



# Groupe Heppner

## Introduction au Machine Learning

*Présenté par Victor Chemla*

# Déroulement de votre formation

## ■ Horaires

9h / 12h30 ○ Pause déjeuner 1h30 ○ 14h / 17h30

Pause : matin (≈11h) + après-midi (≈15h30)

## ■ Présentation

Votre formateur

Tour de table : votre profil, vos attentes, vos besoins

## ■ Les objectifs de votre formation

## ■ Détail du programme

## ■ Plateforme formation PLB

Signatures matin + après midi

Avant de commencer : auto - évaluez vos connaissances

Evaluation en fin de formation



## VOS CONTACTS CHEZ PLB

**Equipe IT** [assistance@plb.fr](mailto:assistance@plb.fr)

01 43 34 34 10

pour toute question technique  
(VM, Teams, login, etc.)

**Référent stagiaire** [rs@plb.fr](mailto:rs@plb.fr)

pour tout autre sujet ... et vous  
accompagner durant votre formation

# Votre formateur

- Victor Chemla, Data Scientist chez Mazars.



# Tour de table

Votre parcours / votre métier  
Votre expérience dans le domaine  
Vos attentes de la formation

# Programme de la formation

## Introduction

Les différents types de données  
Rebaser, normaliser et standardiser

## Qu'est-ce que le *machine learning* ?

Algorithmie  
IA / ML / DL ?  
Qu'est-ce que le *machine learning* ?  
Régression, classification et clusterisation

## Régression linéaire

## Régression logistique

## Réseaux de neurons

## k-Nearest Neighbor

## k-means

## Méthodes ensemblistes

## Support Vector Machine

## Algorithmes de règles d'association

## Evaluation de la performance d'un algorithme

Evaluation de la performance d'un algorithme de régression  
Evaluation de la performance d'un algorithme de classification

*Train, Validation, Test*

*Cross-validation*

*Underfitting et overfitting*

## Formation par la pratique en python

Environnements de développement  
Formation par la pratique en python

# Signatures, documents, évaluations sur le LMS

- Vous avez reçu le lien pour créer votre identifiant et accéder à votre espace en ligne

*Les signatures sont obligatoires pour le suivi de votre session de formation*

**Vous pouvez signer chaque matin + après midi**

plb consultant

Feuille de présence Évaluation Documents Formation à distance Machines de TP

**Feuille de présence**  
Merci de signer les feuilles d'émargement pour la session suivante

Robot Framework (Session : 220866A)

Du 23/05/2022 au 24/05/2022 14 heures sur 2 jours (09h15 - 17h30) Classe à distance PLB PLACE DE LA FORMATION

Par ma signature, j'atteste avoir reçu la formation ci-dessus référencée

**Lundi 23/05/2022**

Matin Après-midi

**Mardi 24/05/2022**

Matin Après-midi

**Vous pourrez remplir votre évaluation le dernier jour**

plb consultant

Feuille de présence **Évaluation** Documents Formation à distance Machines de TP

**Évaluation**  
Merci de remplir votre évaluation à la fin de chaque formation

Robot Framework (Session : 220866A)

Du 23/05/2022 au 24/05/2022 14 heures sur 2 jours (09h15 - 17h30) Classe à distance PLB

**Évaluation**  
Vous n'avez pas encore rempli votre évaluation

Remplir mon évaluation →



# Auto-évaluez vos connaissances avant / après la formation sur le LMS

- Objectif de ces auto-évaluation individuelles
  - Permettre à votre formateur de mieux connaître votre niveau de connaissance avant de commencer
  - Valider la qualité de votre progression en fin de formation

plb consultant

Du 17/08/2022 au 19/08/2022 21 heures sur 3 jours (09h15 - 17h30) Classe à distance PLB CGI FRANCE

Feuille de présence Évaluation **Auto-évaluation** Documents Formation à distance Machines de TP

**Auto-évaluation**

Cliquez sur la note que vous pensez être correcte, de 0 à 9.  
0 équivaut à aucune compétence, 9 équivaut à une maîtrise totale de la compétence.

L'auto-évaluation, qu'est-ce que c'est ?

1. Je renseigne mes compétences avant la formation

2. Je me forme pour améliorer mes compétences

3. Je renseigne mes compétences après la formation

Ⓢ Décrire les différentes technologies Java (Java EE, JVM, Java embarqué, etc.)

Mes compétences avant la formation 0 1 2 3 4 5 6 7 8 9

Mes compétences après la formation 0 1 2 3 4 5 6 7 8 9

Ⓢ Utiliser Eclipse et comprendre les apports d'un IDE (Integrated Development Environment) en général

Mes compétences avant la formation 0 1 2 3 4 5 6 7 8 9

Mes compétences après la formation 0 1 2 3 4 5 6 7 8 9

Pour chaque objectif pédagogique de votre formation, **choisissez votre niveau entre 0 et 9 avant la formation**

**Vous pourrez remplir votre auto-évaluation à la fin de la formation**



# Introduction



# Les différents types de données (1/4)

## Les données tabulaires :

Ce sont des tableaux où toutes les lignes sont appelées des enregistrements et toutes les colonnes des champs. Chaque champ permet de stocker un type spécifique de variable (texte, nombre, date, etc.).

Ce format est particulièrement présent dans les bases de données relationnelles.

	age (n)	job (c)	marital (c)	education (c)	balance (n)	housing (c)
0	30	unemployed	married	primary	1787	no
1	33	services	married	secondary	4789	yes
2	35	management	single	tertiary	1350	yes
3	30	management	married	tertiary	1476	yes
4	59	blue-collar	married	secondary	0	yes
5	35	management	single	tertiary	747	no

Figure: Exemple de données tabulaires..  
(Source : UCI Machine Learning Repository)

# Les différents types de données (2/4)

## Les images :

Elles sont représentées sous la forme de matrices dont les éléments sont des entiers appelés pixels.  
Plus le nombre de pixels dans une image est élevé, plus sa résolution est grande.

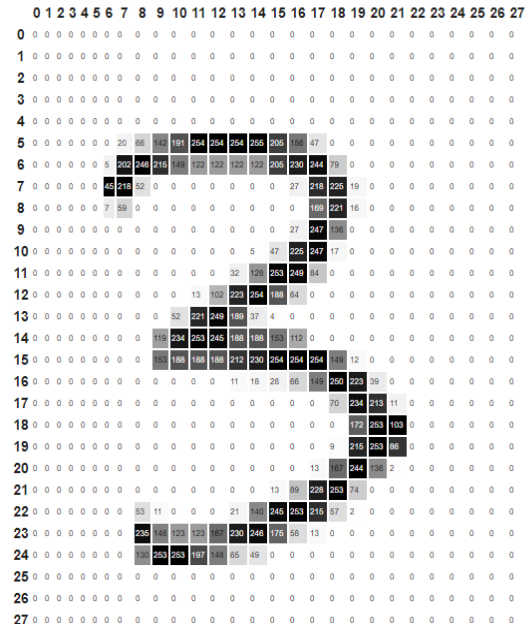


Figure: Exemple d'une image matricielle en niveaux de gris (valeurs des pixels entre 0 et 255 [codage sous 8 bits]).

(Source : Base de données MNIST)

# Les différents types de données (3/4)

## Les textes :

Les données textuelles sont représentées sous la forme de chaînes de caractères. Un code où chaque chiffre représente un caractère peut être utilisé pour convertir un texte en binaire (ASCII ou Unicode) de façon à pouvoir être traité par ordinateur.

Converting the text "hope" into binary

Characters:	h	o	p	e
ASCII Values:	104	111	112	101
Binary Values:	01101000	01101111	01110000	01100101
Bits:	8	8	8	8

Figure: Conversion d'un mot en code binaire.  
(Source : [www.computerhope.com](http://www.computerhope.com))

# Les différents types de données (4/4)

## Les series temporelles :

Ce sont des séries de données indexées par le temps, on parle également de séries chronologiques. Elles représentent l'évolution d'une quantité spécifique ou d'un phénomène physique (à l'aide de capteurs) au cours du temps.

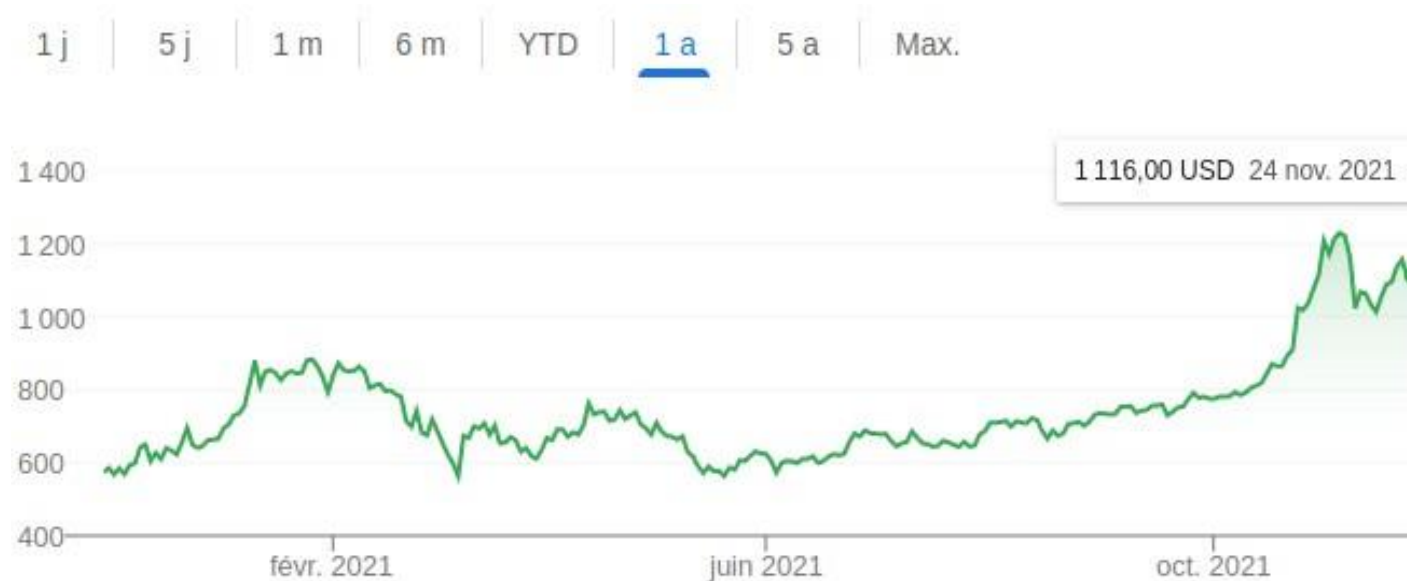


Figure: Exemple d'une série temporelle.  
(Cours de l'action Tesla)

# Rebaser, normaliser et standardiser (1/2)

## Rebaser :

Rebaser un vecteur signifie changer les unités de mesure des données.  
Par exemple, pour convertir une température de Celsius en Fahrenheit.

## Normaliser des données :

La normalisation des données est un processus couramment utilisé en statistiques, en science des données et en apprentissage automatique pour mettre à l'échelle les valeurs de différentes variables dans un même intervalle. L'objectif principal de la normalisation est de rendre les données comparables entre elles et plus facilement interprétables par les algorithmes d'analyse et de modélisation.

Dans de nombreux cas, les données peuvent avoir des échelles très différentes, c'est-à-dire que certaines variables peuvent avoir des valeurs beaucoup plus grandes ou plus petites que d'autres. Cela peut poser des problèmes pour certaines techniques statistiques ou algorithmes d'apprentissage automatique, car ils peuvent être sensibles à l'échelle des données. La normalisation permet de résoudre ce problème en ajustant les valeurs des variables pour qu'elles se situent dans un intervalle spécifié, souvent entre 0 et 1, ou autour de la moyenne avec un écart-type donné.

# Rebaser, normaliser et standardiser (2/2)

## Standardiser des données :

La standardisation transforme les valeurs d'une variable pour qu'elles aient une moyenne de 0 et un écart-type de 1.

Contrairement à la normalisation, la standardisation ne fixe pas de plage spécifique pour les valeurs transformées. La standardisation est utile lorsque les variables ont des échelles très différentes, et elle permet de centrer les données autour de zéro et de les mettre à l'échelle par rapport à l'écart-type, ce qui peut faciliter l'interprétation des coefficients dans certains modèles.

## Normaliser vs. standardiser des données :

La **normalisation** des données (*data normalization*) et la **standardisation** des données (*data standardization*) répondent au même enjeu de représentativité de la donnée mais dans des perspectives différentes. Bien qu'elles soient toutes les deux des techniques de mise à l'échelle des données, elles diffèrent dans la manière dont elles transforment les valeurs des variables.

**En fonction de la nature de vos données et des enseignements que vous souhaitez en tirer, il faudra tantôt recourir à la *data normalization*, tantôt à la *data standardization*.**





# Qu'est-ce que le *Machine Learning* ?

# Algorithmie

## Déterministe :

- IA symbolique

## Non déterministe :

- Machine learning (algorithme à base d'apprentissage)
  - Si les **données changent** >>> la **modélisation bouge** (apprentissage statistique, modélisation dynamique)
  - Si l'**initialisation change** >>> la **modélisation bouge**
  - Si le **paramétrage de l'apprentissage change** (hyperparamètres) >>> la **modélisation bouge**

*Remarque :  $y = x^2$  |  $y \approx x^2$*

# IA / ML / DL ? (1/3)

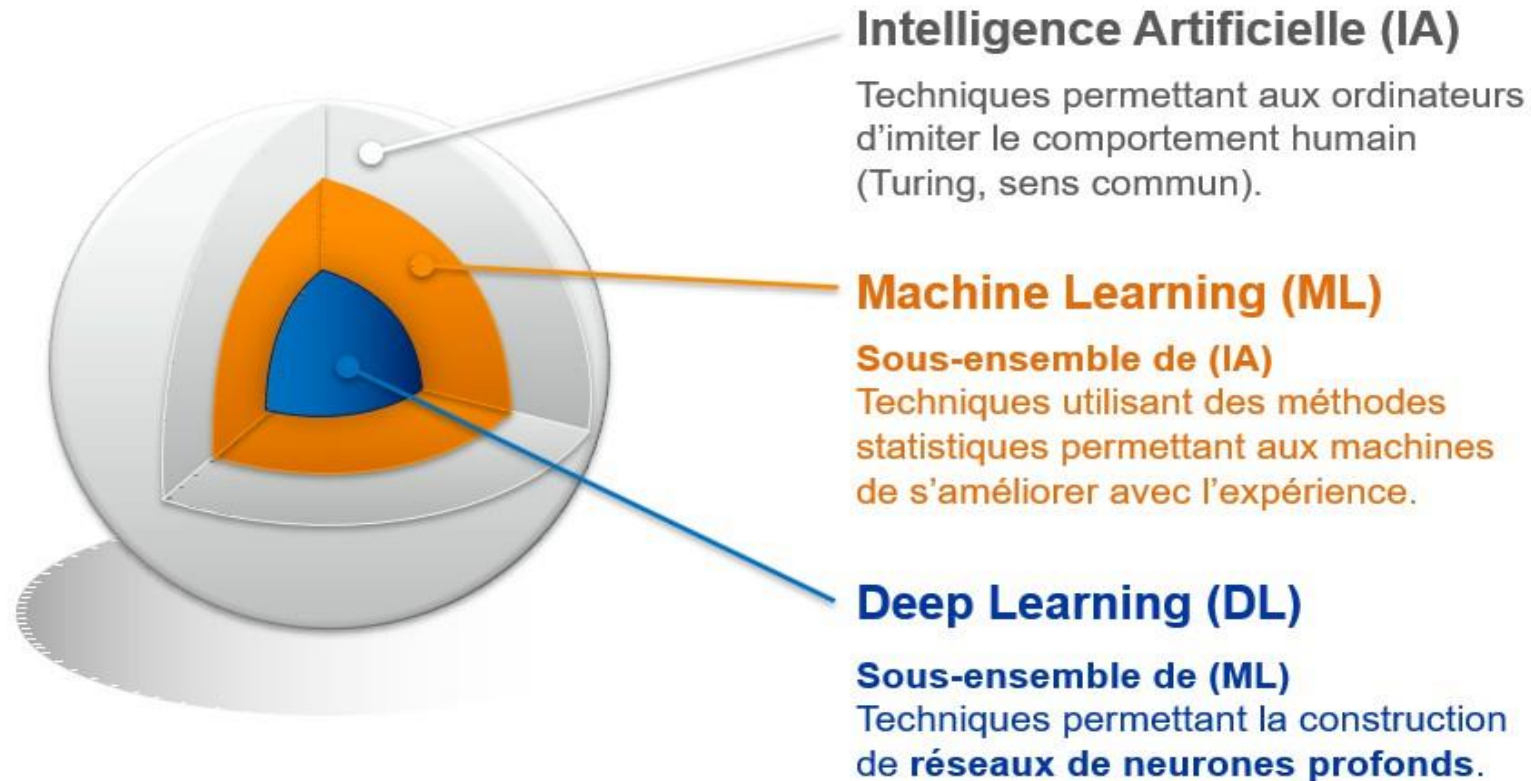


Figure: L'IA et ses sous-ensembles.

# IA / ML / DL ? (2/3)

## IA symbolique :

- L'algorithme à base de règle. Il choisit chaque coup en fonction d'un ensemble de règles prédéfinies.
- L'ordinateur n'apprend pas. Il se contente d'exécuter les ordres qu'on lui donne.

## Machine learning :

- Algorithme à base d'apprentissage.
- L'ordinateur calcule lui-même les meilleures décisions à prendre. Il exécute les coups jugés optimaux.
- Pour atteindre un tel niveau l'ordinateur traverse d'abord une phase d'apprentissage à partir d'exemples.
- On parle d'apprentissage machine. Les algorithmes apprennent à partir d'exemples, à exécuter des tâches pour lesquelles ils n'ont pas explicitement été programmés.

**Apprentissage statistique, inférence, science de l'ingénieur (expérimentations numériques)**

# IA / ML / DL ? (3/3)

## Reinforcement learning (apprentissage par renforcement) :

- On plonge un agent intelligent dans un environnement avec lequel il interagit (lorsque l'on dresse un chien, il est récompensé par un bon comportement et puni pour un mauvais).
- C'est l'environnement qui fournit les données d'entraînement à l'algorithme, en réagissant aux actions de ce dernier.
- Cet apprentissage met souvent en jeu des réseaux de neurones (AlphaGo).

## Generative AI :

- L'intelligence artificielle générative ou IA générative (IAg ou GenAI) est un type de système d'intelligence artificielle (IA) capable de générer du texte, des images ou d'autres médias en réponse à des invites (ou prompts en anglais). Elle est dite multimodale quand elle est construite à partir de plusieurs modèles génératifs, ou d'un modèle entraîné sur plusieurs types de données et qu'elle peut produire plusieurs types de données.
- Par exemple, la version GPT-4 d'OpenAI accepte les entrées sous forme de texte et/ou d'image.

# Qu'est-ce que le *machine learning* ?

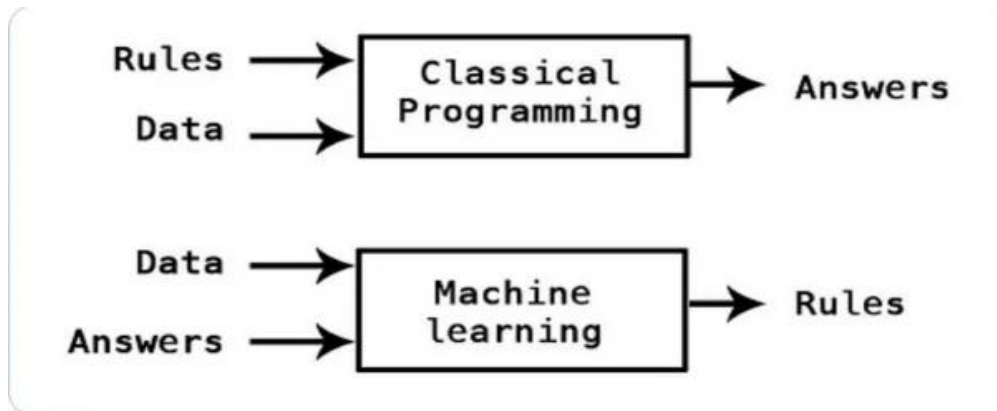
## Définition de Machine Learning :

1998, (American computer scientist) Tom Mitchell:

***“A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .”***

*Example: playing checkers.*

- $E$  = the experience of playing many games of checkers.
- $T$  = the task of playing checkers.
- $P$  = the probability that the program will win the next game.





# Régression, classification et clusterisation (1/2)

## Regression :

Dans une tâche de régression, on cherche à construire un modèle capable de prédire une variable continue quantitative.

Pour cela, plusieurs méthodes d'analyse statistique peuvent être utilisées, permettant d'approcher une variable à partir d'autres qui lui sont corrélées.

→ *Exemple : prédire le prix d'une maison à partir de son emplacement, nombre de pièces, proximité des commerces et des transports, etc.*

## Classification :

La classification et la régression sont deux sous-ensembles de l'apprentissage supervisé.

Dans un exercice de classification, on cherche à construire un modèle capable de prédire une variable discrète qualitative (classe ou label). L'objectif est de pouvoir assigner à chaque observation d'un jeu de données une catégorie spécifique.

→ *Exemple : classer un email dans la catégorie spam ou non.*

# Régression, classification et clusterisation (2/2)

## Clustering :

Le clustering est une discipline particulière du *machine learning* ayant pour objectif de séparer vos données en groupes homogènes ayant des caractéristiques communes.

C'est un domaine très apprécié en marketing, par exemple, où l'on cherche souvent à segmenter les bases clients pour détecter des comportements particuliers.

→ *Exemple : segmentation client, analyse de données, segmentation d'une image, etc.*



# Régression linéaire

# Régression linéaire (1/5)

- **Définition** : **algorithme supervisé** dont le modèle cherche à trouver une **relation linéaire entre une variable dépendante et une ou plusieurs variables indépendantes (ou explicatives)**.
- **Naissance** : le terme originel de régression provient des expériences statistiques de l'anthropologue Francis Galton qui, dans un article de 1886, constate un phénomène de « régression vers la moyenne » de la taille des fils en fonction de la taille des pères.
- **Applications** : Toute application où le modèle linéaire est pertinent, les erreurs sur les variables indépendantes sont décorrélées et issues d'une distribution Gaussienne à variance constante.

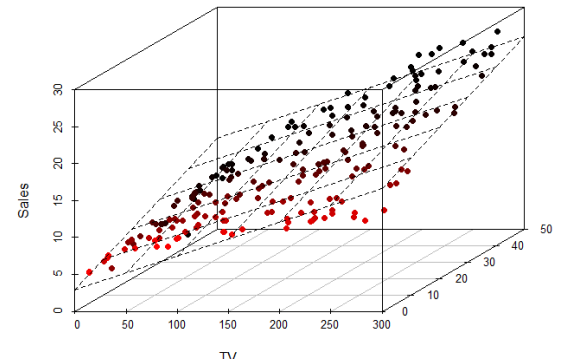
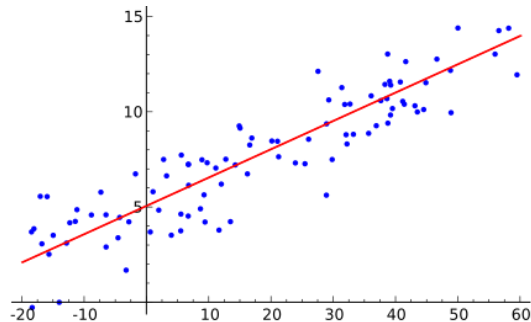
# Régression linéaire (2/5)

## Données :

La Régression (multi-)linéaire est le plus simple algorithme de Machine Learning supervisé. Pour la suite, nous introduisons la notation suivante :

$$\{x_i, y_i\}_{i=1, \dots, n}, \quad x_i \in \mathbb{R}^d$$

Pour chaque observation  $i$ , nous appelons  $x_i$  **explanatory** ou **independent variables** et  $y_i$  **dependent variable**. Le but de la régression (multi-)linéaire est d'apprendre un modèle qui relie  $x_i$  à  $y_i$  à travers **une combinaison linéaire**.



# Régression linéaire (3/5)

## Optimisation des paramètres du modèle :

**Solution numérique** : tous les algorithmes dédiés à la minimisation d'une fonction numérique. Les méthodes basées sur la descente du gradient sont les plus connues.

**Petit rappel** : toute fonction différentiable  $f(x)$  peut être minimisée à travers une descente de gradient.

**Descente de gradient** : méthode qui consiste à ajuster les curseurs du modèle par petits pas successifs. Étant donné un point de départ  $x_0$ , nous pouvons suivre la direction de pente maximale jusqu'à arriver à un point où le gradient est proche de zéro (condition d'arrêt).

Dans le voisinage d'un point  $\tilde{x}$ , la direction ci-dessus est donnée par la direction du gradient calculé sur  $\tilde{x}$ . La méthode est itérative:

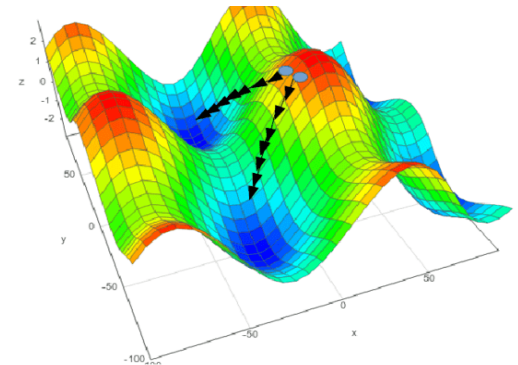
$$\tilde{x} \leftarrow x_0$$

**while not condition:**

$$\tilde{x} \leftarrow \tilde{x} - \alpha * \nabla f(\tilde{x})$$

**solution:**

$$\tilde{x}$$





# Régression linéaire (4/5)

## Régression linéaire appliquée aux series temporelles :

L'équivalent de la régression linéaire pour les séries temporelles s'appelle le modèle **ARIMA**.

De manière générale il s'agit de comprendre l'auto-corrélation de la quantité d'intérêt, c'est-à-dire combien sa valeur à un instant donné influence sa valeur à des instants ultérieurs.

# Régression linéaire (5/5)

## Points clés :

- On peut apprendre les coefficients d'un modèle de régression paramétrique par maximisation de vraisemblance, ce qui équivaut à minimiser le risque empirique en utilisant le coût quadratique comme fonction de perte, et revient à la méthode des moindres carrés.
- On utilise un algorithme à directions de descente pour apprendre le modèle.



# Régression logistique

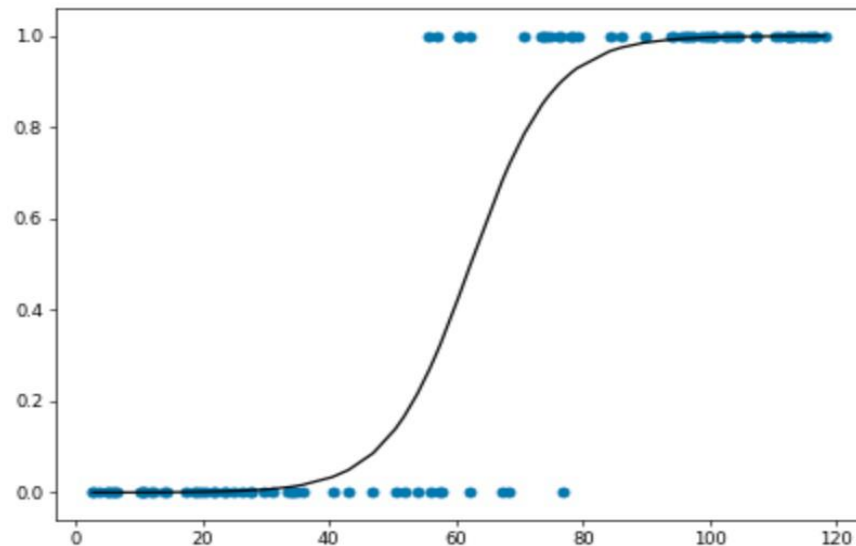
# Régression logistique (1/2)

- **Définition** : algorithme supervisé de classification binaire ou multi-classes.
- **Naissance** : le terme logistique dérive de la fonction mathématique que la méthode utilise : la fonction logistique. Cette fonction a été introduite pour la première fois par le mathématicien Pierre François Verhulst pendant les années 1830 et 1840 pour ses études sur la croissance des populations.  
L'utilisation moderne pour des tâches de classification trouve ses racines dans les travaux de Chester Ittner Bliss, John Gaddum et Ronald A. Fisher pendant les années 1930.
- **Applications** : Santé, score de crédit, caractérisation des images, reconnaissance d'écriture manuscrite.

# Régression logistique (2/2)

La **régression logistique** constitue un **cas particulier de modèle linéaire généralisé**. C'est un modèle de régression binomiale. Elle est largement utilisée en apprentissage.

Comme pour tous les modèles de régression binomiale, il s'agit d'expliquer au mieux une variable binaire (la présence ou l'absence d'une caractéristique donnée) par des observations réelles nombreuses, grâce à un modèle mathématique (en d'autres termes, d'associer une variable aléatoire de Bernoulli à un vecteur de variables aléatoires).





# Réseaux de neurones



# Réseaux de neurones (1/11)

## Quelques éléments historiques de l'apprentissage profond :

L'apprentissage profond possède une longue et riche histoire, a porté de nombreux noms et a connu des hauts et des bas en termes de popularité :

- Années 1940–1960 : on parle plutôt de cybernétique, avec le développement de théories d'apprentissage biologique et l'implémentation des premiers modèles comme le perceptron (Rosenblatt, 1957).
- Années 1980–1990 : le terme en vigueur est le connexionnisme, avec la mise en place de la back-propagation (Rumelhart et al., 1986), permettant l'entraînement d'un réseau de neurones avec une ou deux couches cachées.
- Autour de 2006 : l'expression *Deep Learning* fait son apparition.

# Réseaux de neurones (2/11)

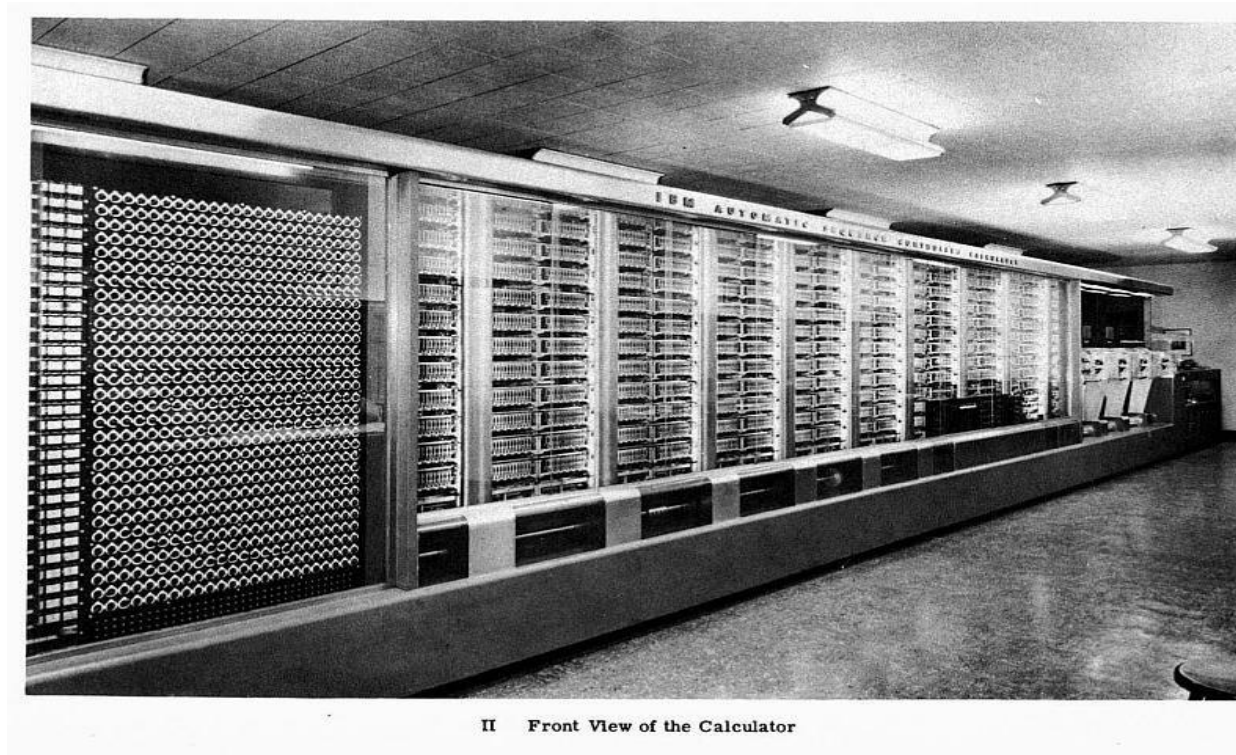


Figure: Premier perceptron, le Mark I. Université Cornell, Frank Rosenblatt, 1957.  
(Source : [www.tomshardware.fr](http://www.tomshardware.fr))

# Réseaux de neurones (3/11)

## Quelques tendances principales de l'apprentissage profond :

- L'apprentissage profond est devenu plus utile à mesure que la quantité des données d'entraînement disponibles a augmenté.
- La taille des modèles d'apprentissage profond s'est accrue au fil du temps, à mesure que l'infrastructure informatique (à la fois matérielle et logicielle) dédiée à l'apprentissage profond s'est améliorée.
- L'apprentissage profond a permis de résoudre des applications de plus en plus complexes avec une précision croissante au fil du temps.
- L'apprentissage profond permet à l'ordinateur de construire des concepts complexes à partir de concepts plus simples.

# Réseaux de neurones (4/11)

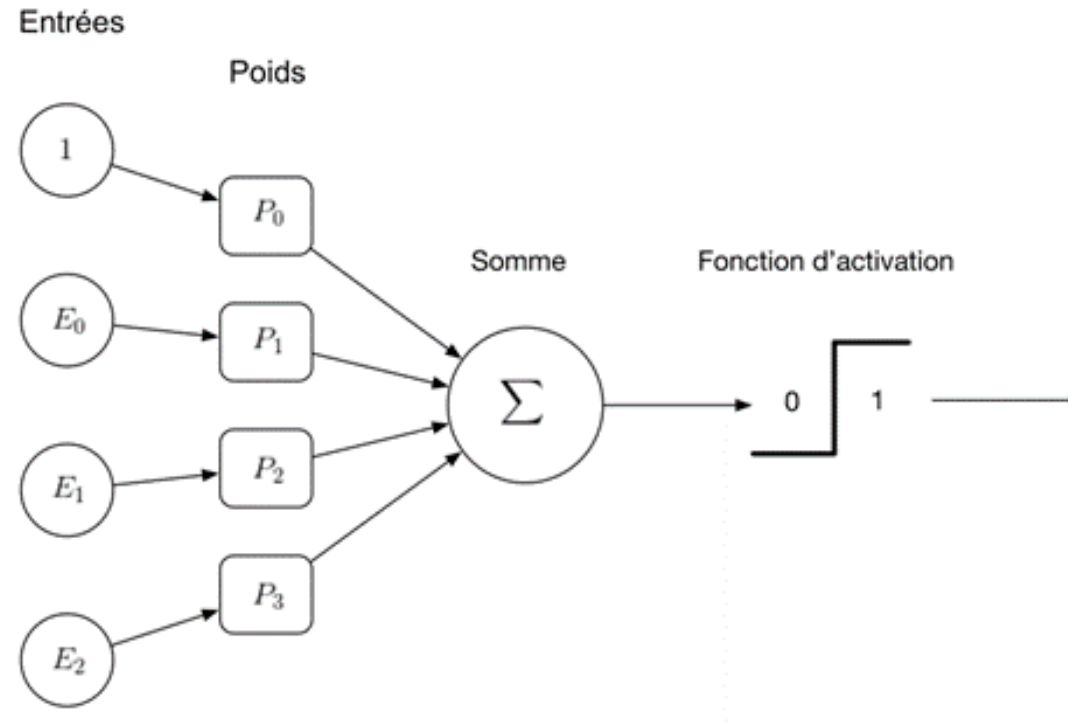


Figure: Schéma d'un perceptron.

*Le neurone effectue une somme pondérée des entrées (avec un biais) avant l'application de la fonction échelon de Heaviside.*

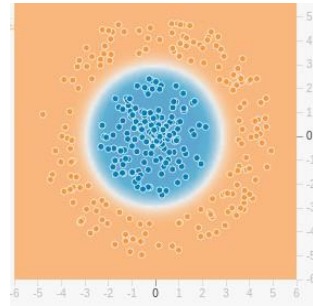
# Réseaux de neurones (5/11)

## Comment les réseaux de neurones apprennent ?

- Un réseau de neurones peut être vu comme un arrangement intelligent de modules linéaires et non-linéaires connectés entre eux.
- Le mécanisme d'apprentissage s'appelle la technique de **back-propagation**. Il s'agit d'une méthode qui met à jour les paramètres du réseau dans la direction (gradient) qui minimise une mesure d'erreur prédéfinie que l'on appelle **fonction de coût**.
- On parle d'apprentissage supervisé car il nécessite des données qui auront été **labellisées** en amont.

# Réseaux de neurones (6/11)

## Exemple : classification binaire



Features d'entrée :

$$x = \{x_1, x_2\}$$

- Et après ? classificateur basé sur les *Neural Networks* [ici](#).

Données d'entrée :

$$x_i$$

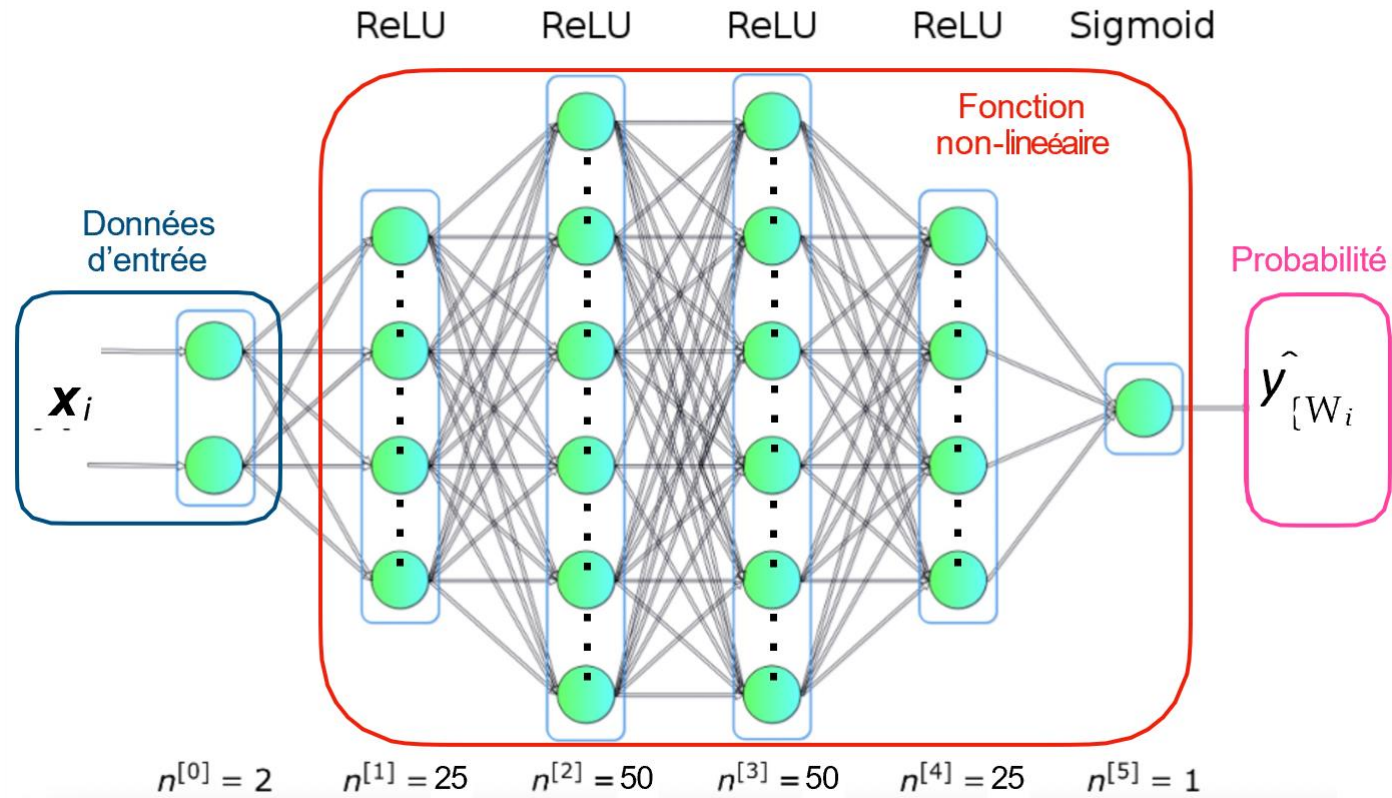
Fonction non-linéaire :

$$f(x_i)$$

Sortie :

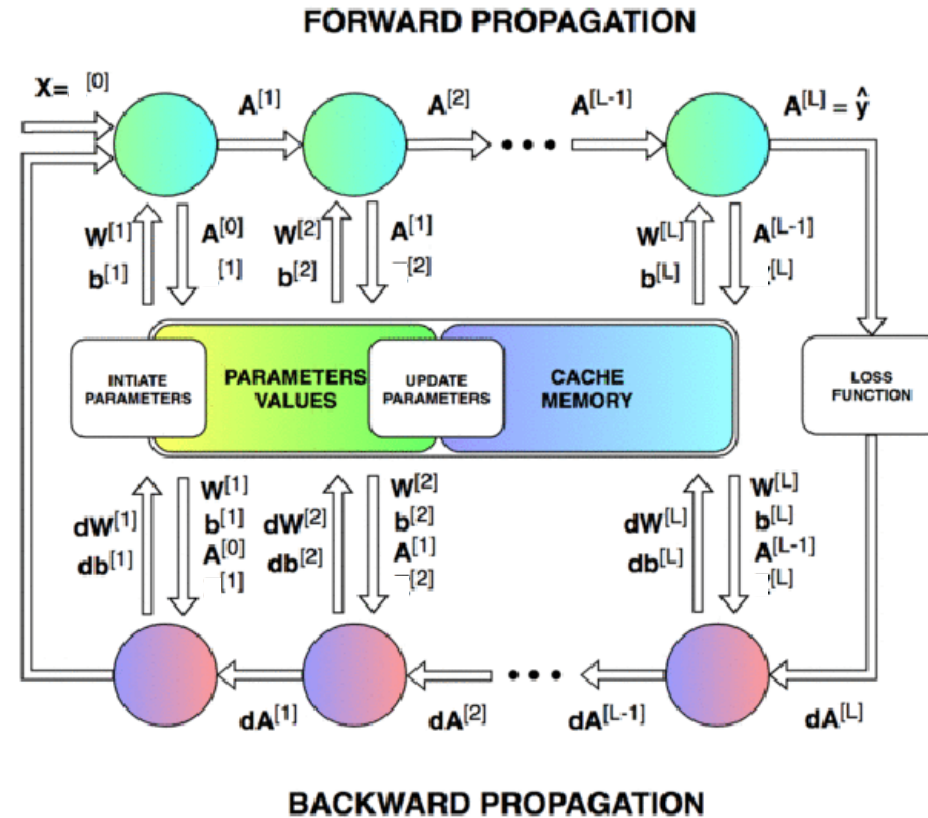
$$\hat{y}_i$$

# Réseaux de neurones (7/11)





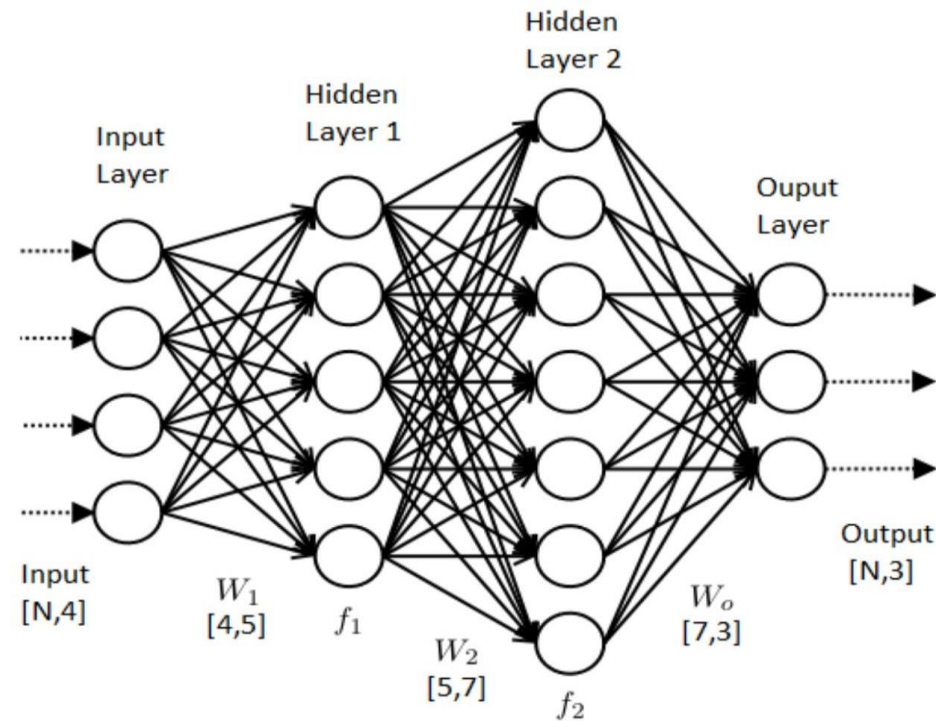
# Réseaux de neurones (8/11)



# Réseaux de neurones (9/11)

## Types d'architectures :

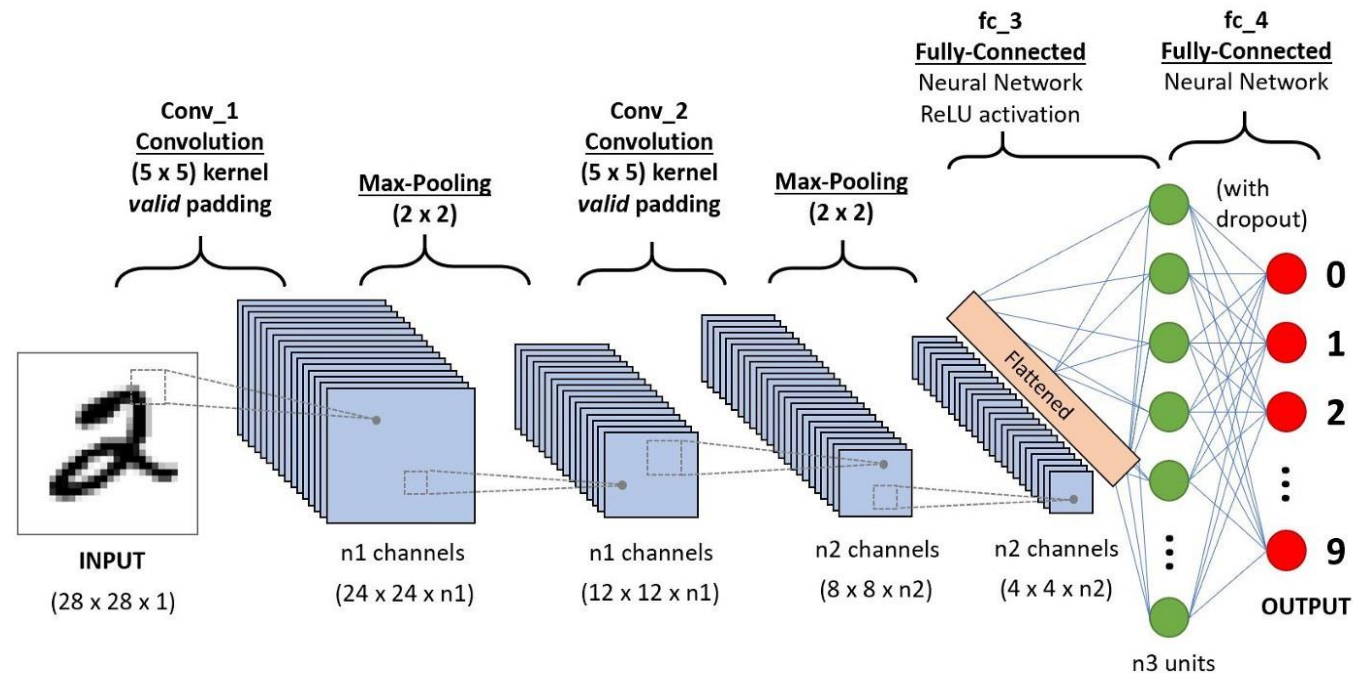
- MLP : Classification multi-classes



# Réseaux de neurones (10/11)

## Types d'architectures :

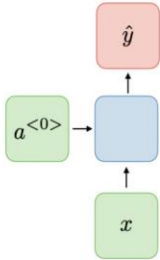
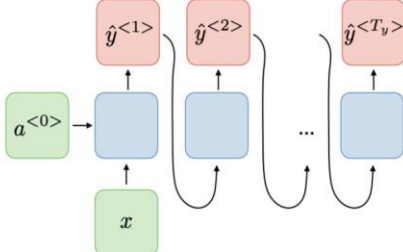
- CNN : Dédié aux images et aux données d'entrée 2D



# Réseaux de neurones (11/11)

## Types d'architectures :

- RNN : Dédié aux données ordonnées (séries temporelles, textes)

Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network
One-to-many $T_x = 1, T_y > 1$		Music generation



# k-Nearest Neighbor

# ***k-Nearest Neighbor (1/4)***

- **Définition** : algorithme supervisé de classification. L'algorithme des k plus proches voisins (*k-Nearest Neighbor*) classe les données en se basant sur leur similarité avec leurs voisins. **k est le nombre de points voisins à considérer pour la classification.**

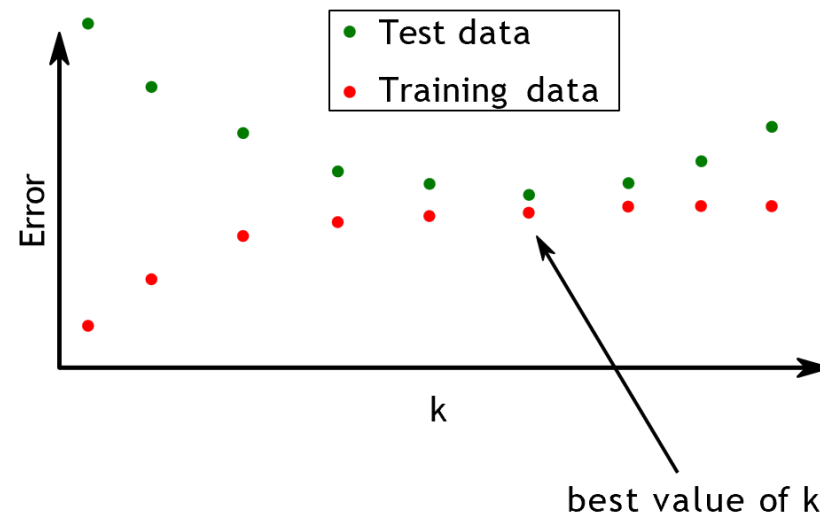
Très simple, cette méthode non paramétrique ne cherche pas à apprendre une relation entrée/sortie mais à répondre à la question suivante :

« **Dis-moi qui sont tes voisins, je te dirais qui tu es.** »

- **Naissance** : les k-NN ont été inventés par Evelyn Fix et Joseph Hodges en 1951 puis approfondis par Thomas Cover.
- **Applications** : biologie, thermique, recherche économique et sociale, algorithmes de recommandation, etc.

# ***k-Nearest Neighbor (2/4)***

- Le choix de  $k$ , le nombre de voisins à utiliser pour catégoriser une nouvelle donnée, va influencer la qualité de l'apprentissage. Sa **valeur optimale dépend du problème traité**, et correspond au point où l'erreur sur le test set commence à croître.
- La bonne pratique est de choisir un  **$k$  impair**, afin d'éviter les égalités.





# ***k-Nearest Neighbor (3/4)***

## **L'algorithme :**

Concrètement, connaissant les attributs  $A = X_1, X_2, \dots, X_D$ , où  $D$  est la dimension de l'espace de données, nous devons prédire la classification correspondante  $G = Y_1, Y_2, \dots, Y_N$  en se servant d'une métrique de proximité pour déterminer le degré de voisinage.

- Charger les données.
- Initialiser la valeur de  $k$ .
- Sur l'ensemble d'apprentissage.
  - Evaluer la proximité ou distance.
  - Ordonner selon la métrique.
  - Sélectionner les  $k$  premiers voisins.
  - Déterminer la classe la plus fréquente.
  - Affecter cette classe au point courant.

# ***k-Nearest Neighbor (4/4)***

## **Les points clés :**

- L'algorithme des k plus proches voisins a un entraînement paresseux. Pour compenser, le coût algorithmique d'une prédiction peut être élevé si la base de données d'entraînement est grande.
- La qualité des prédictions d'un algorithme des k plus proches voisins dépend principalement du choix d'une bonne distance ou similarité.
- L'algorithme des k plus proches voisins fonctionne mieux en faible dimension :
  - Son exécution est plus rapide.
  - Il est moins susceptible d'être biaisé par des variables qui ne sont pas pertinentes.
  - Il est moins susceptible de souffrir du fléau de la dimension.



# k-means

# ***k-means (1/2)***

- **Définition** : algorithme **non-supervisé de regroupement** (*clustering*) de données sans labels. **k est le nombre de cluster** que l'on souhaite obtenir.  
« **Qui se ressemble, s'assemble.** »
- **Naissance** : le *k-means* a été inventé par Stuard Lloyd à des fins de modulation d'impulsion codée. Après plusieurs implémentations, une version stable a été publiée en 1975 par Hartigan et Wong en Fortran.
- **Applications** : logistique, traitement du langage (bag of words), recherche économique et sociale, détection d'anomalie, compression d'image, etc.

# ***k-means (2/2)***

## **L'algorithme :**

Les points de l'ensemble d'apprentissage sont regroupés dans des **clusters définis selon leurs distances aux barycentres des clusters. Ces barycentres sont nommés centroïdes.**

- Choisir la valeur de  $k$ .
- Initialiser les  $k$  centroïdes.
- Tant qu'un critère de convergence n'est pas atteint à chaque itération :
  - Calculer les distances entre chaque point et chacun des centroïdes.
  - Assigner à chaque point le centroïde le plus proche selon une distance  $D$ . Cela définit les clusters  $C$   $\{\tilde{x}_i\}_{i=1,\dots,k}$  pour cette itération de l'algorithme.
  - Recalculer les centroïdes (par exemple, en prenant les points moyens de chaque cluster défini précédemment).



# Méthodes ensemblistes

# Méthodes ensemblistes (1/4)

- **Définition** : algorithme supervisé qui permet de faire de la classification ou de la régression.  
Les apprentissages ensemblistes nécessitent que très peu de paramètres. Ils sont fondés sur le principe de votes majoritaires en agrégeant des *classifiers faibles*.
- **Naissance** : les *Random Forests* ont été inventées en 2001 par Leo Breiman et Adele Cutler (Université de Princeton). Le nom de l'algorithme est une marque déposée depuis 2006.
- **Applications** : industries financières et bancaires, commerce électronique, santé et médecine, etc.

Une série de questions est représentée dans un arbre décisionnel (établir un diagnostic médical à partir d'une série de symptômes).

« Comment l'algorithme peut-il apprendre les bonnes questions à se poser ? »

Les questions sont posées dans un ordre précis et consistent à comparer une valeur à un seuil (curseur).



# Méthodes ensemblistes (2/4)

## Qu'est-ce qu'un algorithme d'ensemble :

L'apprentissage d'ensemble (ou *Ensemble Learning*) est un concept de Machine Learning dans lequel l'idée est de former plusieurs modèles utilisant le même algorithme d'apprentissage. Le terme ensemble fait référence à une **combinaison de modèles individuels** créant un modèle plus fort et plus puissant. Il s'agit de centaines ou de milliers d'apprenants avec un objectif commun fusionnés pour résoudre un problème.

*Remarque : plus les arbres sont profonds, plus grand est le risque d'overfitting.*

- **Random Forests (Forêts Aléatoires)** – Initialement, l'ordre et les seuils des questions sont aléatoires. Ce sont les curseurs que l'algorithme ajuste lors de l'entraînement. L'algorithme utilise plusieurs arbres entraînés séparément et opte pour la décision prise par la majorité : c'est ce qu'on appelle une méthode ensembliste.
- **XGBoost (Xtreme Gradient Boosting)** – C'est un modèle de machine Learning basé sur l'apprentissage d'ensemble séquentiel.

# Méthodes ensemblistes (3/4)

## Quand utiliser le *XGBoost* ?

- Lorsqu'il s'agit de données structurées/tabulaires de petite à moyenne taille, l'algorithme *XGBoost* est considéré comme le meilleur de sa catégorie à l'heure actuelle.
- Grâce à son principe d'auto-amélioration séquentielle, il peut être utilisé pour résoudre des problèmes de régression, de classification et même de classement. Il faut cependant éviter de choisir ce modèle pour résoudre des problèmes de Computer Vision, NLP, extrapolation de données ou dans les cas où le nombre de catégories est largement plus important que le nombre d'observations.
- Comme pour tous les autres algorithmes basés sur des arbres de décision, il faut faire attention à *l'overfitting* (sur-apprentissage). Pour cela *XGBoost* n'est pas le meilleur dans le cas de jeux de données très volumineux. Cela peut être corrigé grâce à des méthodes pour éviter le sur-apprentissage, par exemple on peut limiter la taille des arbres (*max\_depth*) ou bien construire les modèles sur des échantillons du jeu de données de base (on appelle cela *Stochastic Gradient Boosting*).

# Méthodes ensemblistes (4/4)

## ***Random Forests vs. XGBoost : cas d'applications***

- XGBoost vs Random Forest : prédire la gravité d'un accident de la route ([larevueia.fr](http://larevueia.fr))
- GitHub - IlyesTal/xgboost\_vs\_random-forest: XGBoost vs Random Forest pour prédire la gravité d'un accident



# Support Vector Machine

# Support Vector Machine (1/5)

- **Définition** : les **machines à vecteurs de support** ou **séparateurs à vaste marge** (pour garder l'acronyme anglais SVM) font partie de la famille des algorithmes d'apprentissage supervisé. Les **SVM** permettent de résoudre des problèmes de classification ou de régression et sont une généralisation des **classifieurs linéaires**.
- **Naissance** : les SVM ont été inventés en 1992 (Boser, Guyon et Vapnik) et sont issus du développement d'une théorie statistique de l'apprentissage (théorie de Vapnik-Chervonenkis ou théorie VC). Un brevet américain sur les SVM a été déposé en 1997 par les inventeurs originels.
- **Applications** : bio-informatique, data mining, computer vision, finance, etc.

# Support Vector Machine (2/5)

## L'algorithme :

- Il existe une infinité d'hyperplans séparateurs.  
« **Comment trouver l'hyperplan optimal ?** »
- L'objectif d'un SVM est de trouver l'hyperplan qui maximise la **marge**, i.e. la **distance minimale entre les points du jeu d'entraînement et l'hyperplan**.
- Les points du jeu d'entraînement les plus proches de la frontière sont appelés **vecteurs support**.

*Remarque : selon les données, la performance des SVM peut être équivalente voire supérieure à celle d'un réseau de neurones.*

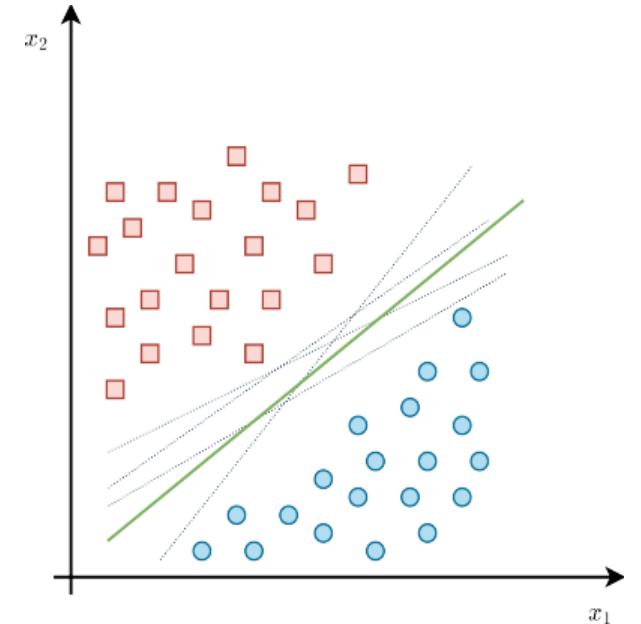


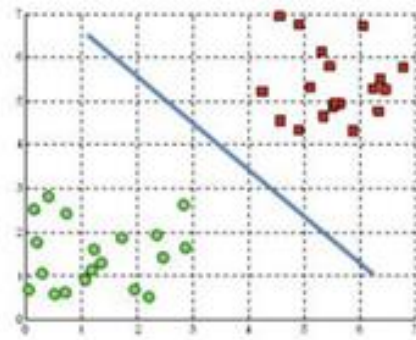
Figure: Exercice de classification (deux classes).  
Ici le problème est linéairement séparable.

# Support Vector Machine (3/5)

## Formalisme :

- **Définition** : Soit  $E$  un espace vectoriel de dimension finie  $n$  non nulle. Un **hyperplan** de  $E$  est un sous-espace vectoriel  $H$  de dimension  $n - 1$ .

A hyperplane in  $\mathbb{R}^2$  is a line



A hyperplane in  $\mathbb{R}^3$  is a plane

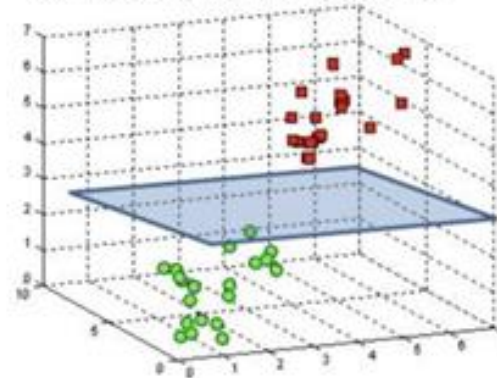


Figure: Un hyperplan de  $\mathbb{R}^2$  est une droite. Un hyperplan de  $\mathbb{R}^3$  est un plan.



# Support Vector Machine (4/5)

## Kernel Trick :

- Dans un cas réel concret, le problème est très souvent non linéairement séparable. Il faut alors utiliser une astuce...
- Il s'agit de l'astuce du noyau (**kernel trick**) qui consiste à reconsidérer le problème dans un espace de dimension supérieure où la probabilité d'existence d'un hyperplan séparateur est plus grande (cf. *Théorème de Cover*).
- Pour cela, on applique aux vecteurs  $X$  de l'espace d'entrée (aussi appelé espace de description) une transformation non-linéaire  $\phi$ .
- Dans l'espace d'arrivée  $\phi(X)$  (aussi appelé espace de redescription), on cherche alors l'hyperplan d'équation :

$$h(x) = l^T \cdot \phi(X) + b \quad \text{qui vérifie} \quad \forall k, l_k h(x_k) \geq 0.$$

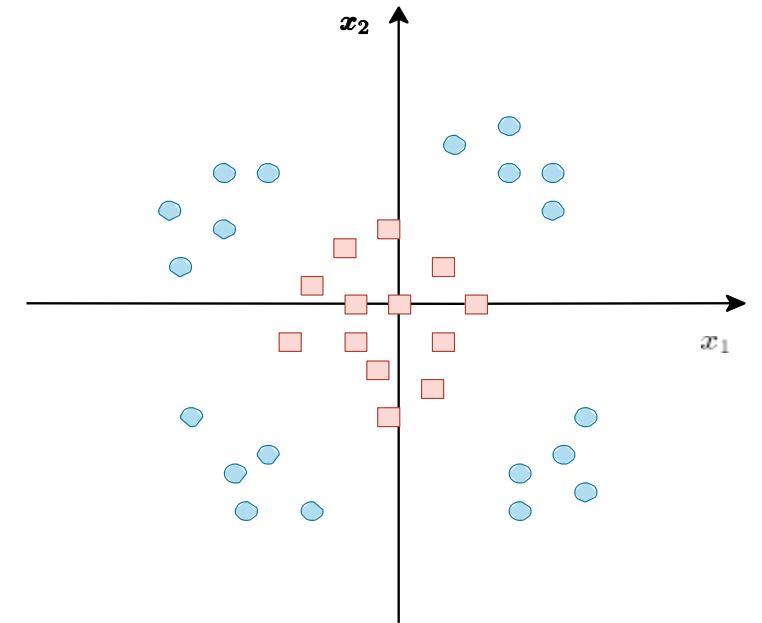


Figure: Exercice de classification (deux classes). Ici, le problème est non linéairement séparable.

# Support Vector Machine (5/5)

## Quelques références :

- [Vidéo Youtube](#) : *"Performing nonlinear classification via linear separation in higher dimensional space"*.
- Site Internet : Zeste de Savoir, *Un peu de Machine Learning avec les SVM*, 2020.
- Site Internet : Page FR Wikipédia sur les SVM, 2021.



# Algorithmes de règles d'association

# Algorithmes de règles d'association (1/4)

- **Définition** : famille d'algorithmes supervisés et non supervisés dont le but est l'**extraction de règles** à partir d'un jeu de données (des **règles d'associations dans le cadre non supervisé** et des **règles prédictives dans le cas supervisé**).
- **Naissance** : la recherche débute dans les années 60. Ces algorithmes ont été les premiers à être utilisés dans un contexte industriel (GASOIL et BTM en 1986 et 1990). L'algorithme le plus emblématique de recherche de règles d'associations est *APriori*.
- **Applications** : le domaine d'application est très varié. Ces algorithmes ont l'avantage d'avoir été pensés de façon générale. La contrepartie est qu'ils sont souvent moins performants dans un contexte supervisé.

# Algorithmes de règles d'association (2/4)

## Qu'est-ce qu'une règle ?

Une règle est une assertion du type *If ... Then ...* :

- La partie *If* de la règle, appelée **condition de la règle** ou **simplement règle**, est composée de la conjonction de **k tests** vérifiant si une feature (une coordonnée de X) vérifie une certaine propriété. Le nombre de k tests est appelé **longueur d'une règle**.
- La partie *Then* de la règle, appelée **conclusion quand la règle est active**, c'est-à-dire lorsque sa condition est satisfaite.

# Algorithmes de règles d'association (3/4)

## Non-supervisé :

Ils sont souvent utilisés dans les **systèmes de recommandation** ou la **prédiction séquentielle d'événements**.

- Exemple : **SI** A aime « Squid Game » **ALORS** proposer « Battle Royale ».
- Exemple : **SI** B aime « How I met your mother » **ALORS** proposer « Friends ».
- Exemple : **SI** C achète « lait, œufs, farine, sucre » **ALORS** proposer « pâte à tartiner ».
- Exemple : **SI** D ["rentre dans sa voiture", "il est 18:00"] **ALORS** proposer « itinéraire pour rentrer ».

Le but de ces algorithmes est d'identifier ce type de règle et de les sélectionner selon des critères statistiques.

# Algorithmes de règles d'association (4/4)

## Trois types d'algorithmes :

- **Arbres de décision** – *Classification and Regression Tree CART (1984), Iterative Dichotomizer 3 ID3 (1993).*
- **Ensemble de règles** – *RuleFit (2008), Sirius (2021), CoveringAlgorithm (2021).*
- **Liste de règles** – *Bayesian Rule List BRL, Scalable Bayesian Rule Lists SBRL.*

## Un algorithme by *Advestis part of Mazars* :

- [AdLearn](#) – Algorithme interprétable de génération de règles développé par *Advestis part of Mazars* pour traiter les séries temporelles (en production depuis 2013).





# Evaluation de la performance d'un algorithme

# Evaluation de la performance d'un algorithme de régression (1/3)

Contrairement à la classification, nous ne pouvons plus utiliser la métrique de précision. Nous devons utiliser des mesures d'erreurs spécifiquement conçues pour évaluer les prédictions faites sur des problèmes de régression.

Soit  $x_i$  un point d'un jeu de  $N$  données de test ayant  $y_i$  vrai label et  $\hat{y}_i$  prédit par un modèle de régression.

- **MAE (Mean Absolute Error)** – Elle représente la **moyenne de la différence absolue entre les valeurs réelles et les valeurs prédites**. Elle mesure la **moyenne des résidus** du jeu de données :

$$MAE = \sum_{i=1}^N |y_i - \hat{y}_i|$$

# Evaluation de la performance d'un algorithme de régression (2/3)

- **MSE (Mean Squared Error)** – Elle représente la **moyenne de la différence au carré entre les valeurs originales et les valeurs prédites**. Elle mesure la **variance des résidus** du jeu de données :

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

- **RMSE (Root Mean Squared Error)** – Il s'agit de la **racine carrée de la MSE**. Elle mesure l'**écart-type des résidus** du jeu de données :

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

# Evaluation de la performance d'un algorithme de régression (3/3)

- $R^2$  (Coefficient de détermination) – Il représente la **proportion de la variance de la variable dépendante** qui est expliquée par le modèle de régression linéaire.

Il s'agit d'un score sans échelle, c'est-à-dire que peu importe le fait que les valeurs soient petites ou grandes, la **valeur de  $R^2$  sera inférieure à 1** :

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

*Remarque : On note ce coefficient de détermination  $R^2$  car il s'agit du carré de la corrélation de Pearson entre les vraies valeurs et les valeurs prédites.*

# Evaluation de la performance d'un algorithme de classification (1/7)

Dans ce type de modèle, les données utilisées sont labellisées et le système doit pouvoir attribuer une classe à chaque observation.

Dans ce qui suit, nous nous intéresserons au cas particulier d'un classificateur binaire :

- Considérons un algorithme capable de prédire la présence d'un incendie dans une pièce donnée. Plusieurs cas sont alors possibles et sont rassemblés dans ce que l'on appelle une **matrice de confusion**.

		Classe réelle	
		-	+
Classe prédite	-	True Negatives (vrais négatifs)	False Negatives (faux négatifs)
	+	False Positives (faux positifs)	True Positives (vrais positifs)

Figure: Matrice de confusion d'un classificateur binaire.

(Source : [www.openclassroom.com](http://www.openclassroom.com))

# Evaluation de la performance d'un algorithme de classification (2/7)

En apprentissage supervisé, une matrice de confusion permet de qualifier rapidement un système de classification.

Appelons positive la classe correspondant à un incendie et négative l'autre classe (pas d'incendie).

		Classe réelle	
		-	+
Classe prédite	-	True Negatives (vrais négatifs)	False Negatives (faux négatifs)
	+	False Positives (faux positifs)	True Positives (vrais positifs)

- **Vrai négatif (TN)** : le système ne prédit pas d'incendie et il n'y en a effectivement pas.
- **Faux négatif (FN)** : le système ne prédit pas d'incendie alors qu'il y en a bien un.
- **Faux positif (FP)** : le système prédit la présence d'un incendie mais il n'y en a pas.
- **Vrai positif (TP)** : le système prédit la présence d'un incendie et il y en a bien un.

## Question :

« Dans cet exemple, y-a-t-il des cas plus importants à considérer que d'autres ? »

# Evaluation de la performance d'un algorithme de classification (3/7)

A partir de la matrice de confusion, nous pouvons dériver plusieurs critères de performance :

- **Accuracy** – Il s'agit de la **proportion d'éléments correctement prédits**. Ce critère reflète la **performance globale** de notre modèle :

$$accuracy = \frac{TN + TP}{TN + FN + FP + TP}$$

- **Recall (Rappel)** – Il s'agit du **taux de vrais positifs**, c'est-à-dire la **proportion de positifs correctement identifiés**. Ce critère reflète la **capacité de notre modèle à détecter tous les incendies** :

$$recall = \frac{TP}{TP + FN}$$

*Remarque : Il est facile d'obtenir un excellent score de rappel en prédisant tout le temps positif. Aucun incendie ne nous échappera mais il s'agit de tout sauf d'apprentissage automatique !*



# Evaluation de la performance d'un algorithme de classification (4/7)

A partir de la matrice de confusion, nous pouvons dériver plusieurs critères de performance :

- **Specificity (Spécificité)** – Il s'agit du **taux de vrais négatifs**. Ce critère reflète la **capacité de notre modèle à détecter toutes les situations où il n'y a pas d'incendie** :

$$specificity = \frac{TN}{FP + TN}$$

- **Precision (Précision)** – Il s'agit de la **proportion de prédictions correctes parmi les éléments qui ont été prédits positifs**. Ce critère reflète la **capacité de notre modèle à ne déclencher une alarme qu'en présence d'un incendie** :

$$precision = \frac{TP}{TP + FP}$$

*Remarque : Là encore, il est possible d'avoir une très bonne précision en ne prédisant que très peu de fois positif. On risque moins de se tromper mais c'est très dangereux !*

# Evaluation de la performance d'un algorithme de classification (5/7)

Une bonne pratique consiste à utiliser un critère de compromis entre le **rappel** et la **précision** que l'on appelle le **F1-score**. Il s'agit simplement de leur **moyenne harmonique** :

$$F_{1\text{ Score}} = 2 \times \frac{\textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}} = \frac{2TP}{2TP + FP + FN}$$

# Evaluation de la performance d'un algorithme de classification (6/7)

La **courbe ROC** (de l'anglais *Receiver Operating Characteristic*) est une courbe représentant le **taux de vrais positifs (*rappel*)** en fonction du **taux de faux positifs ( $1 - \text{spécificité}$ )**. Elle est obtenue en faisant varier un seuil de discrimination.

→ Plus la courbe tend à épouser les axes supérieurs gauches, plus le modèle est performant.

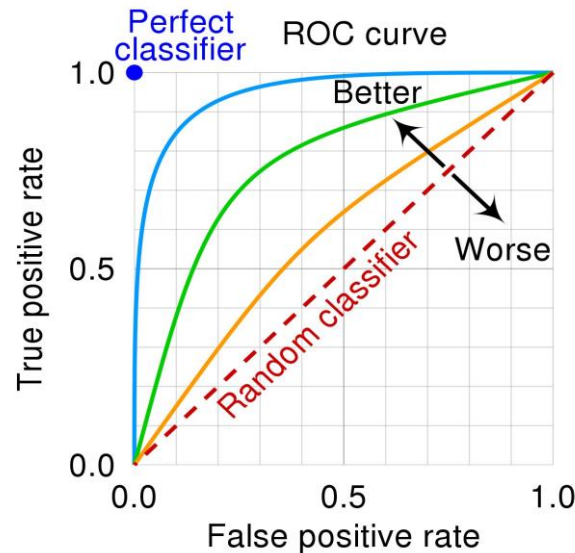


Figure: Une courbe ROC. La performance d'un modèle naïf (aléatoire) est indiquée en pointillés.

(Source : [www.wikipedia.com](http://www.wikipedia.com))

# Evaluation de la performance d'un algorithme de classification (7/7)

Alors que la courbe ROC est une **courbe de probabilité**, l'aire sous la courbe ROC notée **AUC** (*Area Under the Curve*) représente le **degré ou la mesure de séparabilité**.

→ Plus le critère AUC est élevé, meilleur est le modèle à distinguer les deux classes.

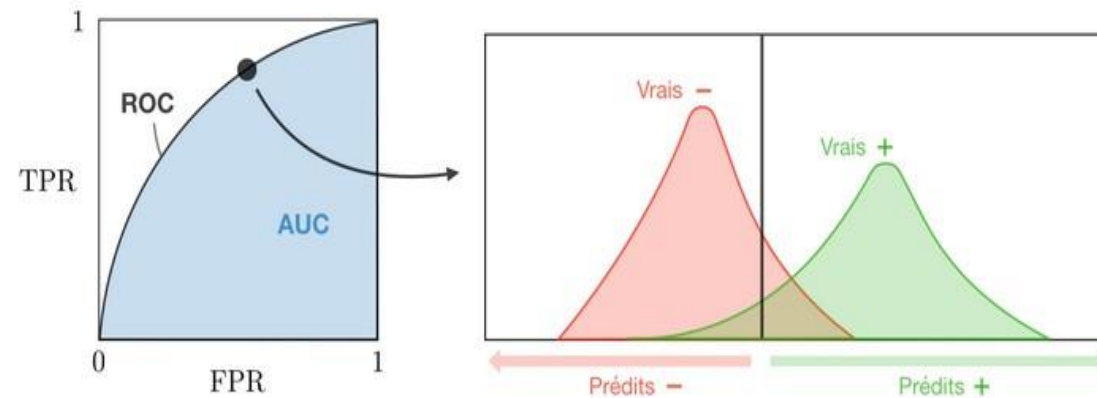


Figure: Illustration du critère AUC.  
(Source : [www.stanford.edu](http://www.stanford.edu))

# *Train, Validation, Test*

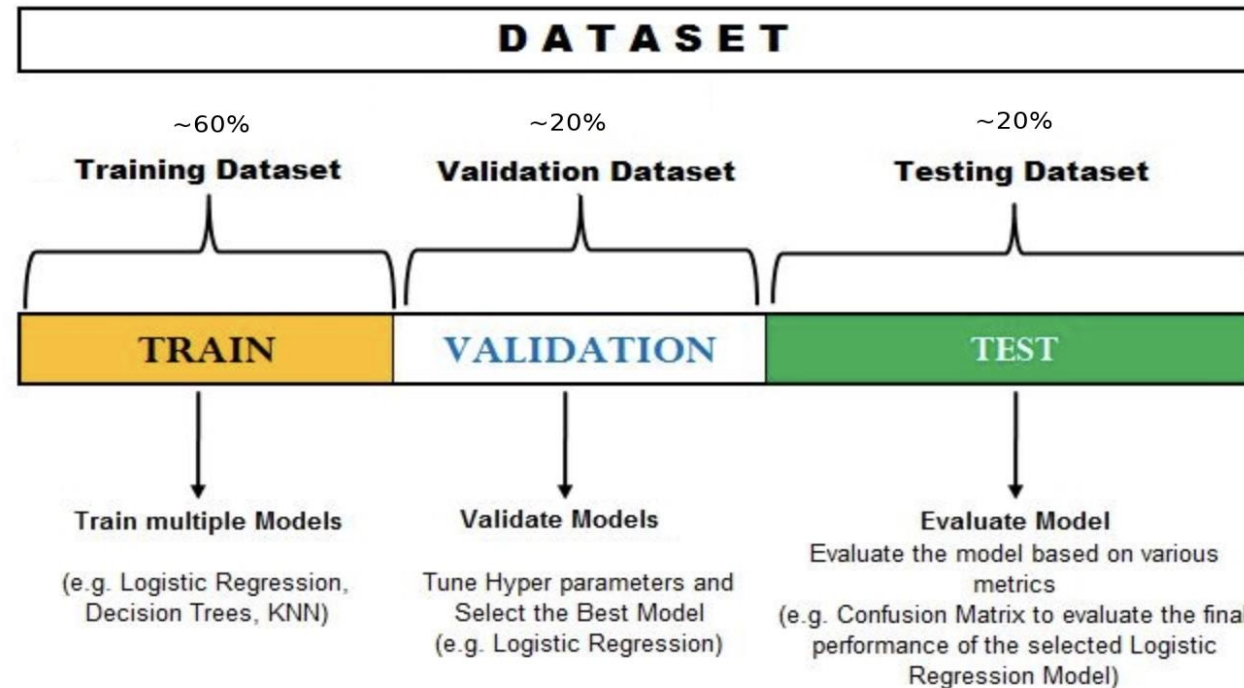


Figure: Séparation d'un jeu de données en 3 sous-ensembles d'apprentissage.

# Cross-validation

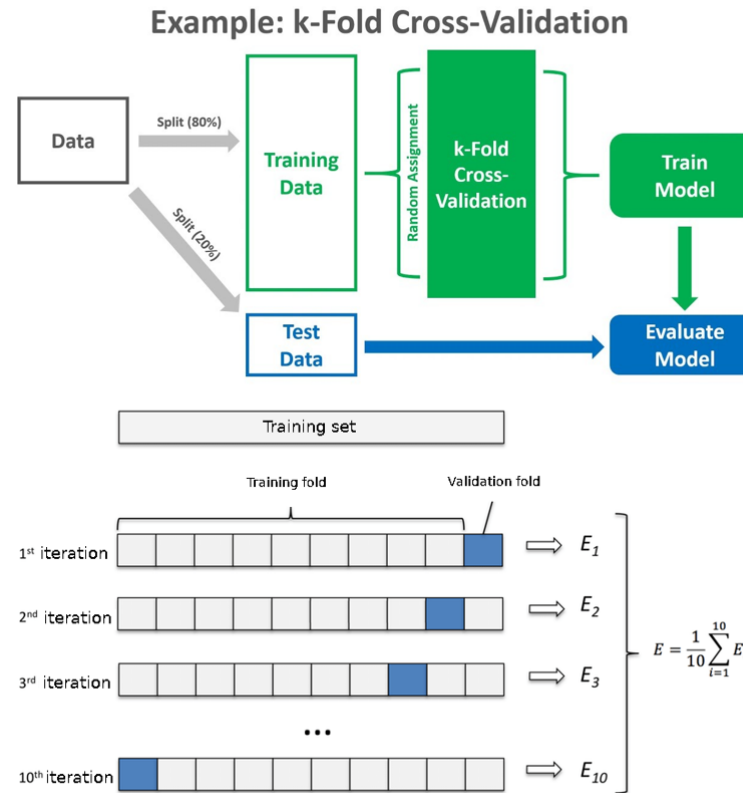


Figure: Exemple d'un k-Fold Cross-validation.

# Underfitting et overfitting (1/4)

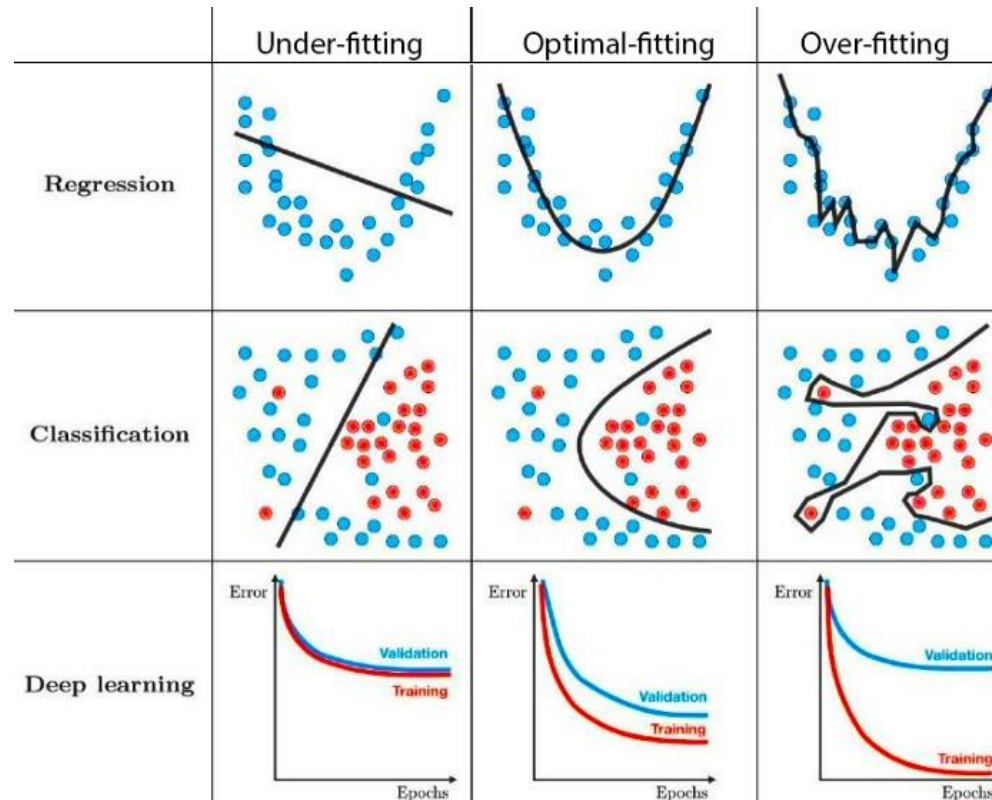


Figure: Underfitting (sous-interprétation) et overfitting (sur-interprétation).

(Source : [www.towardsdatascience.com](http://www.towardsdatascience.com))



# Underfitting et overfitting (2/4)

Compromis biais/variance :

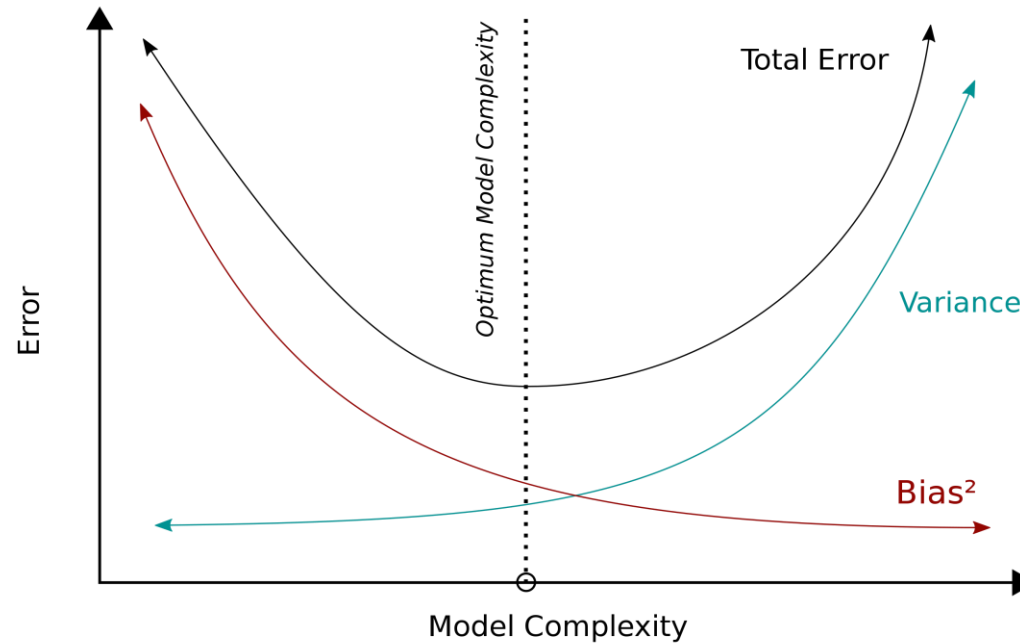


Figure: Compromis biais / variance.  
(Source : [www.wikipedia.com](http://www.wikipedia.com))

# Underfitting et overfitting (3/4)

Les trois ingrédients d'un algorithme d'apprentissage supervisé sont :

- L'espace des hypothèses
- La fonction de coût
- L'algorithme d'optimisation qui permet de trouver l'hypothèse optimale au sens de la fonction de coût sur les données (minimisation du risque empirique).

Le **compromis biais/variance** traduit le **compromis entre l'erreur d'approximation**, correspondant au biais de l'algorithme d'apprentissage, et **l'erreur d'estimation**, correspondant à sa variance.

*Remarque : les algorithmes trop puissants avec beaucoup de curseurs ont une faible perte sur les données d'entraînement mais généralisent mal sur les données de tests.*

La **généralisation** et le **sur-apprentissage** sont des préoccupations majeures en machine learning :

*« Comment s'assurer que des modèles entraînés pour minimiser leur erreur de prédiction sur les données observées seront généralisables aux données pour lesquelles il nous intéresse de faire des prédictions ? »*

# *Underfitting et overfitting (4/4)*

## **Points clés :**

- Pour éviter le sur-apprentissage, il est essentiel lors de l'étape de sélection du modèle de valider les différents modèles testés sur un jeu de données différent de celui utilisé pour l'entraînement.
- Pour estimer la performance en généralisation d'un modèle, il est essentiel de l'évaluer sur des données qui n'ont été utilisées ni pour l'entraînement, ni pour la sélection de ce modèle.
- De nombreux critères permettent d'évaluer la performance prédictive d'un modèle. On les choisira en fonction de l'application.



# Formation par la pratique en python

# Environnements de développement

## IDE (environnement de développement intégré)

- *PyCharm* (développé par JetBrains)
- *Visual Studio Code* (développé par Microsoft)
- *Jupyter* (développé par Fernando Pérez et Brian Granger) & *Google Collab* (développé par Google)

## Librairies python essentielles

- *Numpy* & *Pandas* (manipulation des données)
- *Request* & *BeautifulSoup* (client HTTP & web scrapping)
- *Matplotlib* & *Plotly* & *Seaborn* (visualisation des données)
- *Statsmodel* (statistiques)
- *Scipy* (calculs scientifiques)
- *Scikit-learn* (machine learning)
- *TensorFlow* & *PyTorch* (IA)
- *Mlflow* (MLOps)

# Formation par la pratique en python

## Quelques références :

- **Coursera / Deeplearning.AI** : Andrew Ng's Machine Learning Collection.
- **Patrick Wampé (collection eni)** :
  - Machine Learning et Deep Learning.
  - Intelligence artificielle vulgarisée.
- **Librairie Scikit-learn** : Machine learning in Python — [scikit-learn 1.3.2 documentation](https://scikit-learn.org/stable/documentation/1.3.2/)
- **Formations Victor Chemla & PLB**



# Démo





# Évaluez vos connaissances

QUIZZ / QCM / TP

# Evaluations finales

Auto- évaluation  
Evaluation de la satisfaction

**Merci pour votre attention !**