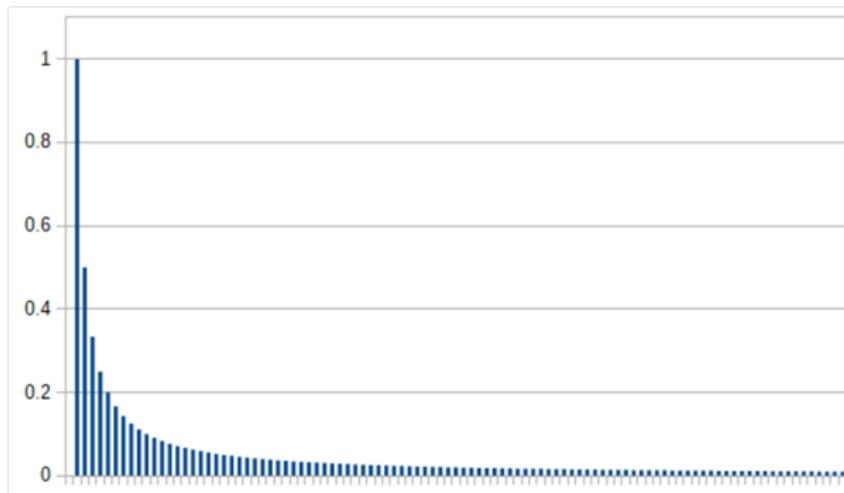
Principes de base :

- Les modèles probabilistes estiment les probabilités des séquences de mots et des événements linguistiques.
- Les algorithmes apprennent à partir des données d'entraînement, sans règles explicites définies à l'avance.

Ces techniques ont permis de remplacer les approches basées sur des règles linguistiques par des modèles plus flexibles et adaptatifs.

Principe des techniques statistiques pour le NLP

- Un document est caractérisé par la distribution des mots qui le composent.
- La distribution des mots obéit généralement à une loi de puissance (ou de Zipf). La majorité du langage que nous utilisons ordinairement est composé d'un nombre très limité de mots.



Introduction aux modèles n-grams

Les modèles **n-grams** sont une des premières techniques statistiques utilisées pour modéliser les séquences de mots dans le langage naturel.

Principe :

- Un modèle n-gram prédit la probabilité d'un mot en fonction des $n - 1$ mots précédents.
- n -gram peut être : un unigram (1 mot), bigram (2 mots), trigram (3 mots), etc.

Exemple (2-gram - bigram) :

- Phrase : "Je mange une pomme."
- Bigrammes : {("Je", "mange"), ("mange", "une"), ("une", "pomme")}

Calcul de la probabilité avec un modèle n-grams

Les modèles n-grams estiment la **probabilité** d'une phrase en se basant sur les probabilités conditionnelles des mots.

Probabilité d'une phrase :

$$P(w_1, w_2, w_3, \dots, w_n) \approx P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2) \cdot \dots \cdot P(w_n|w_{n-1})$$

Exemple (2-gram) :

- Phrase : "Je mange une pomme."
- Probabilité de la phrase :

$$P("Je") \cdot P("mange"|"Je") \cdot P("une"|"mange") \cdot P("pomme"|"une")$$

Applications et limites des modèles n-grams

Applications des n-grams :

- **Modélisation du langage** : Utilisés pour estimer la probabilité des séquences de mots dans des applications comme la reconnaissance vocale ou la correction orthographique.
- **Traduction automatique** : Les n-grams permettent de prédire les séquences de mots dans les langues cibles.
- **Génération de texte** : Génération automatique de phrases probables à partir de contextes.

Limites des n-grams :

- **Explosion combinatoire** : La taille du modèle croît exponentiellement avec n , rendant le stockage des probabilités complexe pour de grandes valeurs de n .
- **Manque de contexte à long terme** : Les n-grams ne capturent que les relations entre les n mots précédents, ignorant les dépendances à longue distance.

Introduction à TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) est une méthode utilisée pour évaluer l'importance d'un mot dans un document, relativement à l'ensemble d'un corpus.

Objectif :

- Mesurer l'importance d'un mot dans un document tout en réduisant l'importance des mots trop fréquents dans le corpus.

Applications :

- Extraction de caractéristiques textuelles pour des tâches de classification.
- Recherche d'information et moteurs de recherche (identifier les mots-clés).
- Résumé automatique de textes.

Term Frequency (TF)

La **fréquence d'un terme (TF)** mesure combien de fois un mot apparaît dans un document. Plus un mot apparaît souvent dans le document, plus sa fréquence est élevée.

Formule :

$$TF(t, d) = \frac{\text{Nombre d'occurrences du terme } t \text{ dans le document } d}{\text{Nombre total de mots dans le document } d}$$

Exemple :

- Document : "Le chat mange une souris. Le chat dort."
- Nombre d'occurrences du mot "chat" = 2
- Nombre total de mots dans le document = 7
- Donc :

$$TF(\text{"chat"}, d) = \frac{2}{7} = 0.285$$

Inverse Document Frequency (IDF)

L'**Inverse Document Frequency (IDF)** mesure l'importance d'un mot dans le corpus. Les mots qui apparaissent dans de nombreux documents ont une faible importance, car ils sont trop communs.

Formule :

$$IDF(t) = \log \left(\frac{\text{Nombre total de documents dans le corpus}}{\text{Nombre de documents contenant le terme } t} \right)$$

Exemple :

- Supposons que le mot "chat" apparaisse dans 100 documents d'un corpus de 1000 documents.
- Le nombre total de documents dans le corpus = 1000
- Le nombre de documents contenant le mot "chat" = 100
- Donc :

$$IDF("chat") = \log \left(\frac{1000}{100} \right) = \log(10) = 1$$

Calcul de TF-IDF

La valeur **TF-IDF** combine la fréquence d'un terme dans un document (TF) et l'inverse de sa fréquence dans le corpus (IDF).

Formule :

$$TF-IDF(t, d) = TF(t, d) \times IDF(t)$$

Exemple :

- Document : "Le chat mange une souris."
- $TF("chat", d) = 0.285$, $IDF("chat") = 1$
- $TF-IDF("chat", d) = 0.285 \times 1 = 0.285$

Exemple de matrice TF-IDF

Documents :

- Document 1 (D1) : "le chat dort."
- Document 2 (D2) : "le chien dort."
- Document 3 (D3) : "le chat mange."
- Document 4 (D4) : "le chien mange."

Matrice TF-IDF :

Matrice TF-IDF

	D1	D2	D3	D4
chat	0.231	0.0	0.231	0.0
chien	0.0	0.231	0.0	0.231
dort	0.231	0.231	0.0	0.0
mange	0.0	0.0	0.231	0.231

Merci de votre attention

redha.moulla@axia-conseil.com