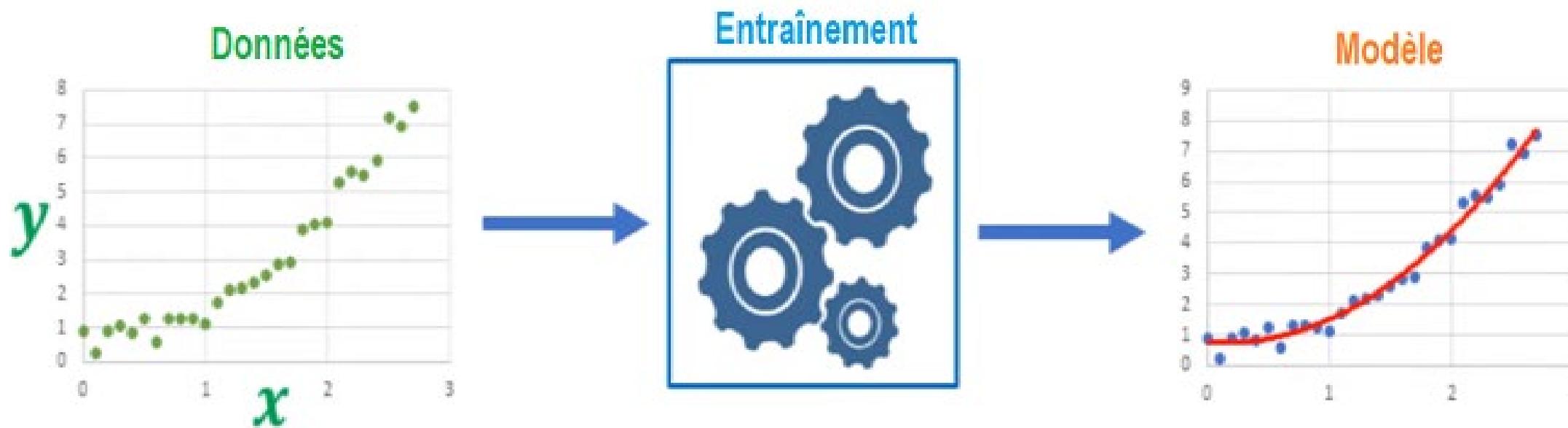


Introduction à la data science

Machine Learning

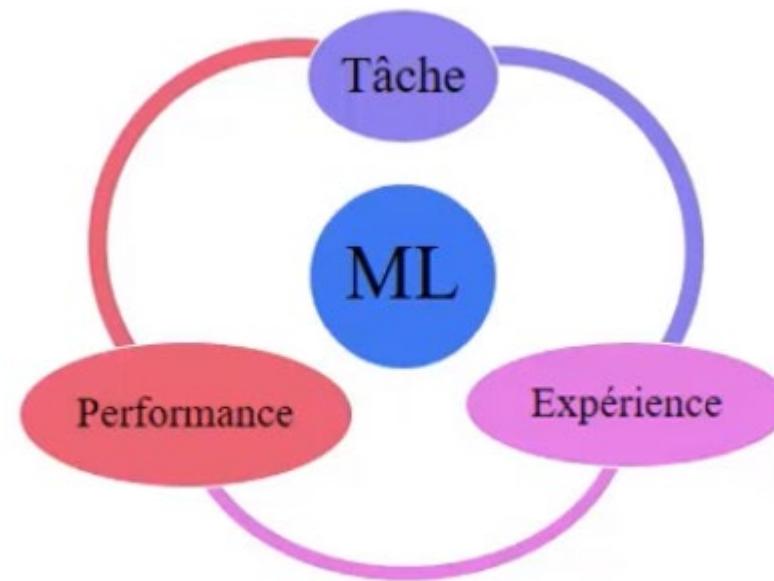
Machine learning

Définition : Machine Learning (ML), inventé par *Arthur Samuel* (Samuel, 1959), est une technologie d'intelligence artificielle permettant aux ordinateurs d'apprendre à partir de données sans avoir été programmés explicitement à cet effet. Il s'agit d'une science moderne permettant de découvrir des modèles et d'effectuer des prédictions à partir de données en se basant sur des statistiques et les analyses prédictives. Autrement dit, ML c'est l'application des méthodes de DM par la machine. En fait, le ML consiste à développer un modèle mathématique à partir de données expérimentales.



Machine learning

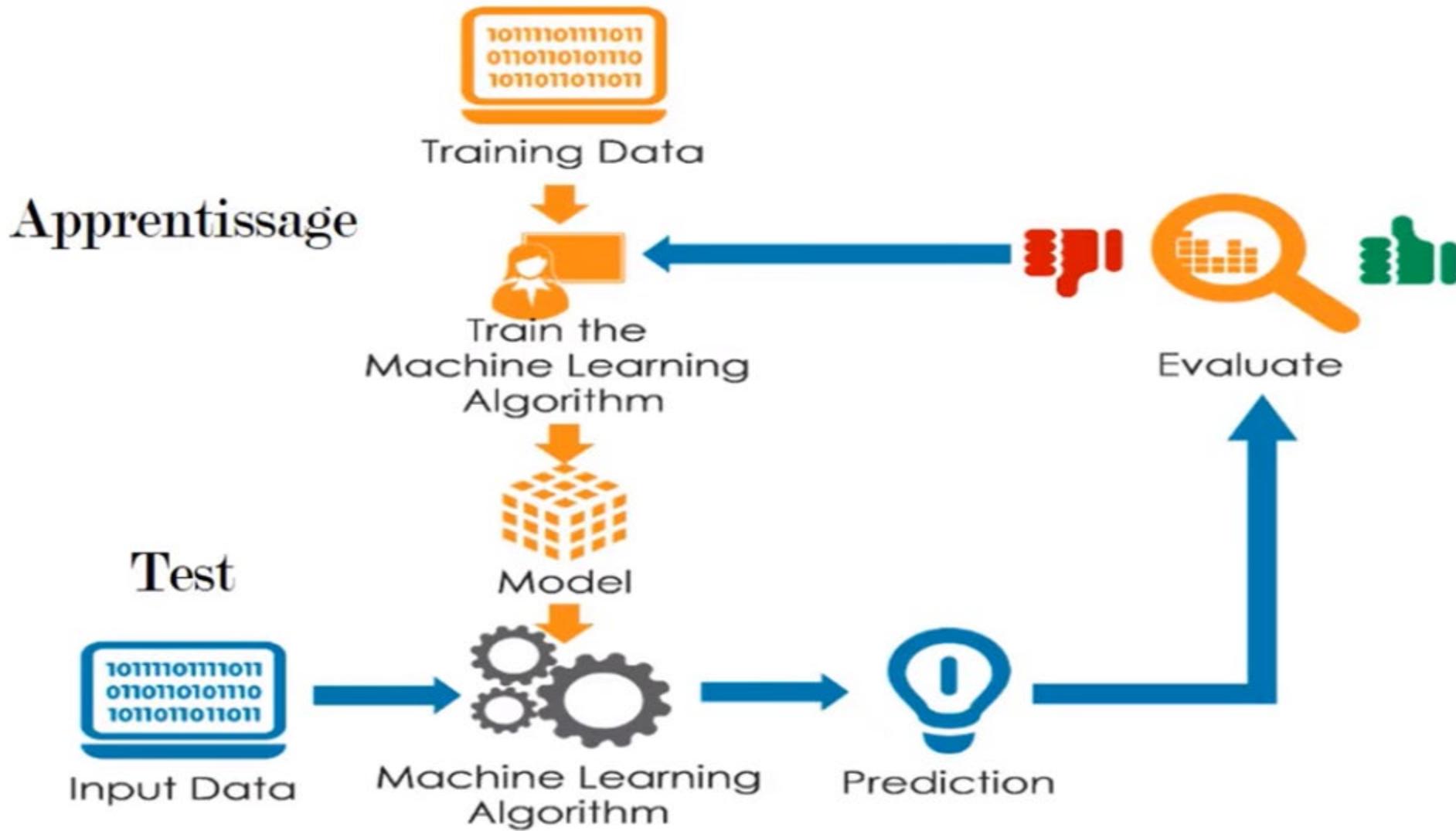
Définition : selon *Tom Mitchell*, Un programme d'ordinateur apprendrait de l'expérience E en ce qui concerne une tâche T et une mesure de rendement P, si son rendement sur T, tel que mesuré par P, s'améliore avec l'expérience E.



Apprendre à faire T = Améliorer P grâce à E

Machine learning

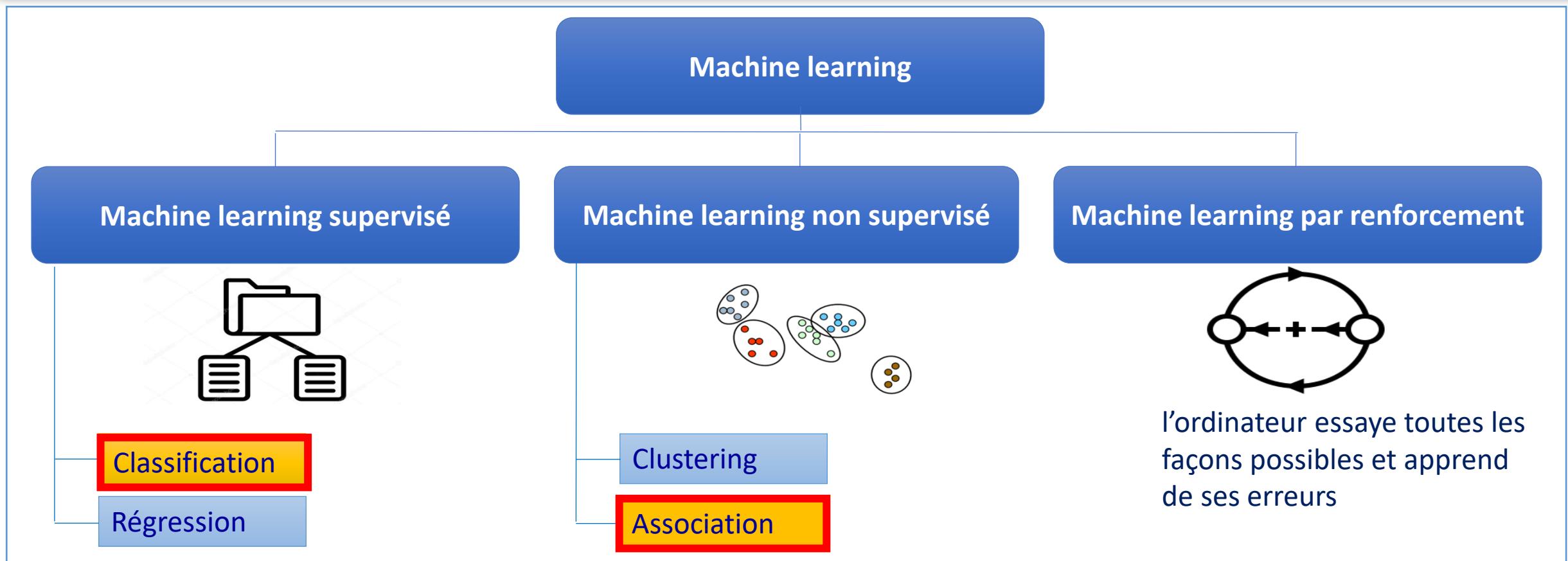
Fonctionnement de machine learning



Machine learning

Type de Machine learning

- ML est une technologie d'intelligence artificielle permettant aux ordinateurs d'apprendre à partir des données sans avoir été programmés explicitement à cet effet. Il consiste à développer un modèle mathématique à partir de données expérimentales. Il existe trois techniques:



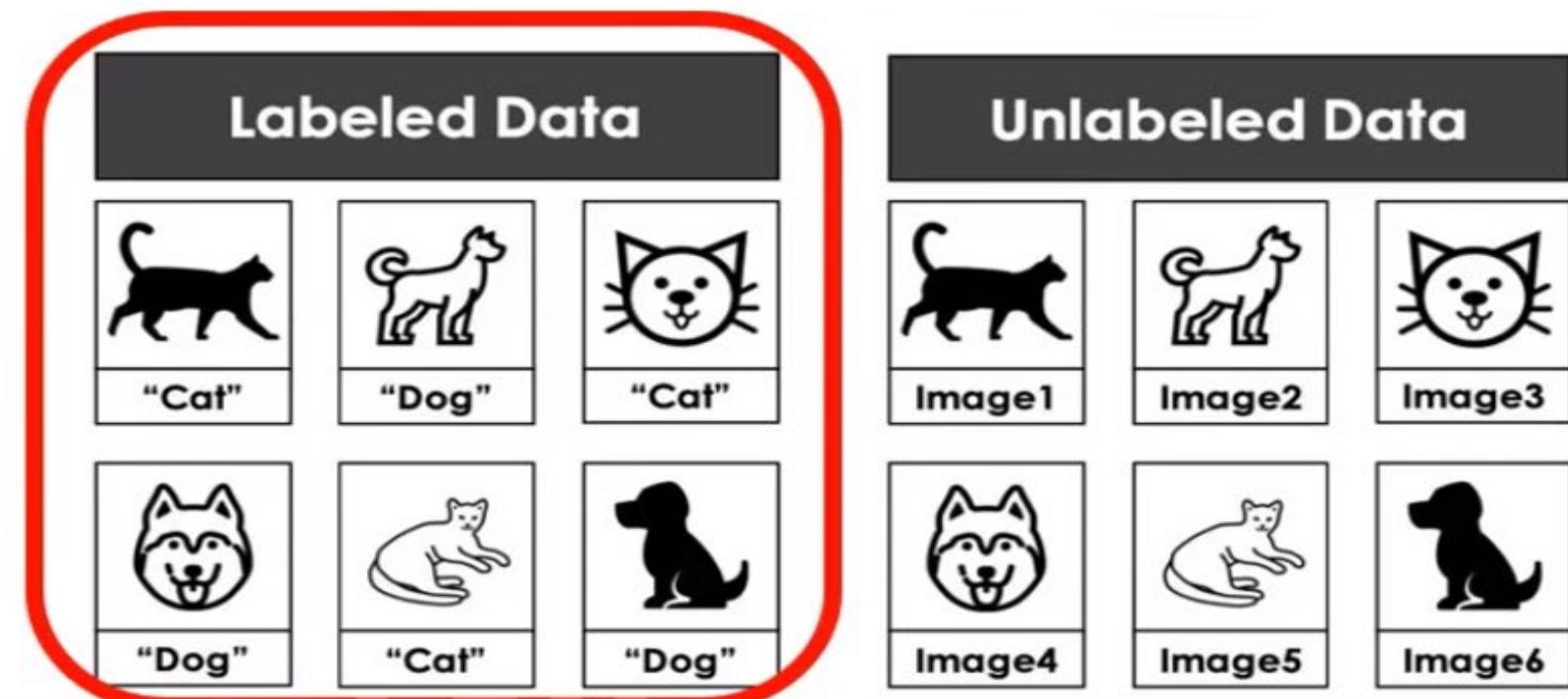
Machine learning

Machine learning supervisé

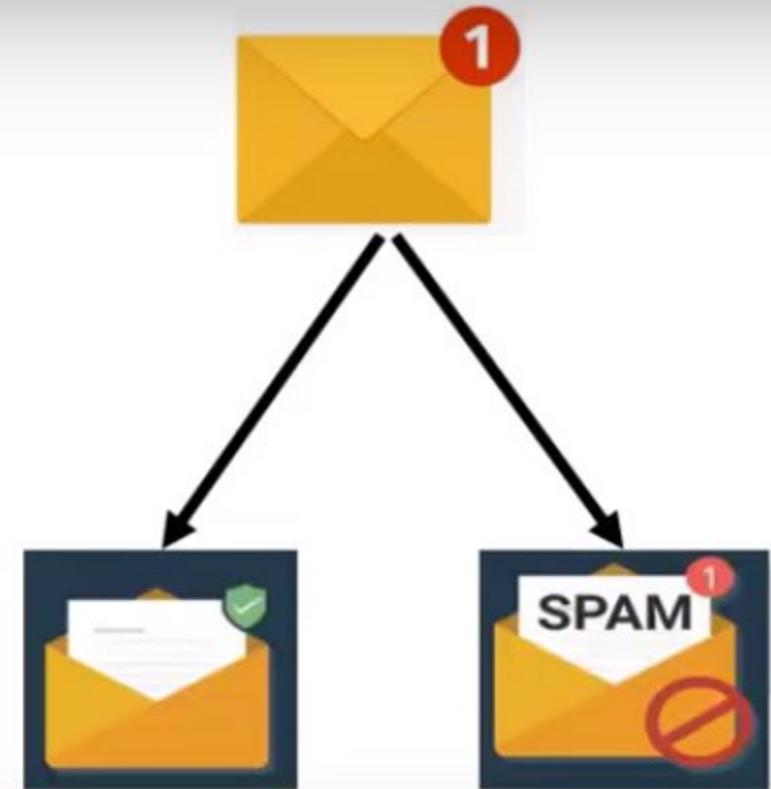
La machine connaît déjà les réponses qu'on attend d'elle. Elle travaille à partir de **données labellisées**



Dataset



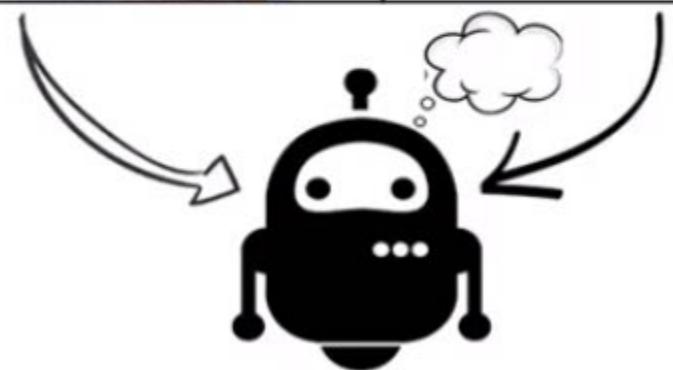
Données labellisées



Machine learning

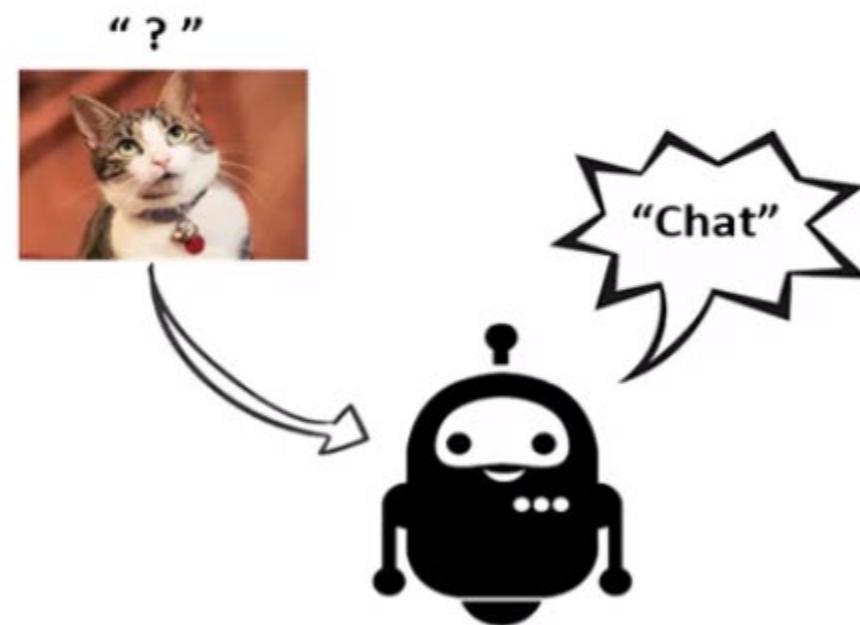
Machine learning supervisé

x	y
	“Chien”
	“Chien”
	“Chat”
	“Chien”



Apprentissage Supervisé

$$f: x \rightarrow y$$



Utilisation finale

Machine learning

Machine learning supervisé

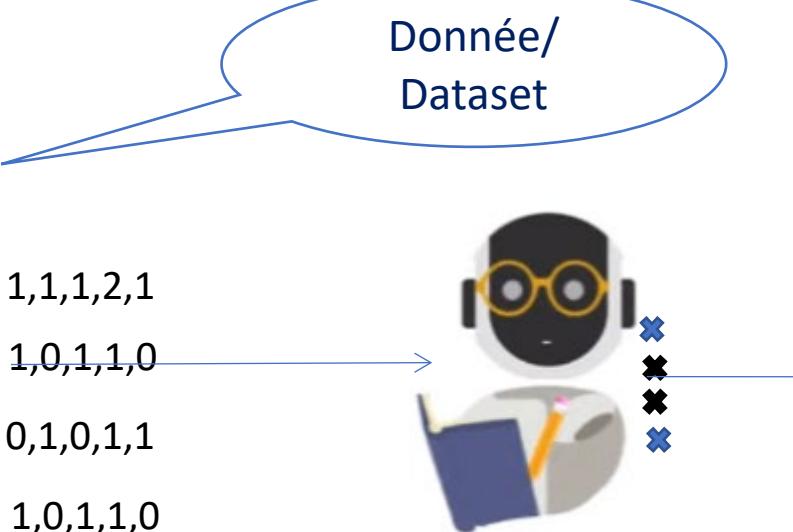
- La machine reçoit des données caractérisées par des variables X et annotées d'une variable Y .

Attributs (features)				Classe
$x_{1,1}$	$x_{1,2}$...	$x_{1,p}$	y_1
$x_{2,1}$	$x_{2,2}$...	$x_{2,p}$	y_2
...
$x_{n,1}$	$x_{n,2}$...	$x_{n,p}$	y_n

Prédire Y en fonction de X , Trouver la relation entre X et Y

Attributs de la maladie				Existence
1	1	1	2	1
1	0	1	1	0
0	1	0	1	1
1	0	1	1	0

1,1,1,2,1
1,0,1,1,0
0,1,0,1,1
1,0,1,1,0



Donnée/
Dataset

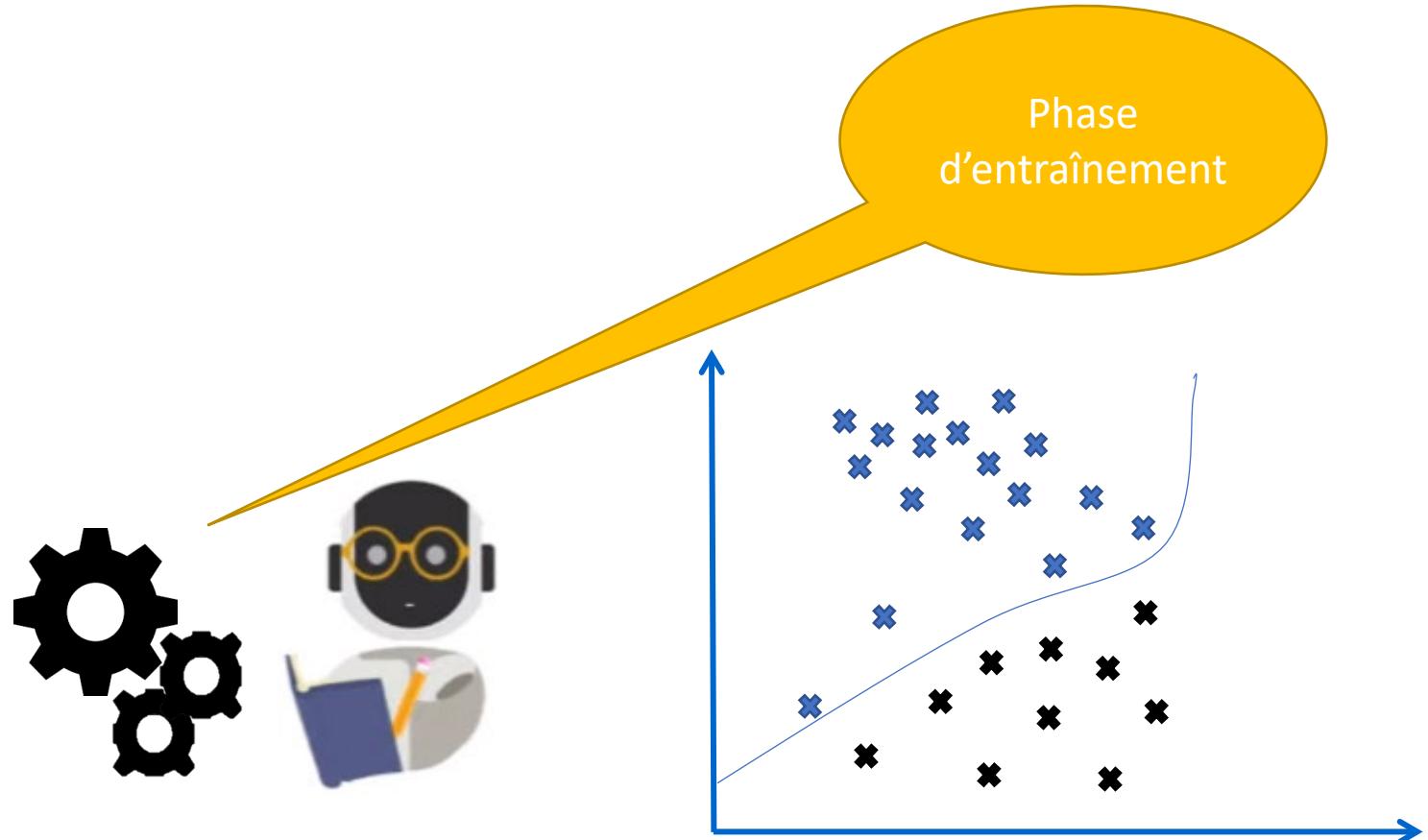
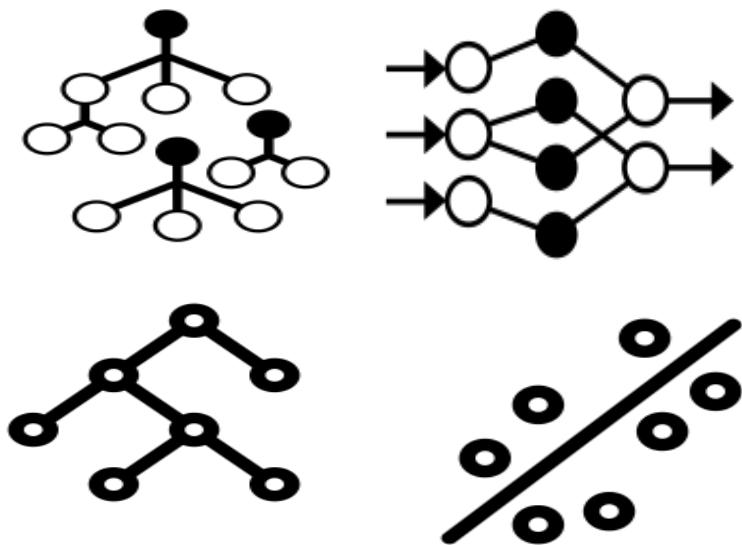
Cas négatifs

Cas positifs

Machine learning

Machine learning supervisé

- Spécifier quel type de modèle la machine doit apprendre en précisant les hyper-paramètres du modèle.

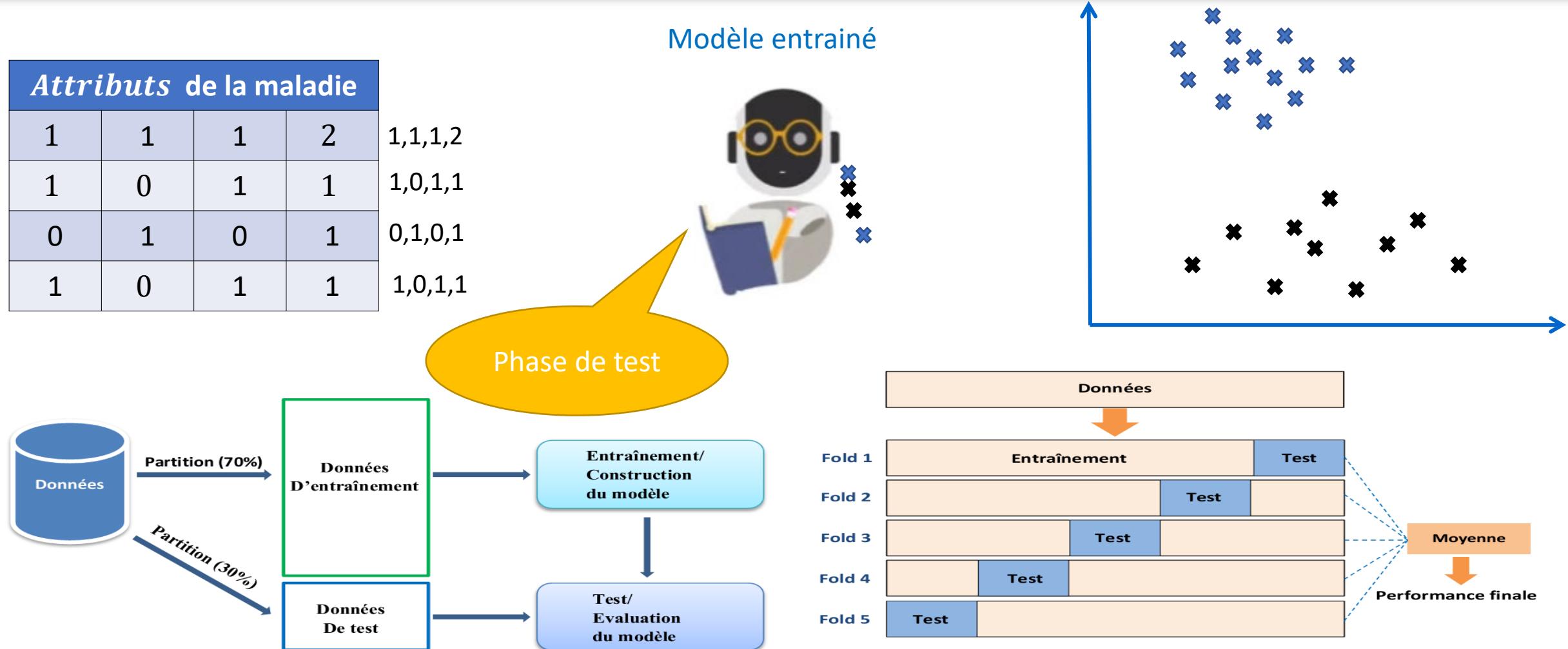


Machine learning

Machine learning supervisé

- La machine reçoit des données caractérisées par des variables X mais cette fois-ci sans la classe (y)

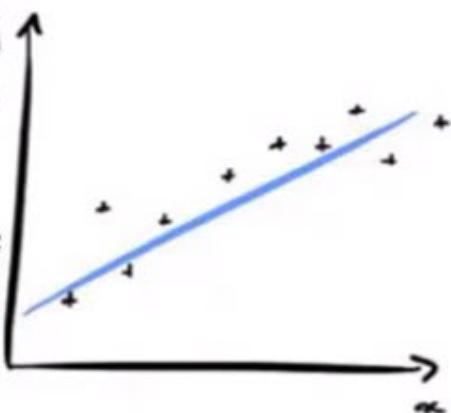
Attributs de la maladie			
1	1	1	2
1	0	1	1
0	1	0	1
1	0	1	1



Régression

Les problèmes de **Régression** correspondent aux situations dans lesquelles la machine doit prédire la valeur d'une **variable quantitative** (variable continue)

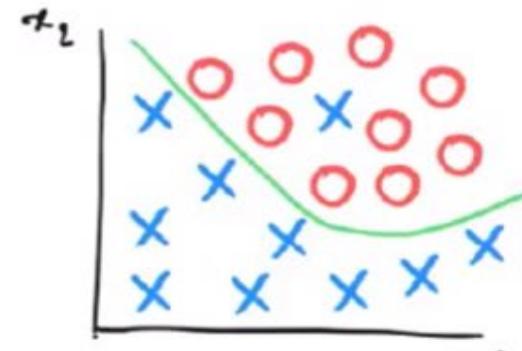
- Le prix d'un appartement
- L'évolution du climat
- Consommation électrique
- Durée de vie d'un patient



Classification

La classification correspondent aux situations dans lesquelles la machine doit prédire la valeur d'une variable qualitative (variable discrète). Autrement dit, la machine doit classer ce qu'on lui donne dans des classes.

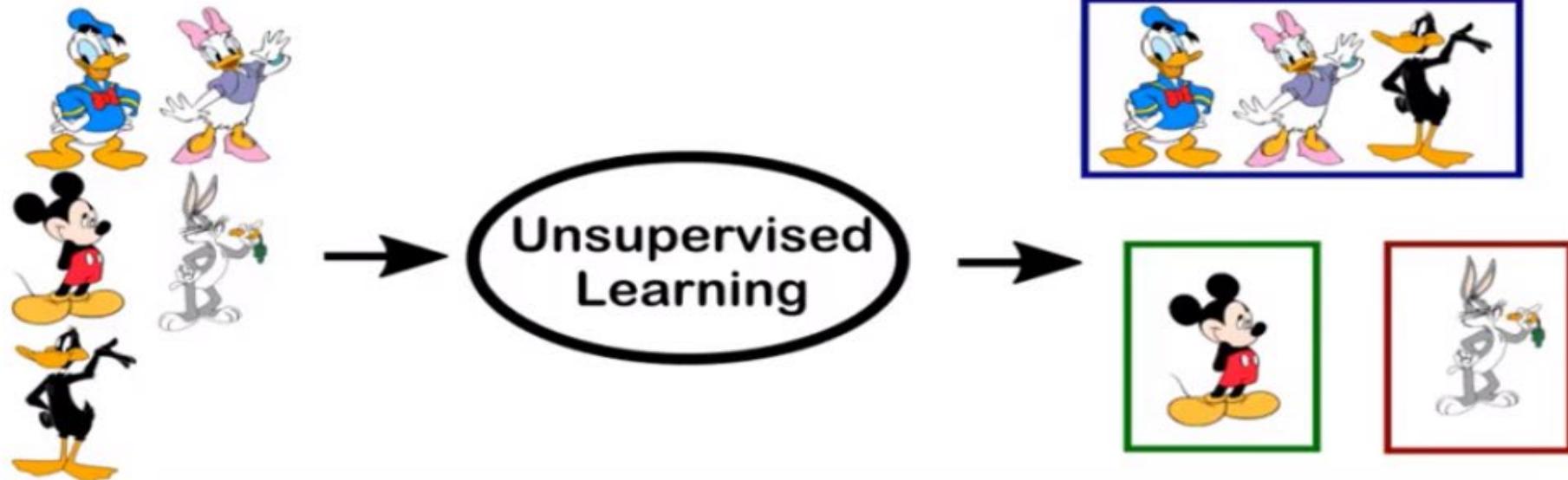
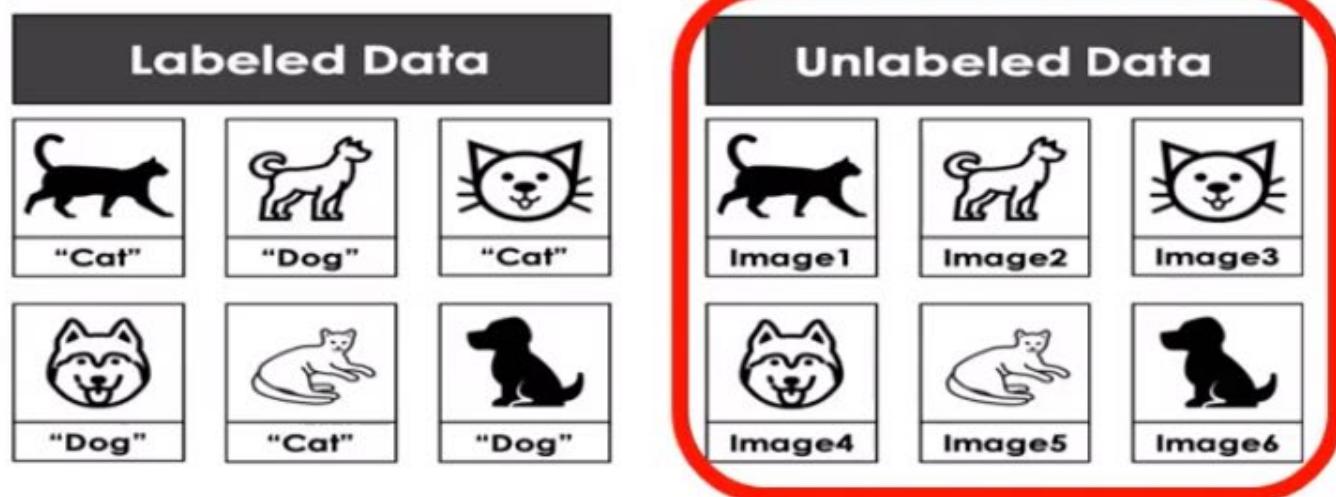
- Email Spam / non Spam
- Cancer / non Cancer
- Photo de Chat / Chien



Machine learning

Machine learning non supervisé

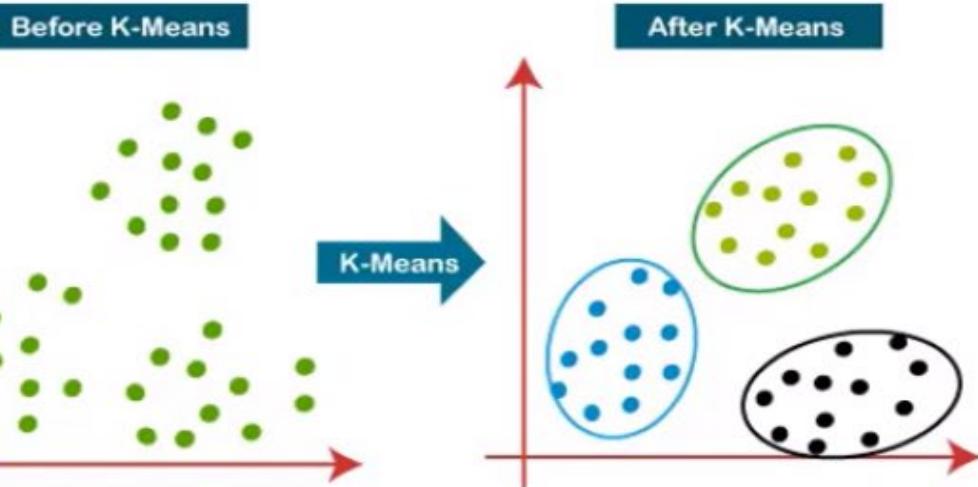
Les réponses que l'on cherche à prédire ne sont pas disponibles dans les jeux de données.



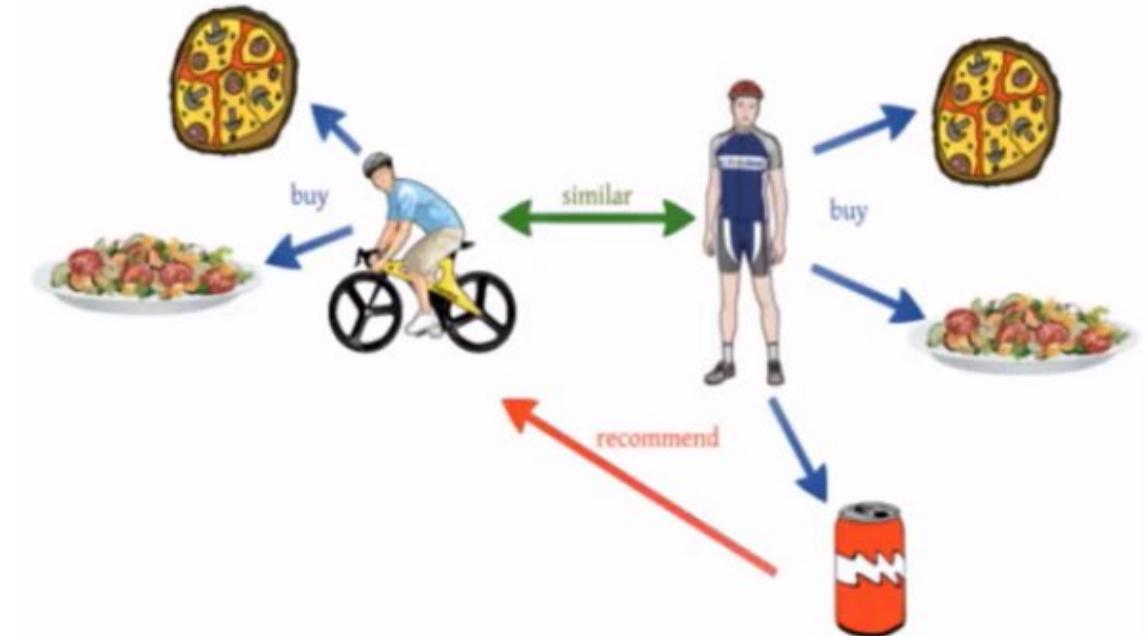
Machine learning

Machine learning non supervisé

Clustering



Filtrage collaboratif



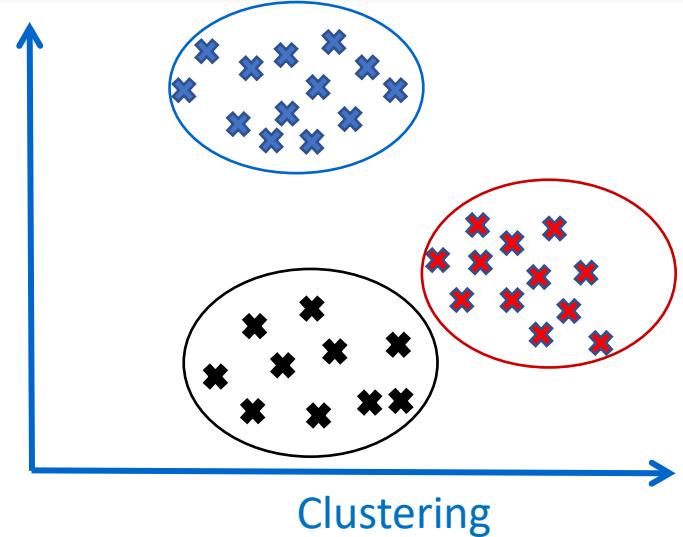
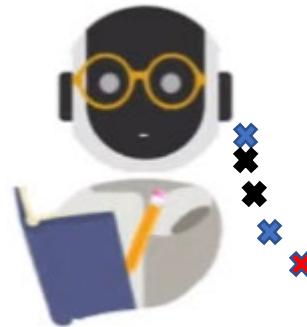
Machine learning

Machine learning non supervisé

- Utilisé lorsque les informations utilisées pour la l'apprentissage ne sont étiquetées.
- Décrire une structure cachée à partir de données non étiquetées.

Attributs de la maladie			
1	1	1	2
1	0	1	1
0	1	0	1
1	0	1	1

Entraînement de modèle



Lait œufs
sucre pain



Client 1

Lait œufs
pain céréales



Client 2

œufs sucre



Client 3

- Qu'est-ce que mon client achète ?
- Quels produits sont achetés ensemble ?

Association

Machine learning

Mesures de performance des modèles de ML

La matrice de confusion

	Prédit comme positif	Prédit comme négatif
Positif actuel	Vrai positif (TP)	Faux négatif (FN)
Négatif actuel	Faux positif (FP)	Vrai négatif (TN)

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

Le rapport entre les cas correctement classés et le nombre total des cas.

$$Spécificité = \frac{TN}{TN + FP}$$

Les instances négatives correctement classées par rapport au nombre total d'instances négatives

$$Sensibilité(Rappel) = \frac{TP}{TP + FN}$$

Les instances positives correctement classées par rapport au nombre total d'instances positives

$$Précision = \frac{TP}{TP + FP}$$

Les instances positives correctement classées par rapport au nombre total d'instances positives prédites

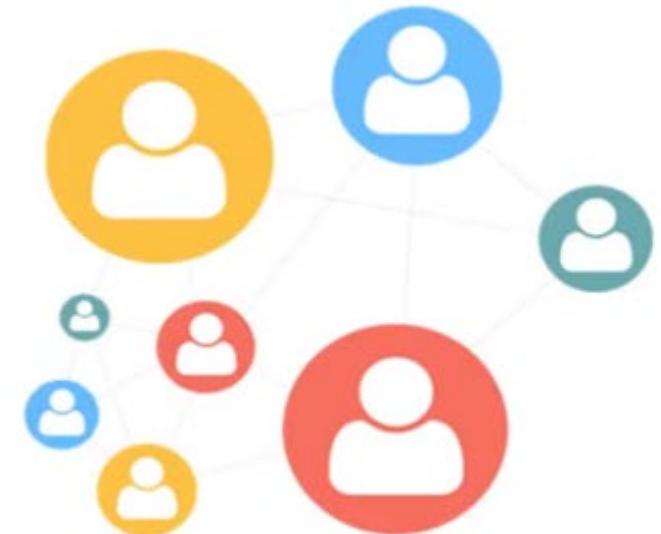
$$F - measure = \frac{2 * Précision * Rappel}{Précision + Rappel}$$

Elle prend en compte à la fois la précision et le rappel à calculer

- **Algorithmes de Machine learning**

- **Régression linéaire (RL) (Linear Regression (LR))**

La régression linéaire est un algorithme de machine learning qui consiste à trouver la meilleure fonction permettant de définir une variable de sortie(l' élément à prédire) à partir des variables explicatives en entrée(les prédicteurs)



Prédire le prix d'un appartement à partir de sa superficie et sa position

- **Algorithmes de Machine learning**

- **Régression linéaire (RL) (Linear Regression (LR))**

- **Principe**

La variable cible (**Y**) est quantitative

Le prix d'une maison



La variable (**X**) est quantitative ou qualitative

Nombre de pièces
Localisation

Avec ou sans balcon
Nombre de façades

...

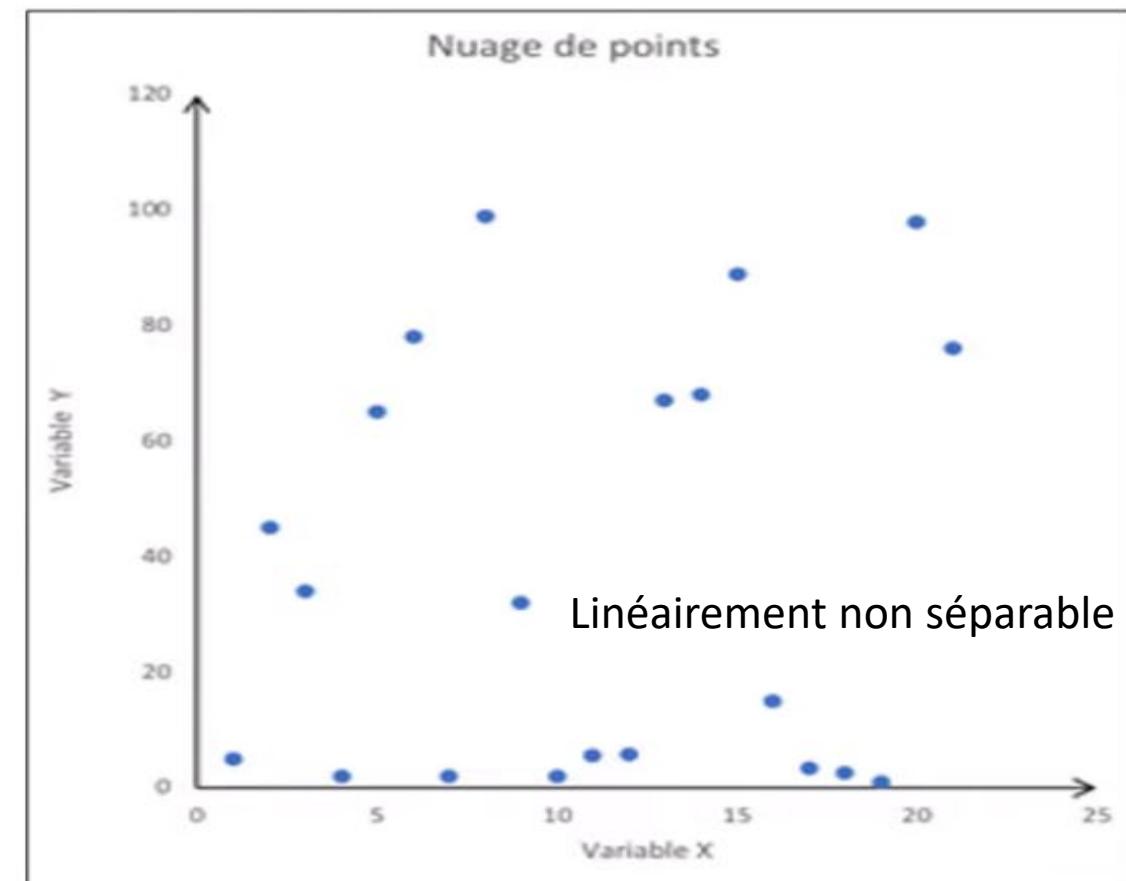
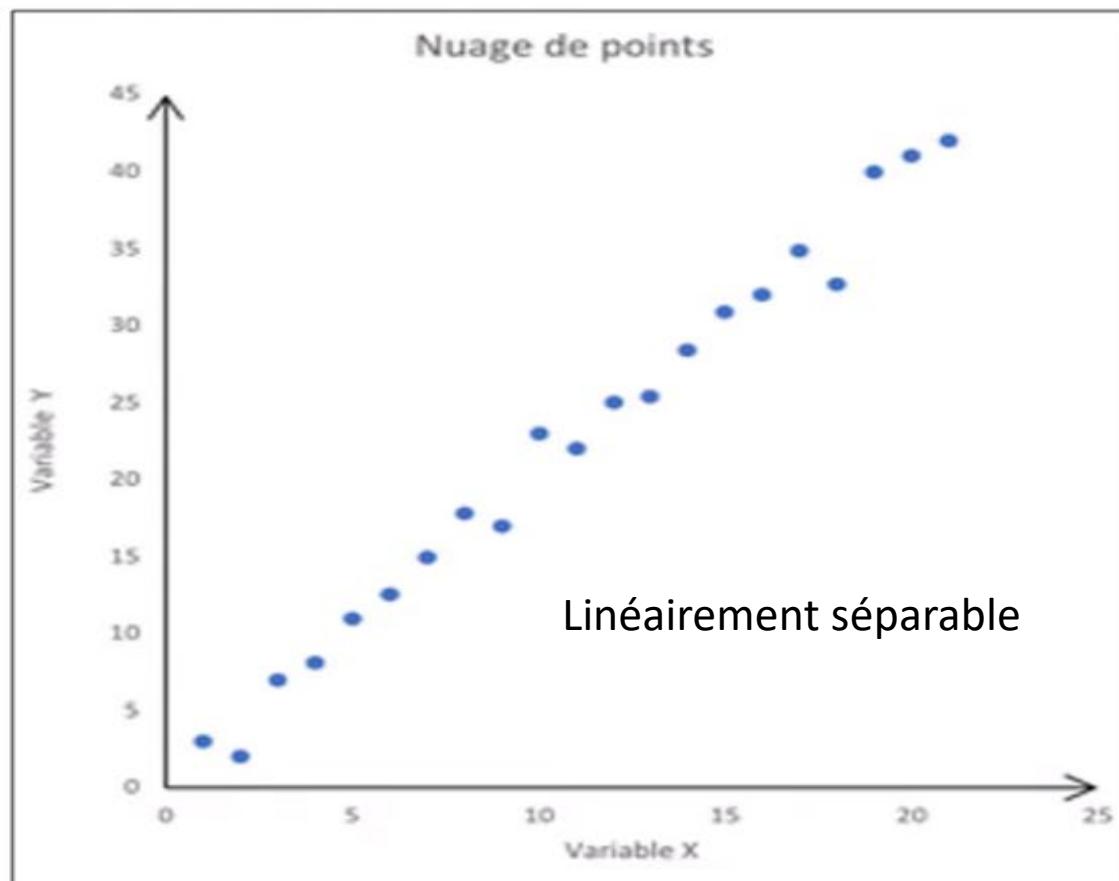
Objectif : de trouver une fonction dite de prédiction ou une fonction coût qui décrit la relation entre **X** et **Y**

La fonction recherchée est : **Y= f(X)** Avec **f(X)** une fonction linéaire.

- **Algorithmes de Machine learning**

- **Régression linéaire (RL) (Linear Regression (LR))**

- **Principe**



- **Algorithmes de Machine learning**

- **Régression linéaire (RL) (Linear Regression (LR))**

- **Modélisation**

Une seule variable explicative X



Régression simple

Plusieurs variables explicatives $X_j (j=1, \dots, n)$



Régression multiple

Un modèle de régression linéaire simple est sous la forme :

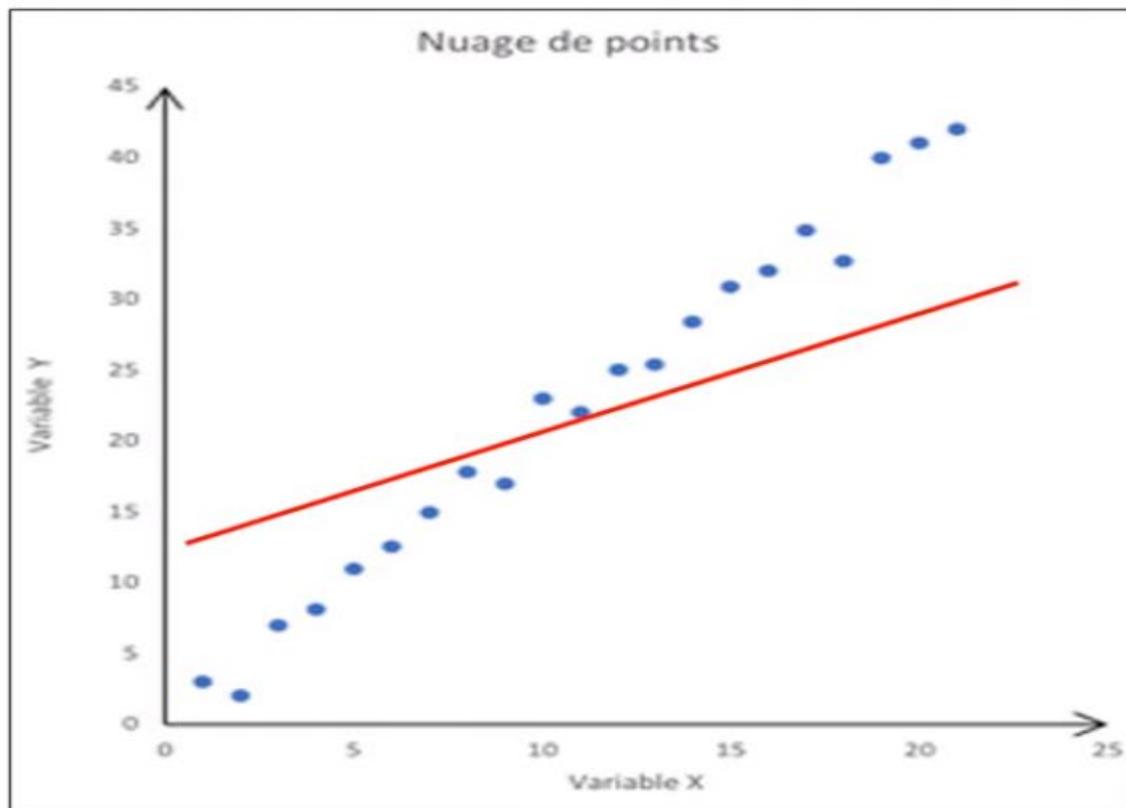
$$Y = aX + b + \varepsilon \text{ où } f(X) = aX + b$$

- **Algorithmes de Machine learning**

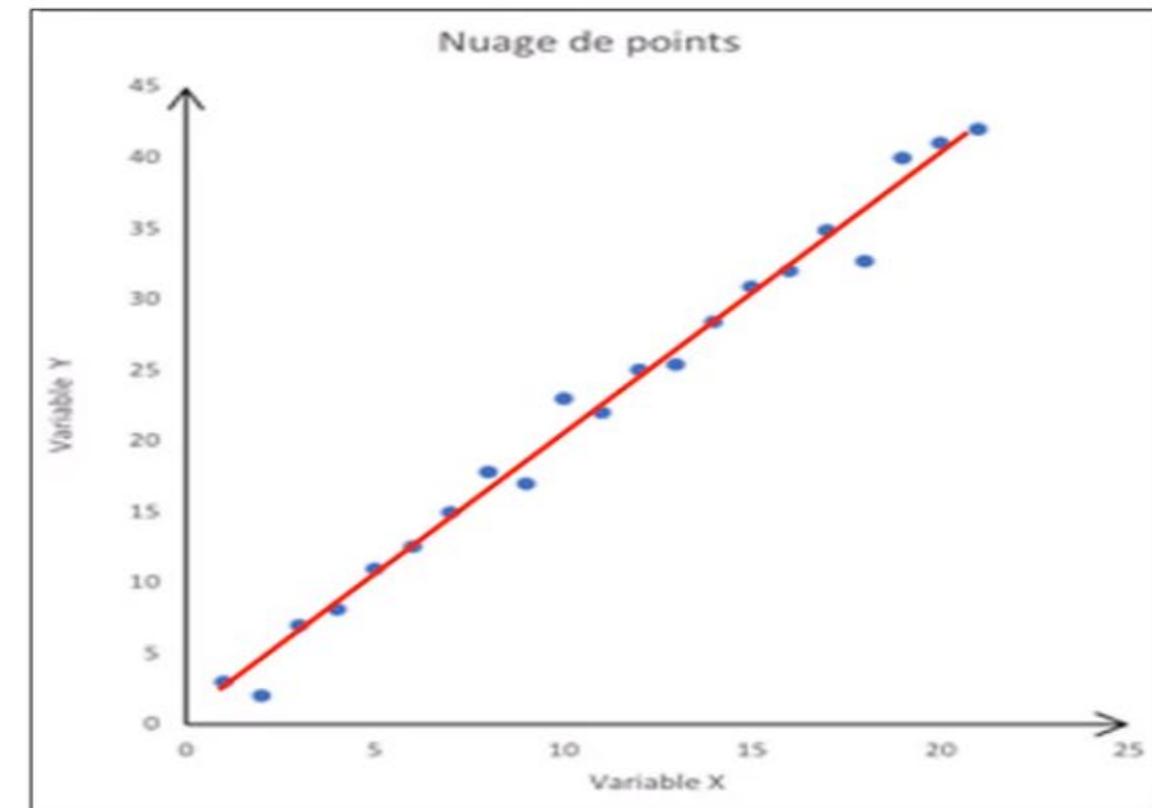
- **Régression linéaire (RL) (Linear Regression (LR))**

- **Modélisation**

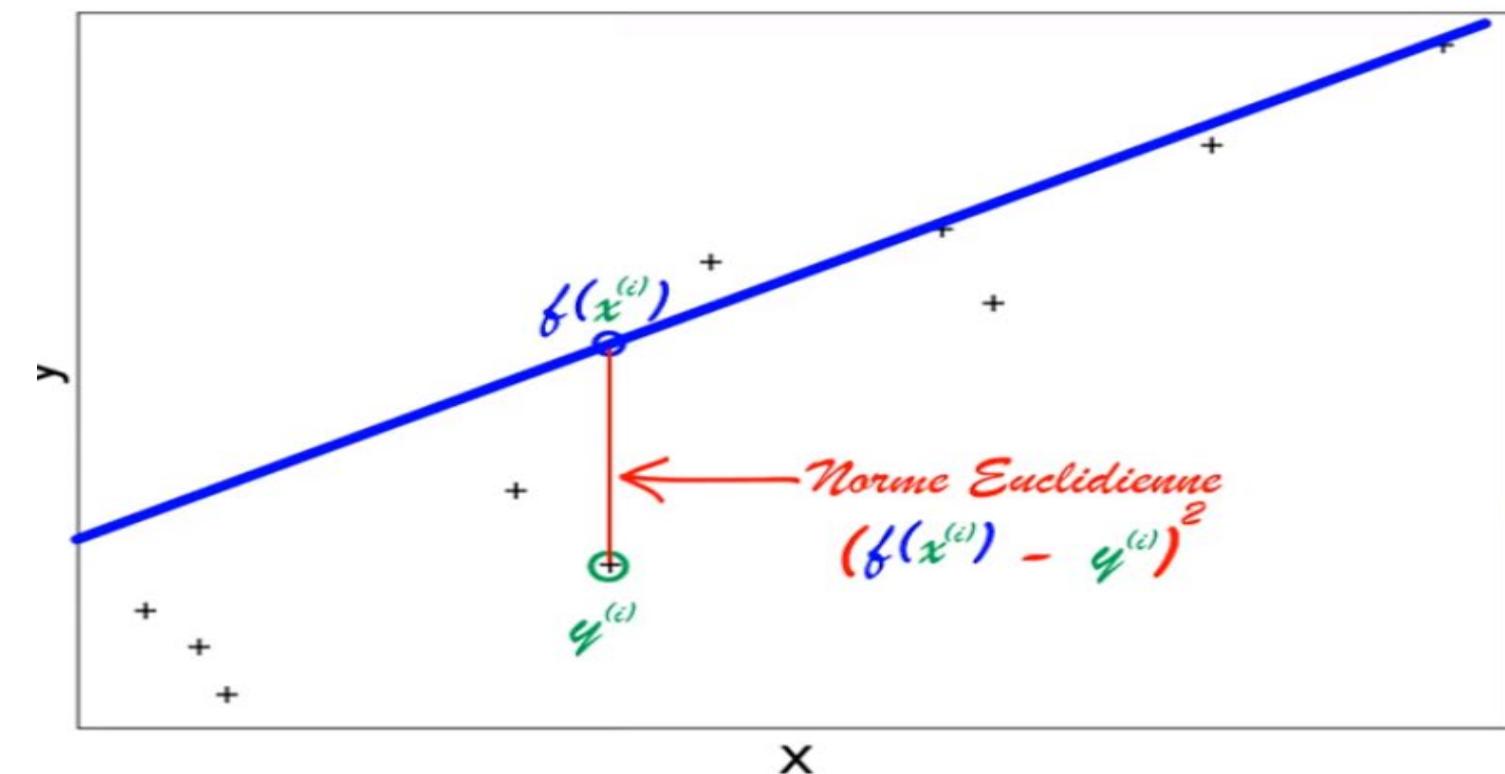
Modèle initial (paramètres aléatoires)



Modèle final (paramètres trouvés)



- Algorithmes de Machine learning
- Régression linéaire (RL) (Linear Regression (LR))
- Modélisation





600k Dhs

700k Dhs

Erreur = $f(x) - y$.

Une erreur de $700k - 600k = 100k$ dhs.

$800k - 900k = -100k$ dhs !!!!

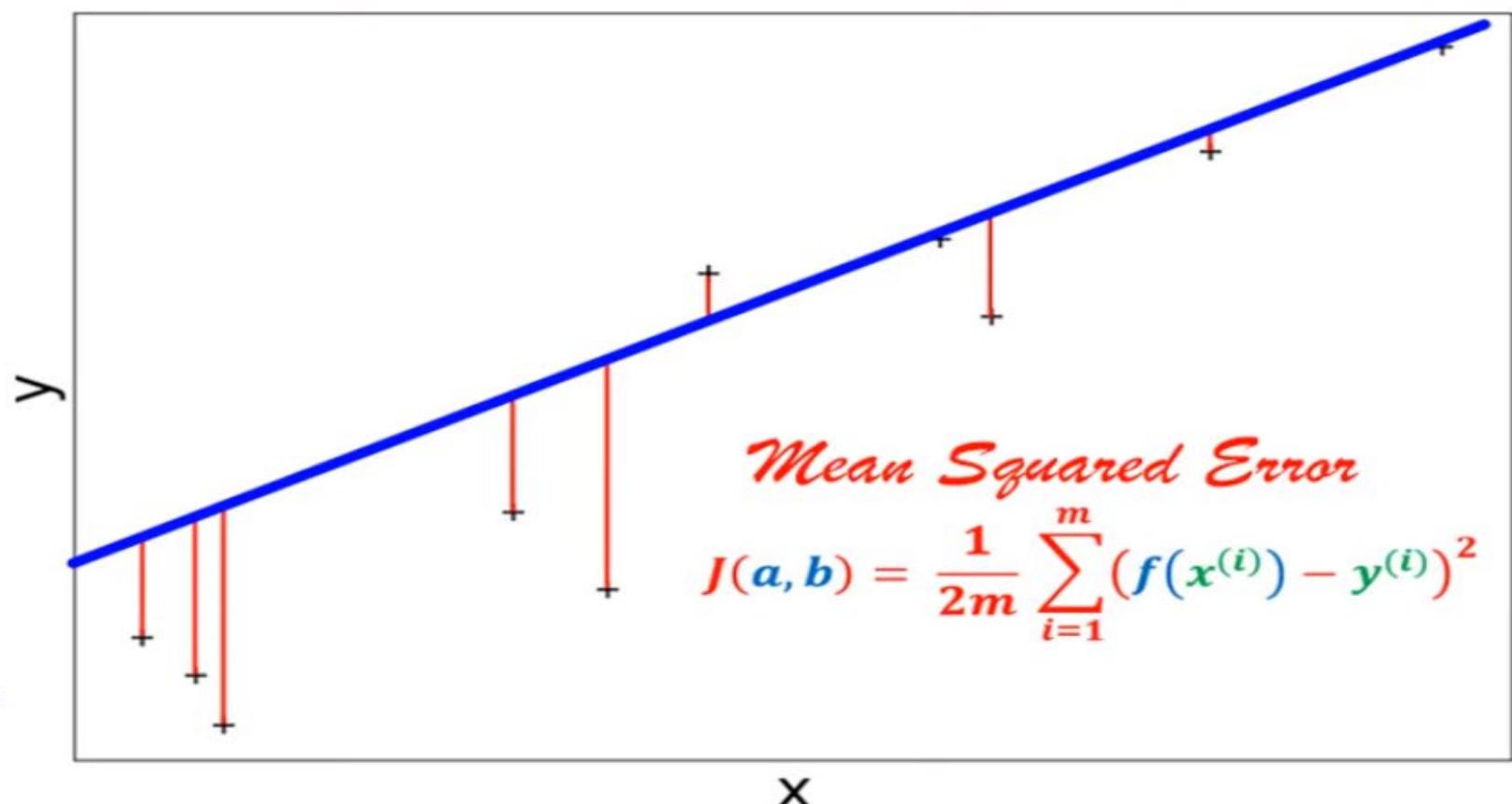
Erreur = $(f(x) - y)^2$.

- Algorithmes de Machine learning
- Régression linéaire (RL) (Linear Regression (LR))
- Modélisation

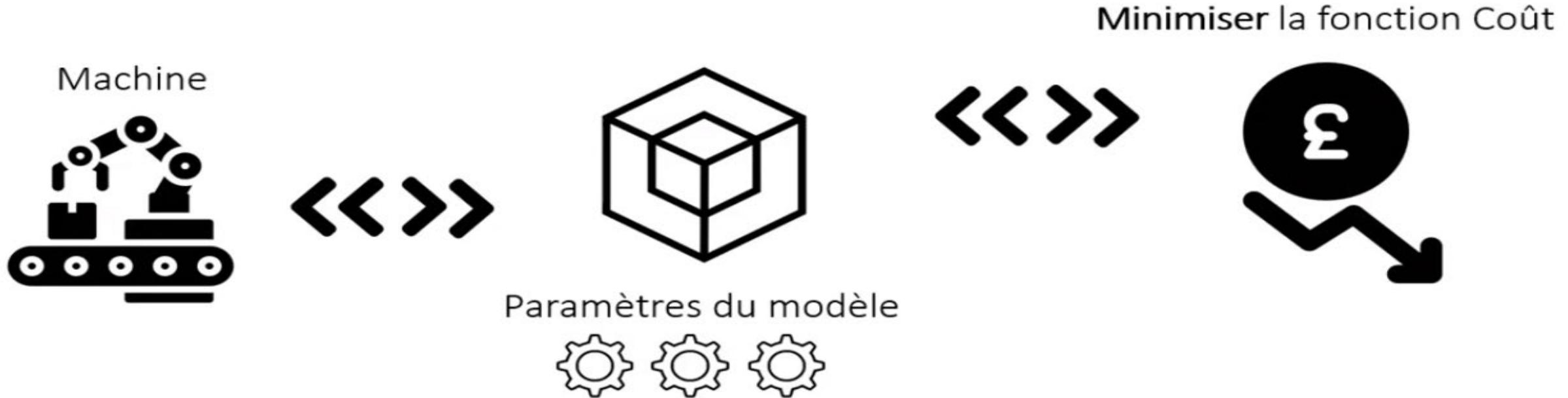
$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2$$

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})^2$$

Erreur Quadratique Moyenne



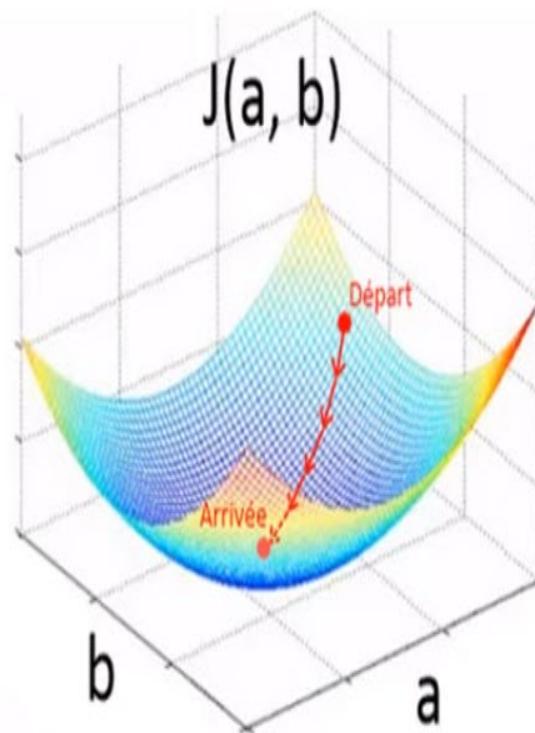
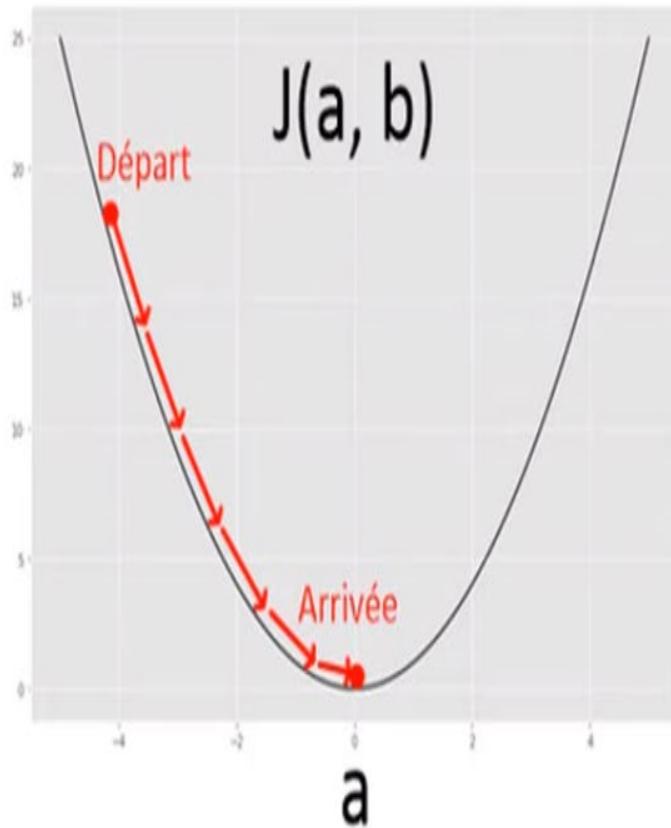
- **Algorithmes de Machine learning**
- **Régression linéaire (RL) (Linear Regression (LR))**
- **Modélisation (Descente de gradient)**



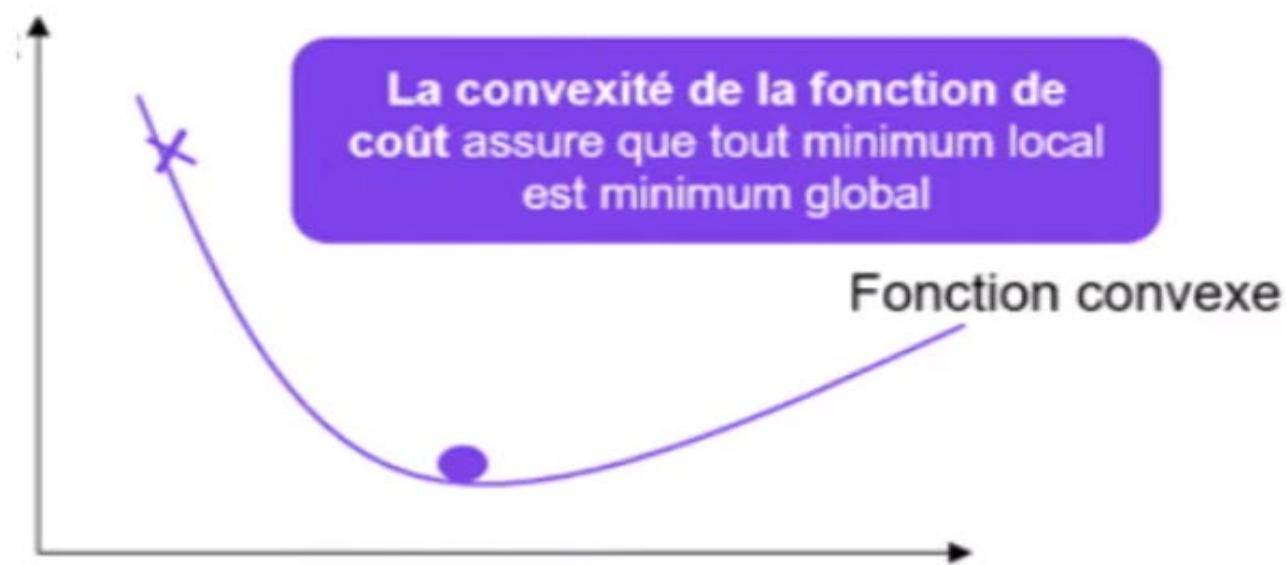
✓ **Méthode des moindres carrés**

✓ **Gradient Descent**

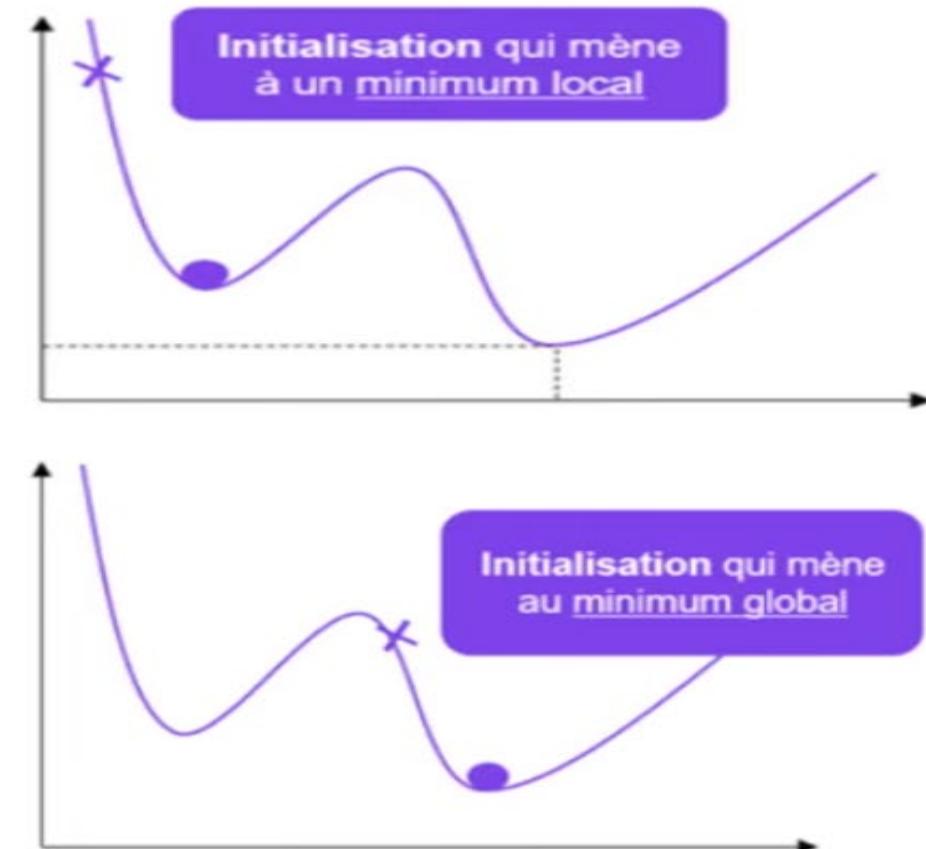
- **Algorithmes de Machine learning**
- **Régression linéaire (RL) (Linear Regression (LR))**
- **Modélisation (Descente de gradient)**



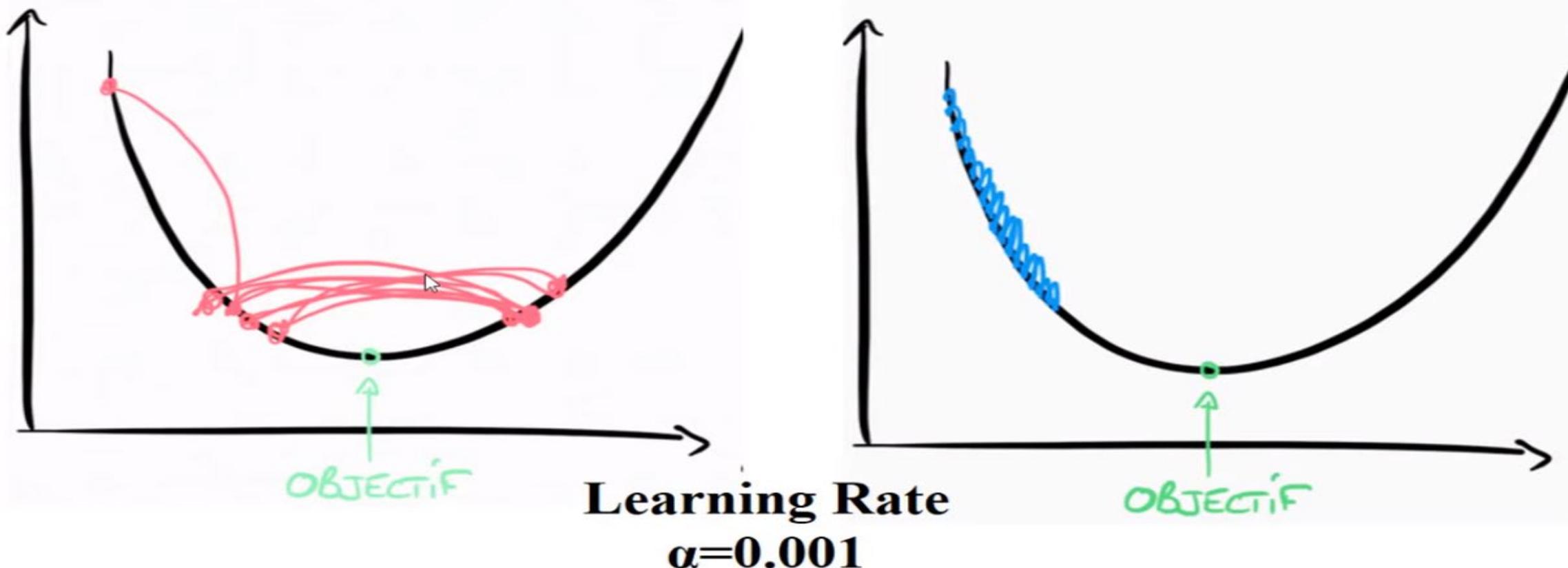
- Algorithmes de Machine learning
- Régression linéaire (RL) (Linear Regression (LR))
- Modélisation (Descente de gradient)



$$\begin{pmatrix} a_i \\ b_i \end{pmatrix} := \begin{pmatrix} a_i \\ b_i \end{pmatrix} - \alpha \frac{\partial}{\partial \beta_i} C(a, b)$$



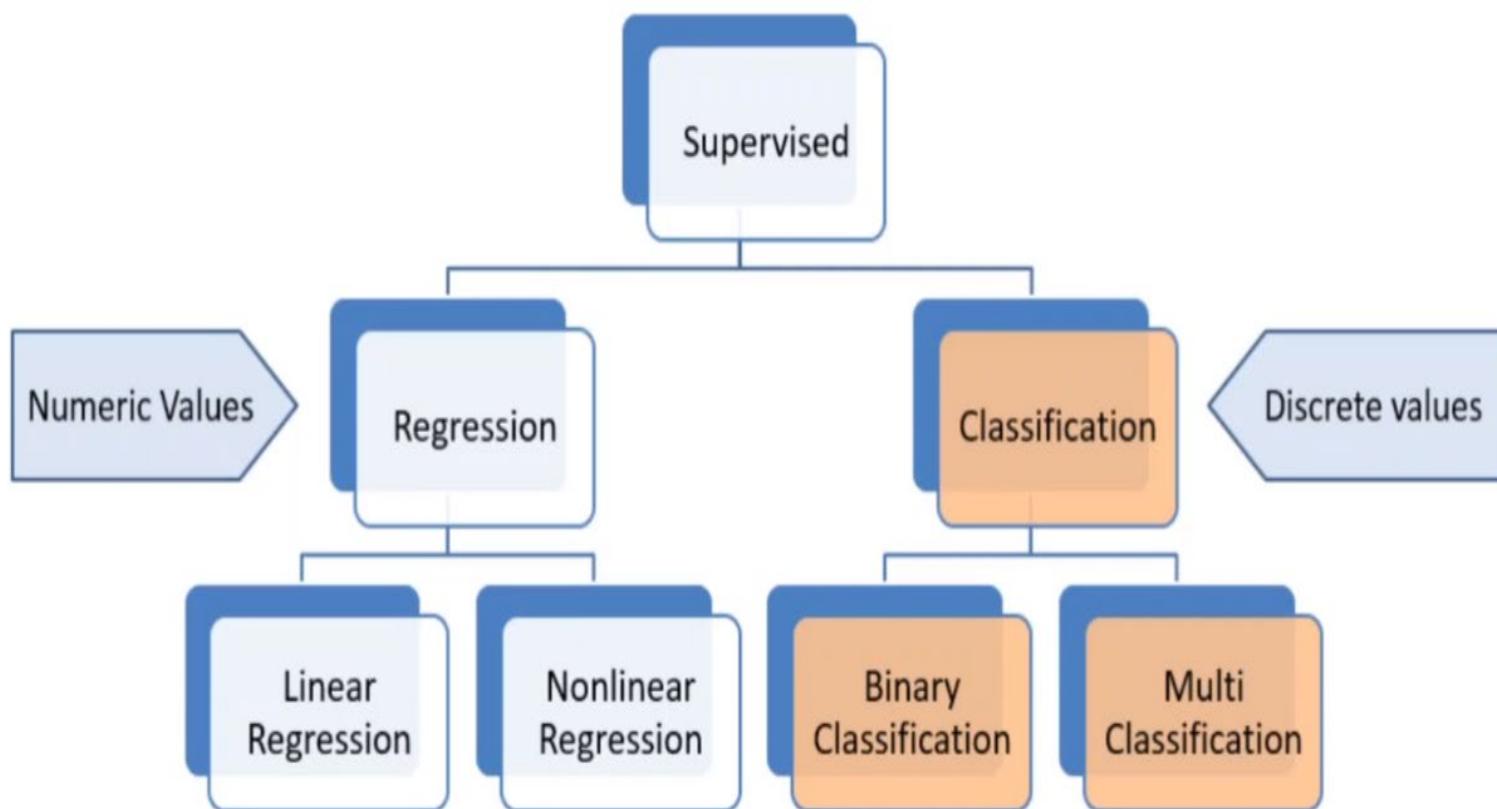
- Algorithmes de Machine learning
- Régression linéaire (RL) (Linear Regression (LR))
- Modélisation (Descente de gradient)



▪ Algorithmes de Machine learning

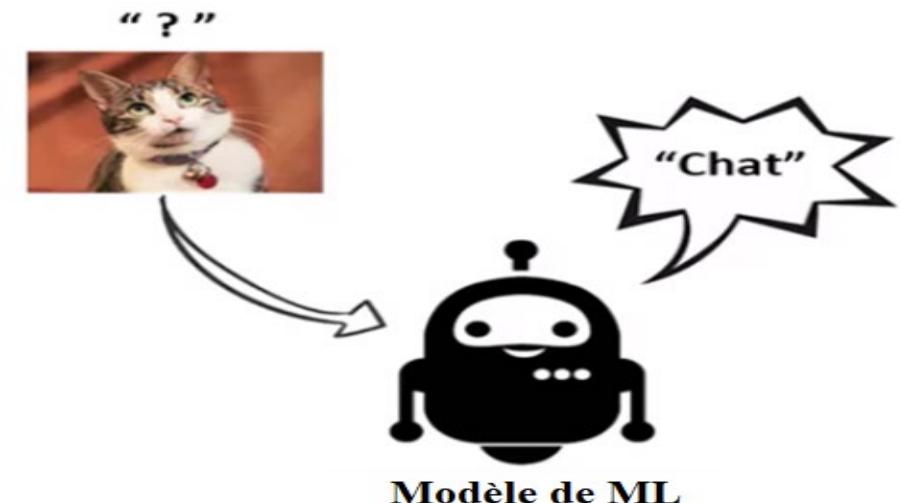
➤ La classification

Supervised Learning



La classification consiste à examiner les caractéristiques d'un objet et lui attribuer une classe pré-déterminée. Le rôle de la classe est joué par un attribut sélectionné dans l'ensemble de données déterminant un objet. En statistique, l'attribut est appelé la variable dépendante.

Exemple

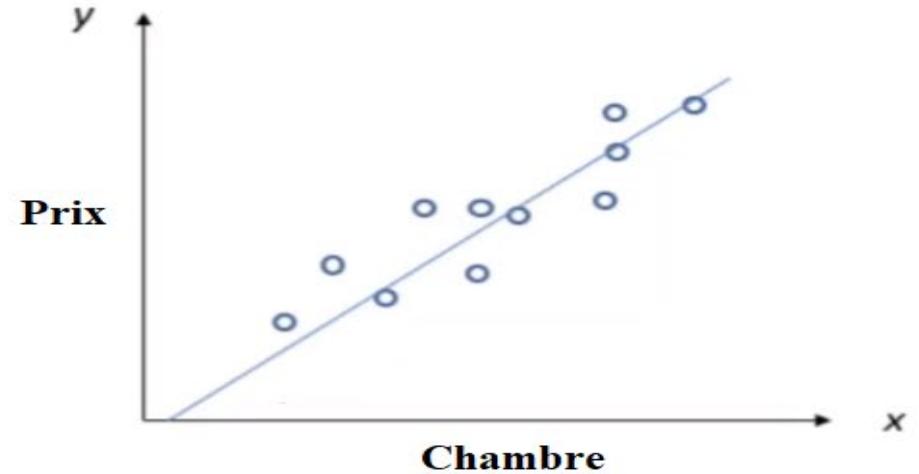


- Algorithme de Machine learning

- Régression Logistique-RL (Logistic Regression-LR)

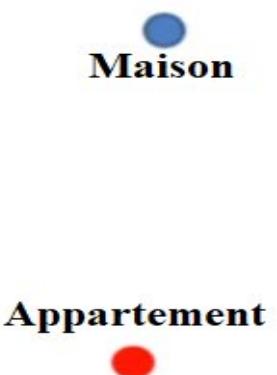
Chambre	Prix
3	50
5	70
4	100
6	150

Régression

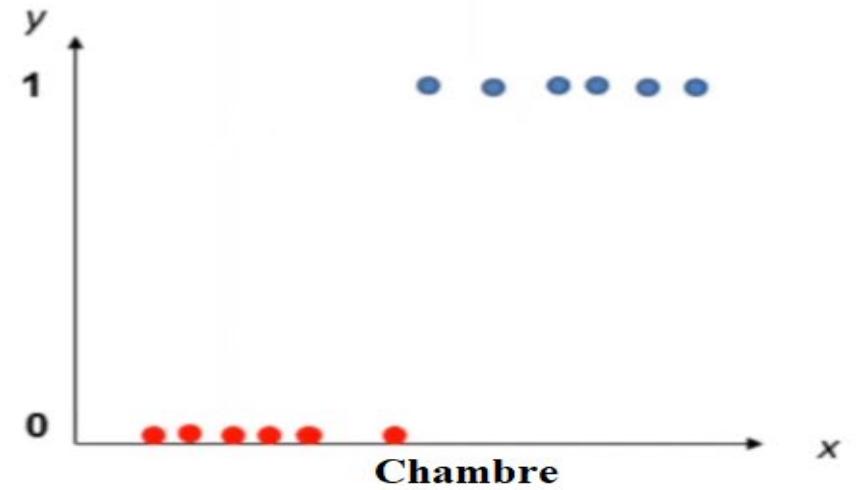


Chambre	Type
3	0
5	1
4	0
6	1

Classification



0 - Appartement
1 - Maison



■ Algorithmes de Machine learning

➤ Régression Logistique-RL (Logistic Regression-LR)

La régression logistique est un algorithme supervisé de classification. L'idée principale de la régression logistique est d'estimer la probabilité d'une réponse binaire basée sur une ou plusieurs variables indépendantes.

L'algorithme LR utilise également une équation linéaire avec des prédicteurs indépendants pour prédire une valeur. La valeur prédite peut se situer n'importe où entre l'infini négatif et l'infini positif.

Nous avons besoin que la sortie de l'algorithme soit une variable de classe, c'est-à-dire 0-non, 1-oui. Par conséquent, nous écrasons la sortie de l'équation linéaire dans une plage de [0,1]. Pour écraser la valeur prédite entre 0 et 1, nous utilisons la fonction sigmoïde.

Equation linéaire

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

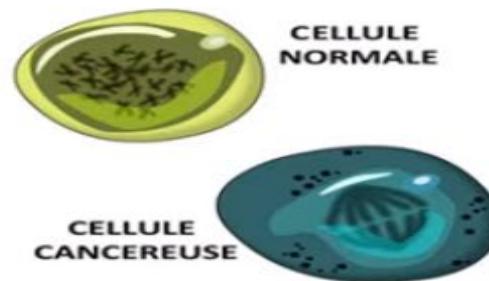
fonction sigmoïde

$$g(z) = \frac{1}{1 + e^{-(z)}}$$

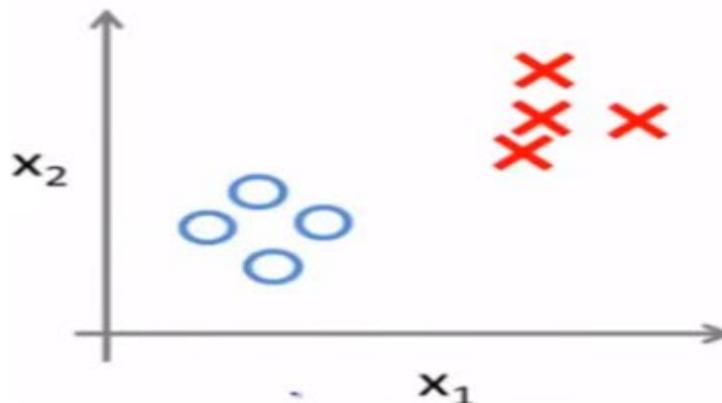
- Algorithmes de Machine learning

- Régression Logistique-RL (Logistic Regression-LR)

Classification binaire



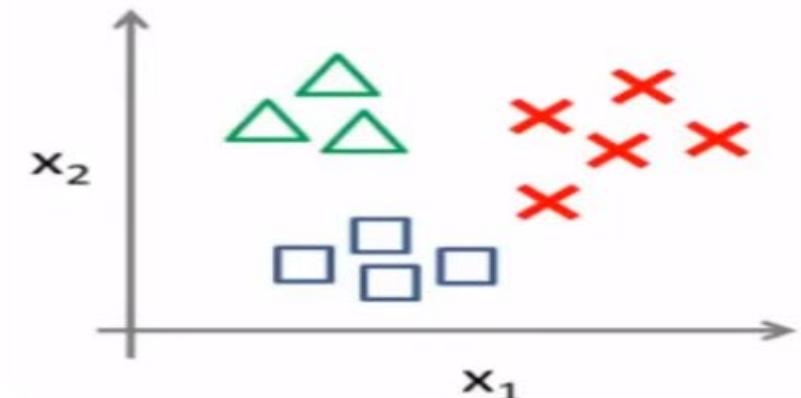
Binary classification:



Classification multi classes



Multi-class classification:

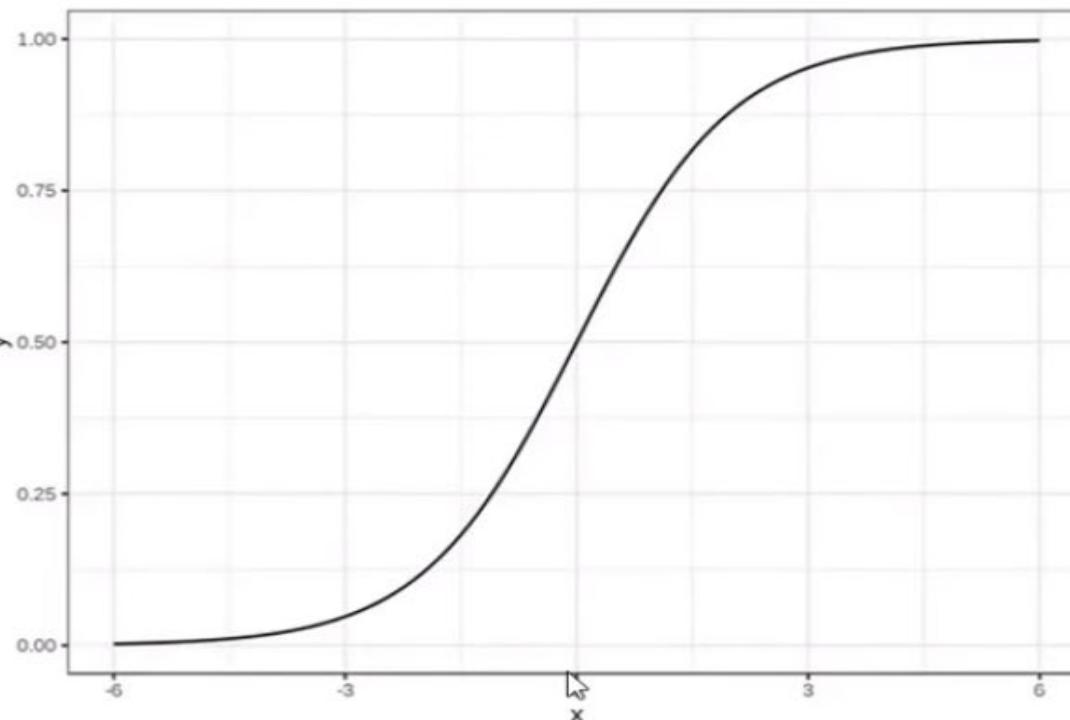


■ Algorithmes de Machine learning

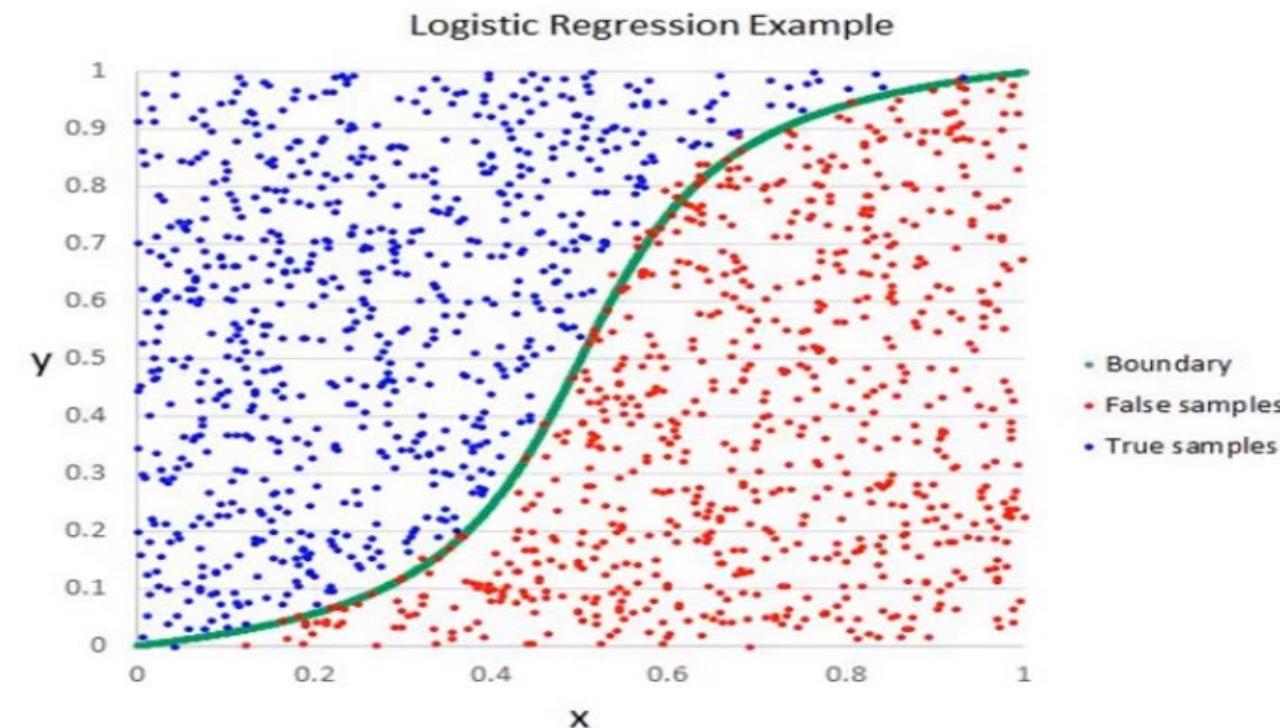
➤ Régression Logistique-RL (Logistic Regression-LR)

Un modèle de régression logistique permet de prédire la probabilité qu'un événement arrive (1) ou non (0) à partir d'optimisation des coefficients de régression.

La fonction qui remplit le mieux ces conditions est la fonction **sigmoïde**



$$Y = \begin{cases} 1 & \text{si } f(X) \geq \text{seuil} \\ 0 & \text{si } f(X) < \text{seuil} \end{cases}$$



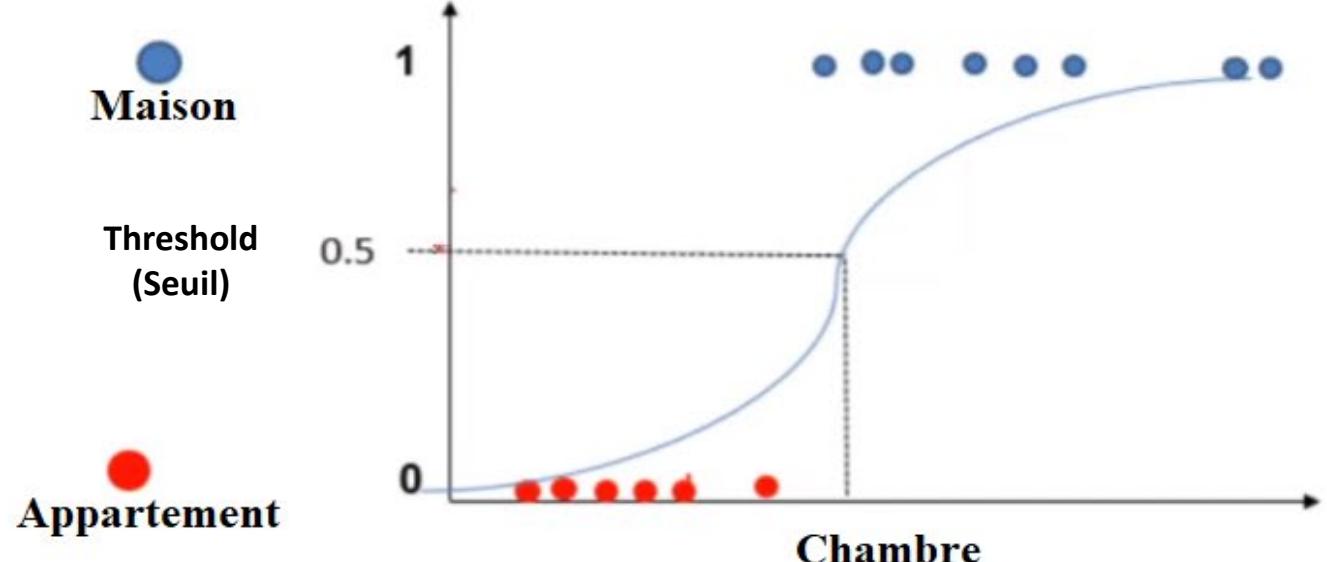
■ Algorithmes de Machine learning

➤ Régression Logistique-RL (Logistic Regression-LR)

Chambre	Type
3	0
5	1
4	0
6	1

Si $h(x) \geq 0.5$, $y = 1$

Si $h(x) < 0.5$, $y = 0$



Régression Logistique

$$0 \leq h(x) \leq 1$$

$$h(x) = ax + b$$

fonction sigmoïde

$$g(z) = \frac{1}{1 + e^{-(z)}}$$

$$z = h(x)$$

- **Algorithmes de Machine learning**

- **Régression Logistique-RL (Logistic Regression-LR)**

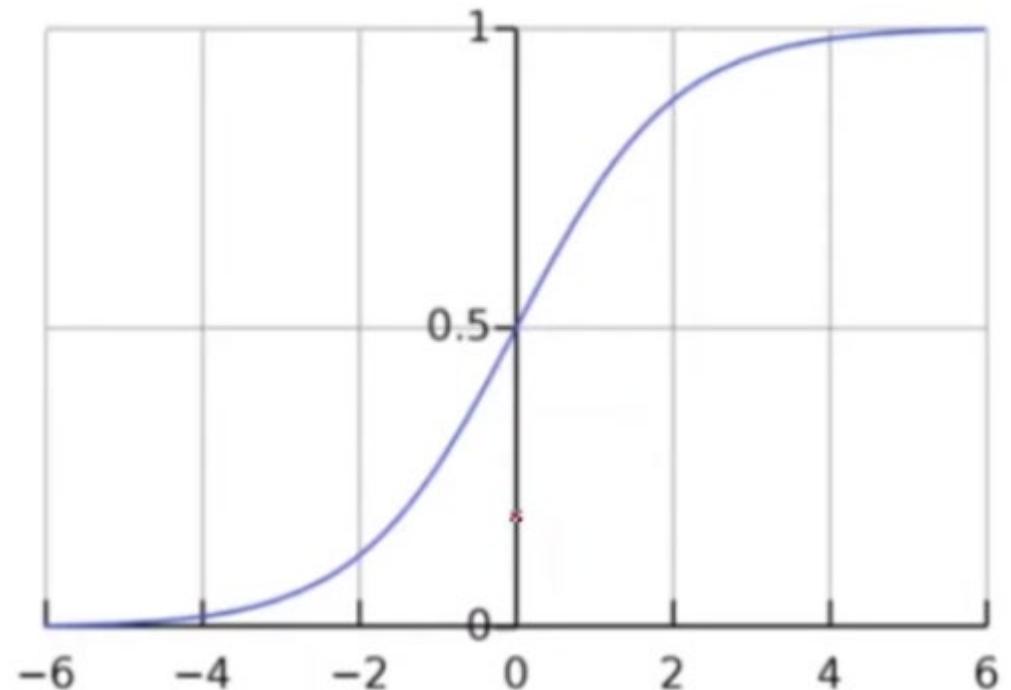
- Malgré son nom, la régression logistique est un modèle de classification, pas de régression

- Utilisé pour modéliser la probabilité de la classe de x via une fonction linéaire.

- Il peut être interprété comme la probabilité qu'une entrée X appartienne à la classe par défaut ($y=1$)

- La probabilité que x appartienne à une classe est comprise entre 0 et 1

- C'est une courbe en forme de S qui peut prendre n'importe quel nombre à valeur réelle et le mapper en une valeur comprise entre 0 et 1



■ Algorithmes de Machine learning

➤ Régression Logistique-RL (Logistic Regression-LR)

Exemple

$$y \in \{0, 1\}$$

0 : Bénin (négative)
1: Malin (positive)

x	y
3	1
2	1
1	1
5	0
4	0
6	0

Si $z \geq 0, g(z) \geq 0.5$

Si $z < 0, g(z) < 0.5$

$$h(x) = -2x + 6$$

$$z = h(x)$$

$$g(z) = \frac{1}{1 + e^{-(z)}}$$

x	z	$e^{(-z)}$	$1 + e^{(-z)}$	g(z)	y
3	0	1,000	2,000	0,500	1
2	2	0,135	1,135	0,881	1
1	4	0,018	1,018	0,982	1
5	-4	54,598	55,598	0,018	0
4	-2	7,389	8,389	0,119	0
6	-6	403,429	404,429	0,002	0

Si $g(z) \geq 0.5$
 $\rightarrow Y=1$

Si $g(z) < 0.5$
 $\rightarrow Y=0$



La probabilité examinée

Implémentation pratique avec Python et sklearn (TP - Régression logistique)

Prédire si un client achètera un produit en fonction de son **âge**, de son **revenu annuel** et de son **score de fidélité** (basé sur son historique d'achats).

ID_Client	Age	Revenu_annuel (€)	Score_fidelite	Achat (1=Oui,0=Non)
1	22	25000	30	0
2	25	32000	40	0
3	47	80000	75	1
4	52	90000	85	1
5	46	78000	70	1
6	35	50000	50	0
7	28	40000	45	0
8	42	65000	65	1
9	39	62000	60	1
10	30	42000	38	0

- **Algorithmes de Machine learning**

- **Arbre de décision (Decision Tree-DT)**

Définition Un arbre de décision (DT) est un modèle de ML qui divise les données en sous-ensembles en se basant sur des critères comme le gain et l'entropie

But: d'identifier les classes auxquelles appartiennent des objets à partir de certains traits descriptifs.

Citons par exemple :

- L'aide au diagnostic médical : à partir de la liste des symptômes d'un malade (sa description) la procédure de classification indique sa maladie probable (sa classe).
- Accord d'un prêt bancaire : à partir de la situation d'un client (sa description) la procédure de classification donne la réponse à la demande de prêt : oui / non (sa classe).

- **Algorithmes de Machine learning**

- **Arbre de décision (Decision Tree-DT)**

Ces quelques exemples précédents font déjà apparaître trois objets essentiels :

1. La population : les malades ou les clients.

2. Les descriptions : les symptômes ou les revenus, âge, statut marital d'un client.

3. Les classes : les maladies ou les réponses à la demande de prêt (oui / non).

Remarque: la donnée d'une description avec la classe correspondante est un exemple.

Exemples:

(30 ans, marié(oui), 8000 DH, oui (classe))

(50 ans, marié(oui), 3000 DH, Non (classe))

- **Algorithmes de Machine learning**
- Arbre de décision (Decision Tree-DT)

Fonctionnement:

Un arbre de décision accepte en entrée une situation ou un objet décrit par un ensemble d'attributs et retourne une décision (valeur de sortie prédite).

Un arbre de décision atteint une décision en exécutant une séquence de tests.

Exemple introductif

Des objets (des patients),
des attributs (Température et Gorge irritée),
des classes (malade ou sain).

L'objectif est de construire un arbre de décision qui classe les patients en utilisant un algorithme de classification et un ensemble d'exemples bien choisis.

- **Algorithmes de Machine learning**
 - Arbre de décision (Decision Tree-DT)
- Fonctionnement (Algorithme)**

1.

Déterminer la meilleure caractéristique dans l'ensemble de données .

2.

Diviser les données en sous-ensembles contenant les valeurs possibles de la meilleure caractéristique.

3.

Générer de manière récursive de nouveaux arbres de décision en utilisant les sous-ensembles de données créés.

4.

Lorsqu'on ne peut plus classifier les données, on s'arrête.

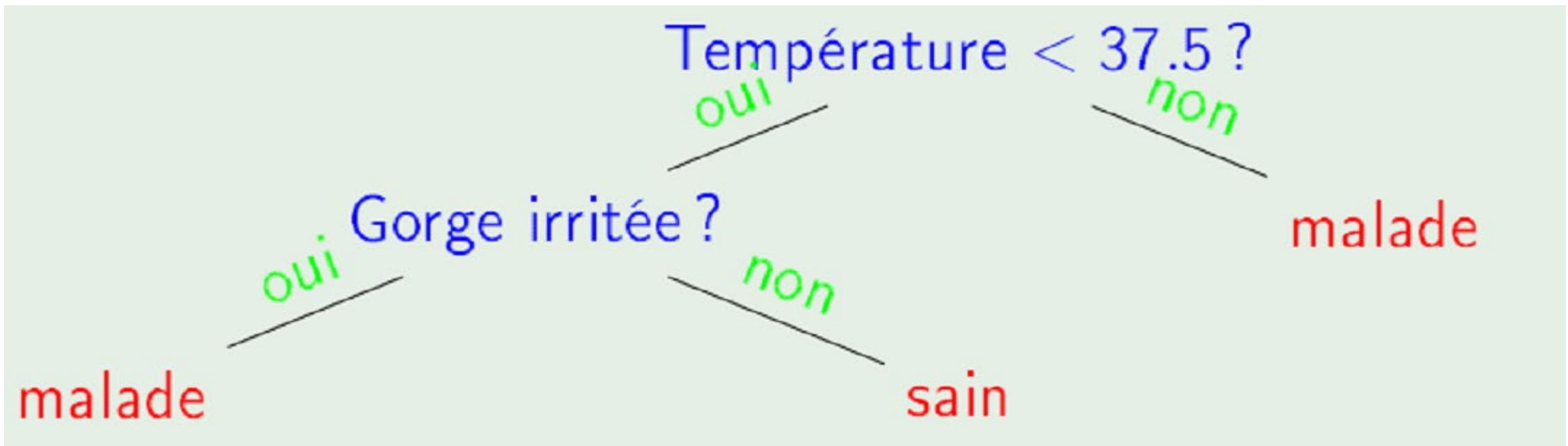
- Algorithmes de Machine learning
- Arbre de décision (Decision Tree-DT)

Fonctionnement:

Noeuds: Attributs

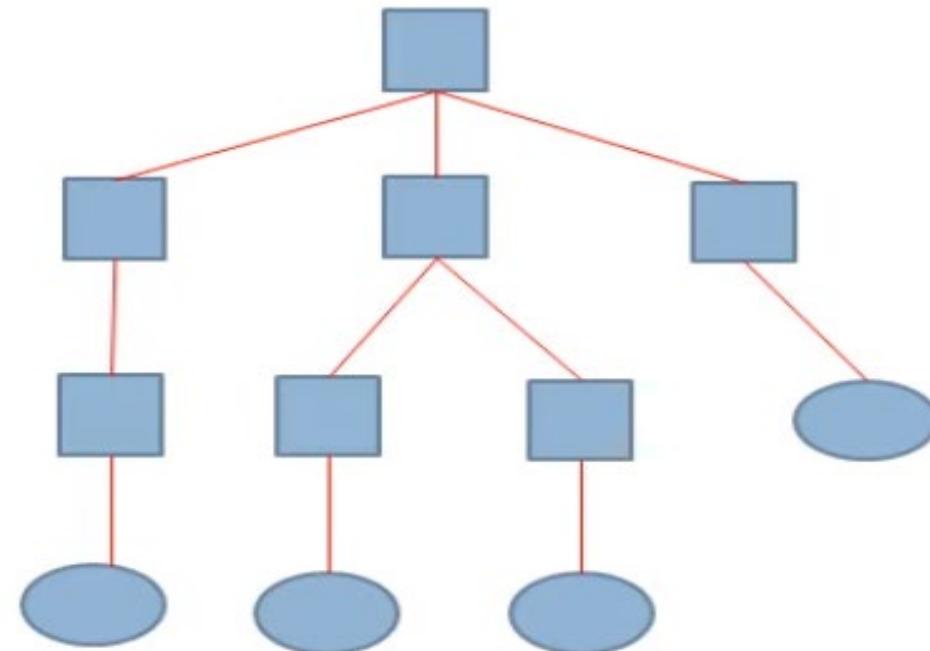
Branches: Valeurs prises par les Attributs

Feuilles: Décisions (Classes)



Chaque nœud interne correspond à un test sur un attribut, et les branches partant du nœud sont annotées avec les valeurs possibles du test (de l'attribut)

- Algorithmes de Machine learning
- Arbre de décision (Decision Tree-DT)



Les attributs



Les valeurs de l'attribut

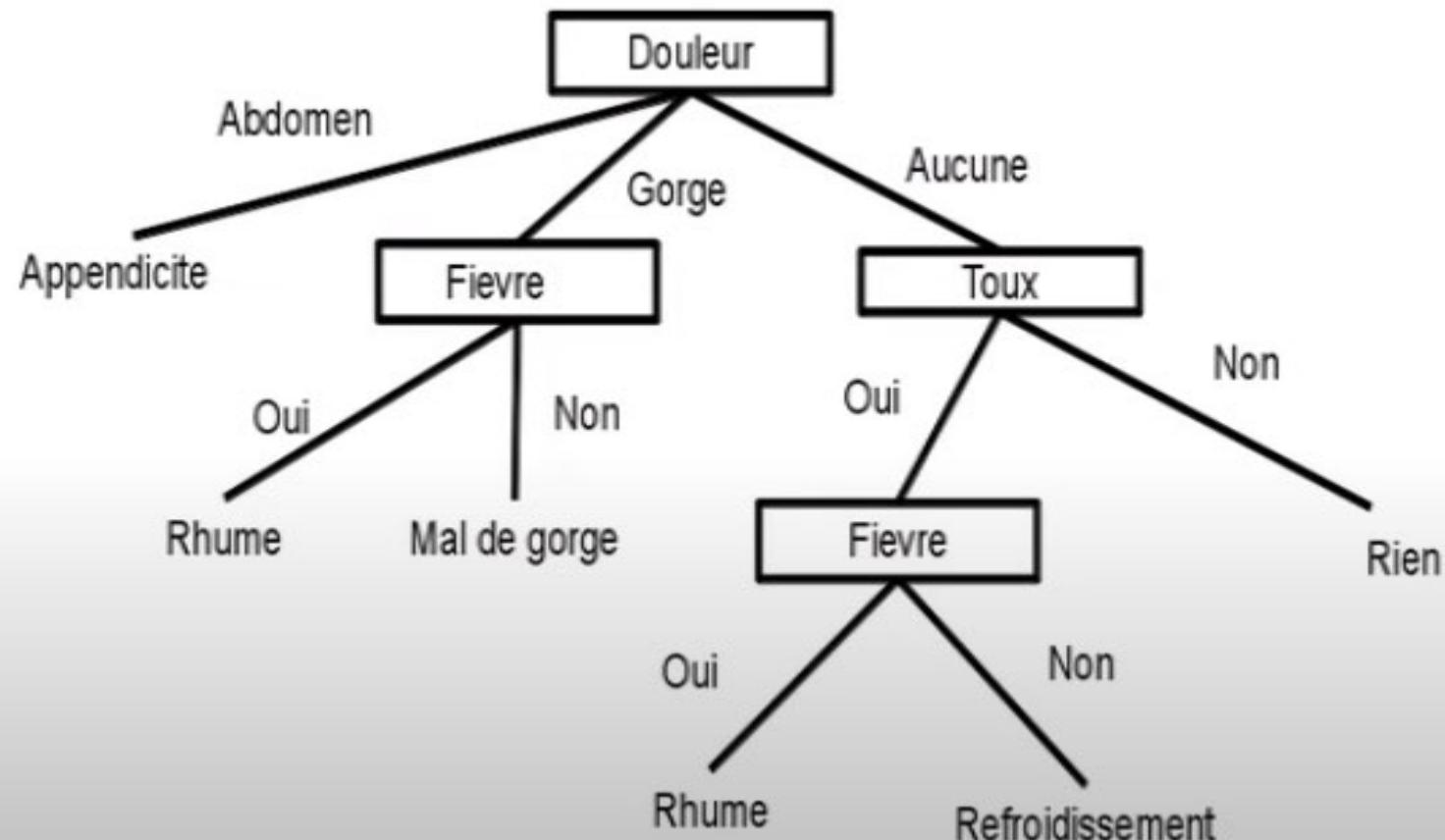


Classe = feuille = noeud pur

- Algorithmes de Machine learning
- Arbre de décision (Decision Tree-DT)

Exemple

Fievre	Douleur	Toux	Maladie
oui	Abdomen	non	Appendicite
non	Abdomen	oui	Appendicite
oui	gorge	non	Rhume
oui	gorge	oui	Rhume
non	gorge	oui	mal de gorge
oui	non	non	aucune
oui	non	oui	Rhume
non	non	oui	refroidissement
non	non	non	aucune



- **Algorithmes de Machine learning**

- **Arbre de décision (Decision Tree-DT)**

Sélectionner la meilleure caractéristique (ID3)

- Etant donnée un ensemble de classes K ,
- Entropie: $E(S) = - \sum_{i=1}^K p_i \log_2(p_i)$
- Où (en divisant le nombre des échantillons d'une certaine classe sur le nombre de tous les échantillons dans les données d'entraînement):
 - $p_i = \frac{|i|}{|S|}$
 - Remarques:
 - $0 \leq E(S) \leq 1$
 - $\log_2(x) = \frac{\log(x)}{\log(2)}$

S : Ensemble de données

$$p_i = \frac{\text{cardinalité de } i}{\text{cardinalité de } S}$$

- **Algorithmes de Machine learning**

- **Arbre de décision (Decision Tree-DT)**

- Sélectionner la meilleure caractéristique (ID3)

Etant donné un vecteur d'attributs, on peut diviser l'ensemble de données S en plusieurs sous-ensemble.

Le gain d'information est mesuré en se basant sur la différence entre l'entropie original de S et celle après sa division (S_i) en se basant sur un attribut.

- **Le gain d'information:**

- $Gain(S, attribut) = E(S) - \sum_{i=1}^K p_i E(S_i)$

- **La caractéristiques ayant plus de gain d'information est celle sélectionnée comme meilleure.**

- **La valeur avec entropie nulle est considérée comme feuille de l'arbre.**

■ Algorithmes de Machine learning

➤ Arbre de décision (Decision Tree-DT)

But : déterminer si le client est intéressé à acheter un certain produit ou non à partir de plusieurs caractéristiques?

Exercice 1

ID	Genre	Âge	État civil	Revenu	Achat
1	Homme	18 – 35	Marié	Moyen	Non
2	Homme	< 18	Célibataire	Faible	Non
3	Homme	> 35	Marié	Élevé	Oui
4	Femme	< 18	Célibataire	Moyen	Non
5	Homme	18 – 35	Célibataire	Moyen	Non
6	Femme	18 – 35	Célibataire	Élevé	Oui
7	Femme	18 – 35	Marié	Faible	Non
8	Homme	18 – 35	Marié	Élevé	Oui
9	Homme	> 35	Célibataire	Faible	Oui
10	Femme	< 18	Célibataire	Moyen	Non
11	Femme	> 35	Célibataire	Moyen	Oui
12	Femme	> 35	Marié	Élevé	Oui
13	Homme	18 – 35	Célibataire	Faible	Non
14	Femme	18 – 35	Marié	Moyen	Oui

- Construisez l'arbre de décision résultant de ces exemples en utilisant l'algorithme ID3.
- Exprimez la classe des exemples positifs sous la forme d'un prédictat logique.
- Décrivez comment un nouvel exemple est classifié dans un arbre de décision entraîné.

■ Algorithmes de Machine learning

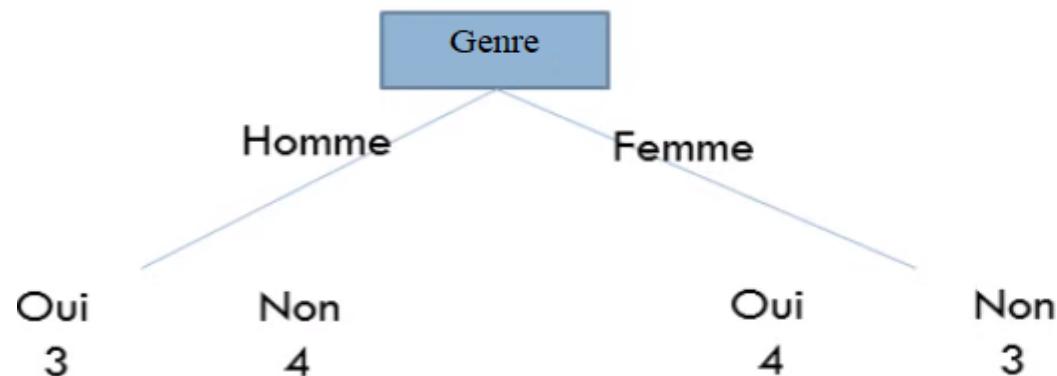
➤ Arbre de décision (Decision Tree-DT)

Au total on a 14 attributs: 7 Oui et 7 Non:

Exercice 1(solution)

$$E(S) = - \sum_{i=1}^K p_i \log_2(p_i)$$

$$E(S) = - \left[\frac{7}{14} \log_2 \frac{7}{14} \right] - \left[\frac{7}{14} \log_2 \frac{7}{14} \right] = 1$$



$$E(Homme) = - \sum p_i \log_2 p_i$$

$$E(Homme) = - \left[\frac{3}{7} \log_2 \frac{3}{7} \right] - \left[\frac{4}{7} \log_2 \frac{4}{7} \right] = 0,985$$

$$E(Femme) = - \left[\frac{4}{7} \log_2 \frac{4}{7} \right] - \left[\frac{3}{7} \log_2 \frac{3}{7} \right] = 0,985$$

$$Gain(S, \text{Genre}) = E(S) - \sum_{i=1}^K p_i E(S_i) = 1 - \left[\frac{7}{14} \times 0,985 \right] - \left[\frac{7}{14} \times 0,985 \right] = 0,015$$

- Algorithmes de Machine learning

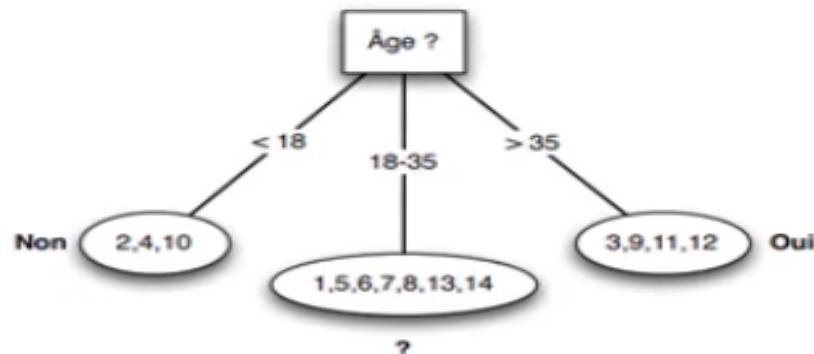
- Arbre de décision (Decision Tree-DT)

- Nous avons les gains d'entropie suivants:

Exercice 1(solution)

Attribut	Gain
Genre	$1 - [(7/14)(0.985) + (7/14)(0.985)] = 0.015$
Âge	$1 - [(3/14)(0) + (7/14)(0.985) + (4/14)(0)] = 0.507$
État civil	$1 - [(8/14)(0.954) + (6/14)(0.918)] = 0.061$
Revenu	$1 - [(4/14)(0.811) + (6/14)(0.918) + (4/14)(0)] = 0.375$

- Le plus gros gain provient de l'attribut Age et on choisit celui-ci pour séparer les exemples :



- Les nœuds Age < 18 et Age > 35 sont purs et nous n'avons pas besoin de les subdiviser. Par ailleurs, pour le nœud Age = 18 – 35, nous testons le gain d'entropie pour les attributs restants :

- **Algorithmes de Machine learning**
- Arbre de décision (Decision Tree-DT)

Exercice 1(solution)

ID	Genre	Age	Etat civil	Revenu	Achat
1	Homme	18-35	Marié	Moyen	Non
5	Homme	18-35	Célibataire	Moyen	Non
6	Femme	18-35	Célibataire	Elevé	Oui
7	Femme	18-35	Marié	Faible	Non
8	Homme	18-35	Marié	Elevé	Oui
13	Homme	18-35	Célibataire	Faible	Non
14	Femme	18-35	Marié	Moyen	Oui

■ Algorithmes de Machine learning

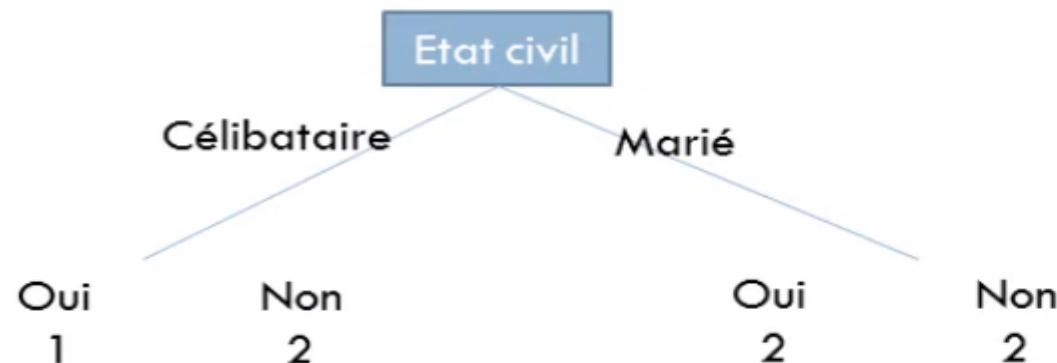
➤ Arbre de décision (Decision Tree-DT)

Au total on a 7 attributs: 3 Oui et 4 Non:

Exercice 1(solution)

$$E(S) = - \sum_{i=1}^K p_i \log_2(p_i)$$

$$E(S) = - \left[\frac{3}{7} \log_2 \frac{3}{7} \right] - \left[\frac{4}{7} \log_2 \frac{4}{7} \right] = 0,985$$



$$E(\text{Célibataire}) = - \sum p_i \log_2 p_i$$

$$E(\text{célibataire}) = - \left[\frac{1}{3} \log_2 \frac{1}{3} \right] - \left[\frac{2}{3} \log_2 \frac{2}{3} \right] = 0,918$$

$$E(\text{Marié}) = - \left[\frac{2}{4} \log_2 \frac{2}{4} \right] - \left[\frac{2}{4} \log_2 \frac{2}{4} \right] = 1$$

$$\text{Gain}(S_{age=18-35}, \text{etat civil}) = E(S) - \sum_{i=1}^K p_i E(S_i) = 0,985 - \left[\frac{3}{7} \times 0,918 \right] - \left[\frac{4}{7} \times 1 \right] = 0,020$$

- Algorithmes de Machine learning

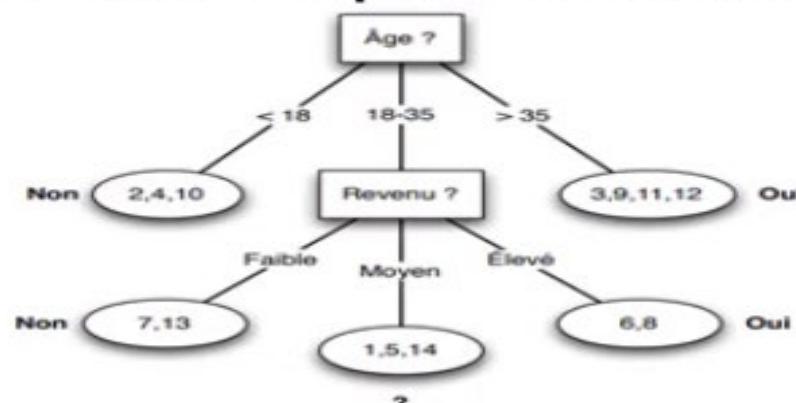
- Arbre de décision (Decision Tree-DT)

- Le gain d'entropie pour les attributs restants :

Exercice 1(solution)

Attribut	Gain
Genre	$0.985 - [(4/7)(0.811) + (3/7)(0.918)] = 0.128$
État civil	$0.985 - [(3/7)(0.918) + (4/7)(1)] = 0.020$
Revenu	$0.985 - [(2/7)(0) + (3/7)(0.918) + (2/7)(0)] = 0.592$

- On choisit donc l'attribut Revenu pour subdiviser les exemples :



- Les nœuds Revenu = Faible et Revenu = Elevé sont purs, il ne reste que le nœud Revenu = Moyen à subdiviser :

- Algorithmes de Machine learning
- Arbre de décision (Decision Tree-DT)

Exercice 1(solution)

ID	Genre	Age	Etat civil	Revenu	Achat
1	Homme	18-35	Marié	Moyen	Non
5	Homme	18-35	Célibataire	Moyen	Non
14	Femme	18-35	Marié	Moyen	Oui

Attribut	Gain
Genre	$0.918 - [(2/3)(0) + (1/3)(0)] = 0.918$
Etat civil	$0.918 - [(1/3)(0) + (2/3)(1)] = 0.252$

- L'attribut **Genre** donne le meilleur gain d'entropie et on choisit celui-ci pour séparer les exemples.

- Algorithmes de Machine learning
- Arbre de décision (Decision Tree-DT)

Exercice 1(solution)

Arbre de décision final



- Pour exprimer la classe des acheteurs potentiels, on identifie tous les chemins depuis la racine de l'arbre jusqu'à une feuille de l'arbre contenant des exemples positifs. La classe des exemples positifs s'exprime ensuite comme une disjonction (OU - V) de conjonctions (ET - \wedge) pour chaque chemin :
- $AchatOui(x) \Leftarrow (\text{Age}(x) > 35) \vee (\text{Age}(x) \in [18 - 35] \wedge \text{Revenu}(x) = \text{Élevé}) \vee (\text{Age}(x) \in [18 - 35] \wedge \text{Revenu}(x) = \text{Moyen} \wedge \text{Genre} = \text{Femme})$.

- Algorithme de Machine learning
- Arbre de décision (Decision Tree-DT)

Exercice 1(solution)

□ Pour classifier un nouvel exemple, on traverse l'arbre depuis la racine jusqu'à une feuille. Pour chaque nœud interne rencontré, on emprunte la branche correspondant au résultat du test de ce nœud. Une fois dans la feuille, on assigne l'exemple à la classe la plus fréquente des exemples d'entraînement contenus dans cette feuille.

- Algorithmes de Machine learning

- Arbre de décision (Decision Tree-DT)

- Construire l'arbre de décision à partir des données suivantes.

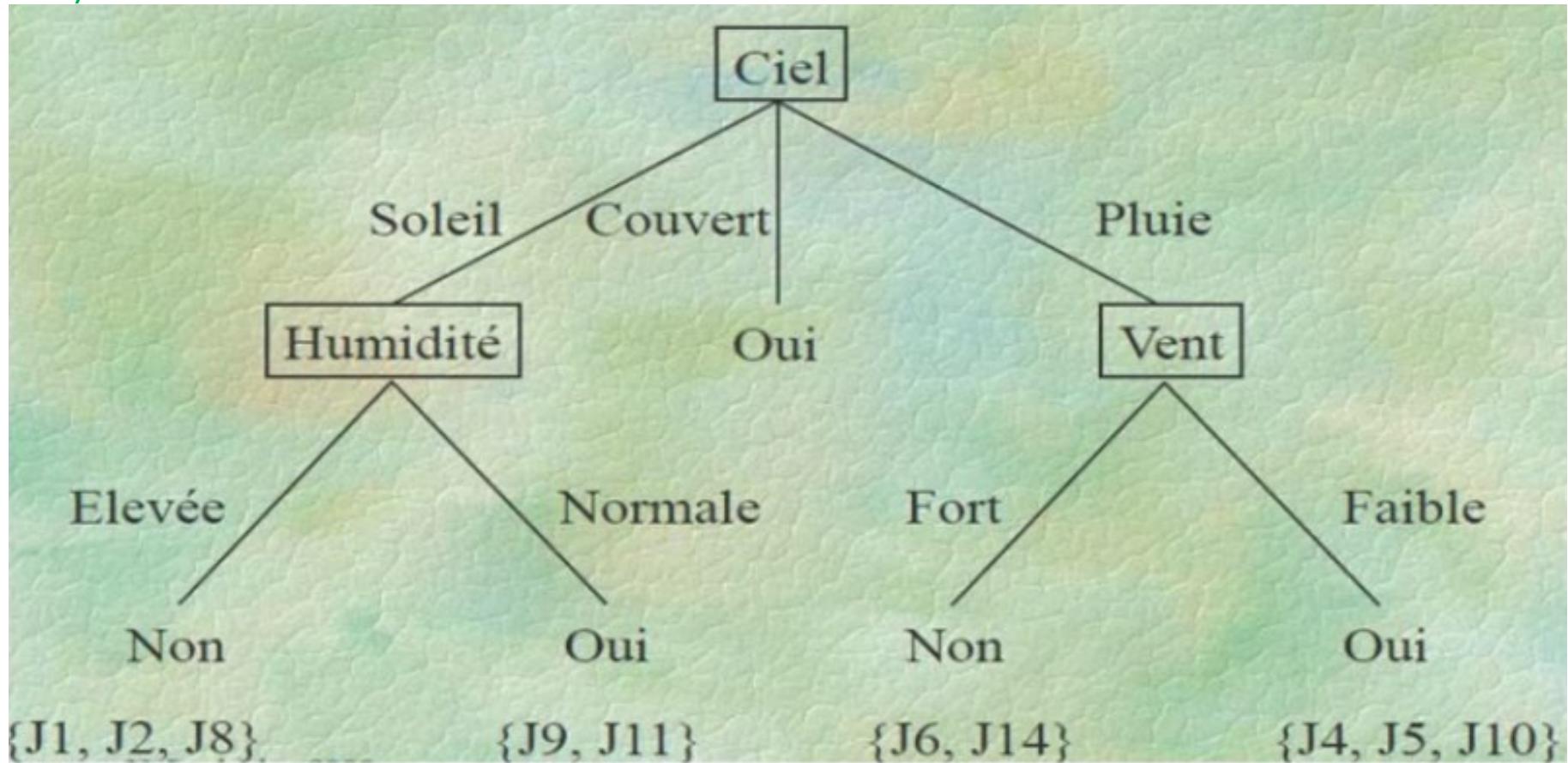
Exercice 2

- Exprimez la classe des exemples positifs sous la forme d'un prédictat logique.

Jour	Ciel	Température	Humidité	Vent	Jouer
J1	Soleil	Chaud	Elevée	Faible	Non
J2	Soleil	Chaud	Elevée	Fort	Non
J3	Couvert	Chaud	Elevée	Faible	Oui
J4	Pluie	Doux	Elevée	Faible	Oui
J5	Pluie	Froid	Normale	Faible	Oui
J6	Pluie	Froid	Normale	Fort	Non
J7	Couvert	Froid	Normale	Fort	Oui
J8	Soleil	Doux	Elevée	Faible	Non
J9	Soleil	Froid	Normale	Faible	Oui
J10	Pluie	Doux	Normale	Faible	Oui
J11	Soleil	Doux	Normale	Fort	Oui
J12	Couvert	Doux	Elevée	Fort	Oui
J13	Couvert	Chaud	Normale	Faible	Oui
J14	Pluie	Doux	Elevée	Fort	Non

- Algorithmes de Machine learning
- Arbre de décision (Decision Tree-DT)

Exercice 2 (solution)



■ Algorithmes de Machine learning

➤ Arbre de décision (Decision Tree-DT)

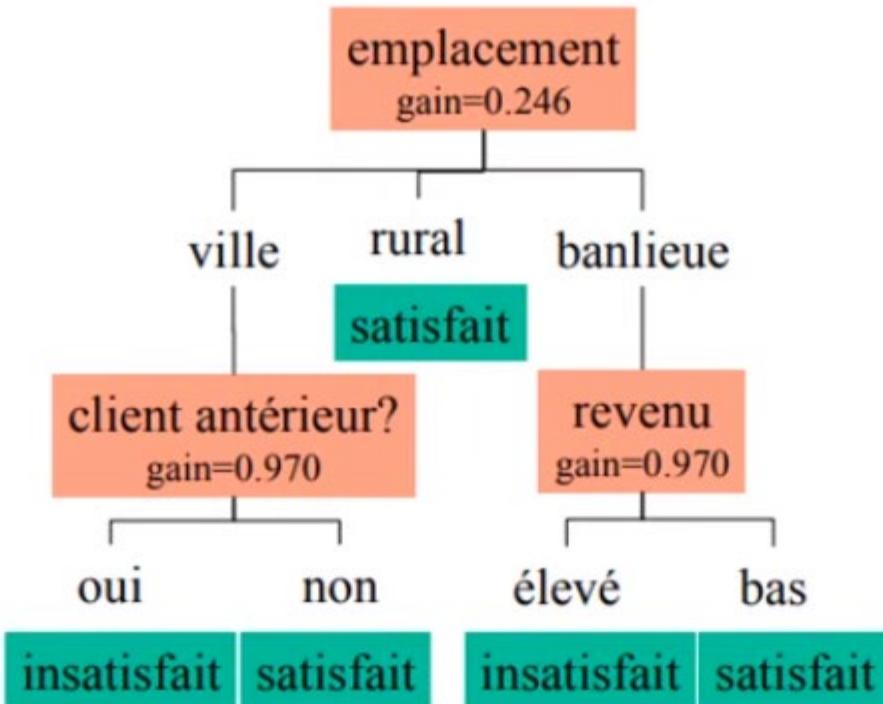
Exercice 3

Emplacement	Type de maison	Revenu	Client antérieur ?	Résultat
Banlieue	Unifamiliale	Elevé	Non	Insatisfait
Banlieue	Unifamiliale	Elevé	Oui	Insatisfait
Rural	Unifamiliale	Elevé	Non	Satisfait
Ville	Jumelée	Elevé	Non	Satisfait
Ville	Jumelée	Bas	Non	Satisfait
Ville	Jumelée	Bas	Oui	Insatisfait
Rural	Jumelée	Bas	Oui	Satisfait
Banlieue	Rangée	Elevé	Non	Insatisfait
Banlieue	Jumelée	Bas	Non	Satisfait
Ville	Rangée	Bas	Non	Satisfait
Banlieue	Rangée	Bas	Oui	Satisfait
Rural	Rangée	Elevé	Oui	Satisfait
Rural	Unifamiliale	Bas	Non	Satisfait
Ville	Rangée	Elevé	Oui	Insatisfait

- 1. Construire l'arbre de décision résultant de ces exemples.
- 2. Exprimer la classe des exemples positifs sous la forme d'un prédicat logique.

- Algorithmes de Machine learning
- Arbre de décision (Decision Tree-DT)

Exercice 3 (solution)



- Le prédictat logique des exemples positifs est :
- $P = (\text{emplacement}=\text{ville} \text{ et } \text{client antérieur}=\text{non}) \text{ ou } (\text{emplacement}=\text{rural}) \text{ ou } (\text{emplacement}=\text{banlieue} \text{ et } \text{revenu}=\text{bas})$

- **Algorithmes de Machine learning**
- Arbre de décision (Decision Tree-DT)

Les arbres de décision sont simples à comprendre et à visualiser.

Ils nécessitent peu de préparation des données.

Ils sont capables d'utiliser des données catégorielles et numériques.

Ils sont capables de traiter des problèmes multi-classe.

Implémentation pratique avec Python et sklearn (TP – Arbre de décision)

- **Algorithmes de Machine learning**

- **Machine à vecteurs de support (Support Vector Machine – SVM)**

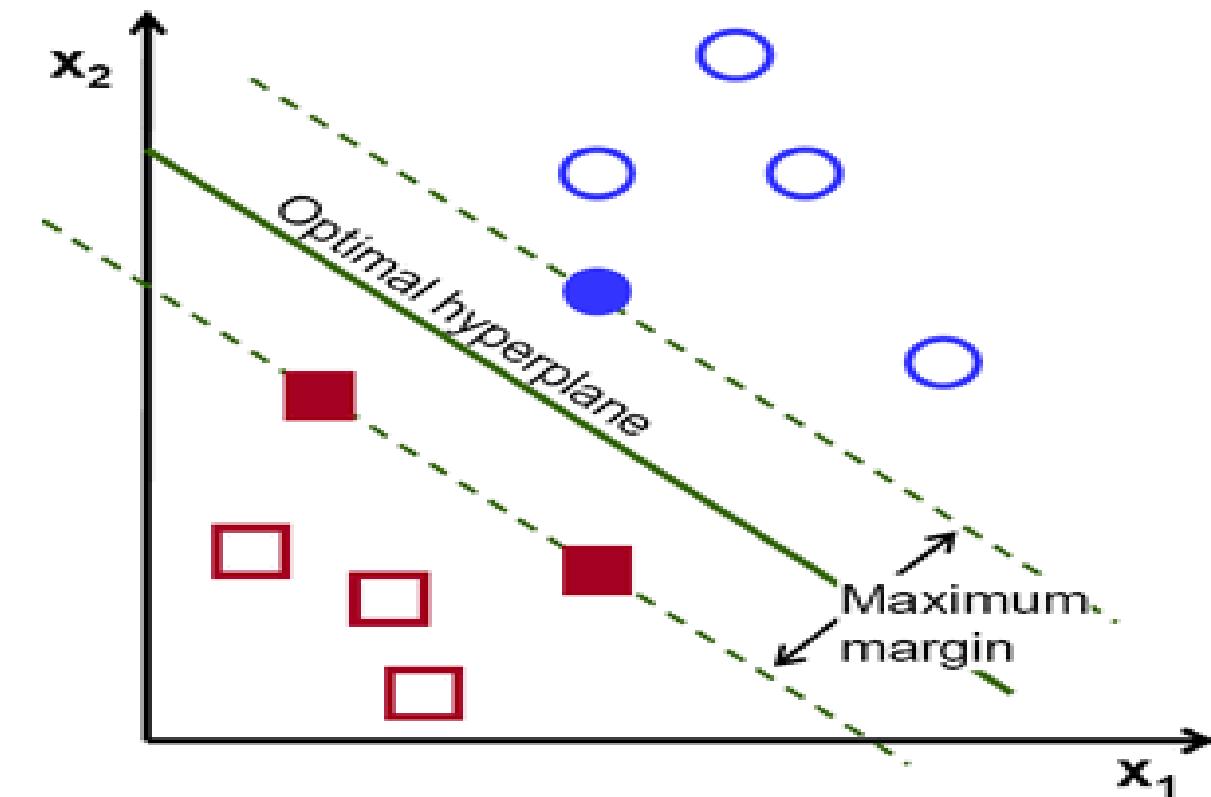
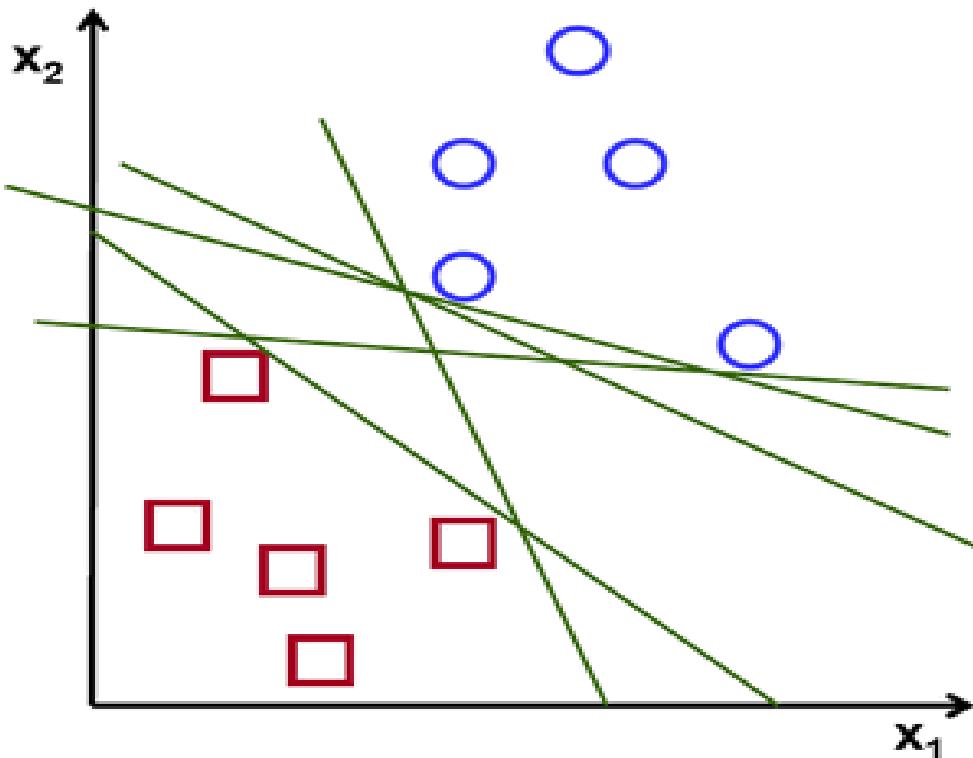
Définition

- La machine à vecteurs de support est un autre algorithme simple que en apprentissage automatique . La machine à vecteurs de support est fortement préférée car elle produit une précision significative avec moins de puissance de calcul. Support Vector Machine, abrégé en SVM, peut être utilisé à la fois pour les tâches de régression, de classification et la détection d'anomalie. Mais, il est largement utilisé dans les objectifs de classification.
- Il est considéré comme une généralisation des modèles linéaires
- L'un des modèles les plus populaires en machine learning, connue par sa garantie théorique, sa flexibilité, la simplicité d'utilisation (bio-informatique, recherche d'information, vision par ordinateur...)

- **Algorithmes de Machine learning**

- **Machine à vecteurs de support (Support Vector Machine – SVM)**

L'objectif de l'algorithme SVM est de trouver un hyperplan dans un espace à N dimensions (N - le nombre de caractéristiques) qui classe distinctement les points de données.

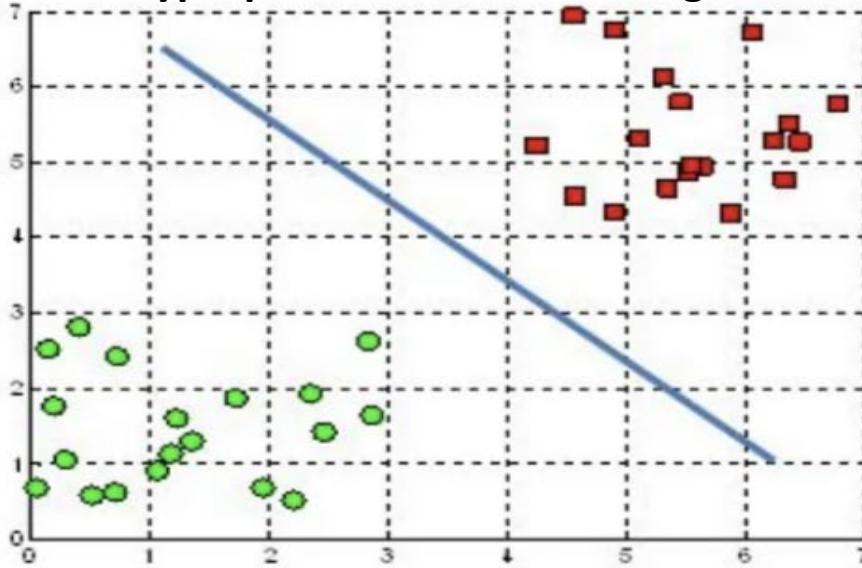


- **Algorithmes de Machine learning**

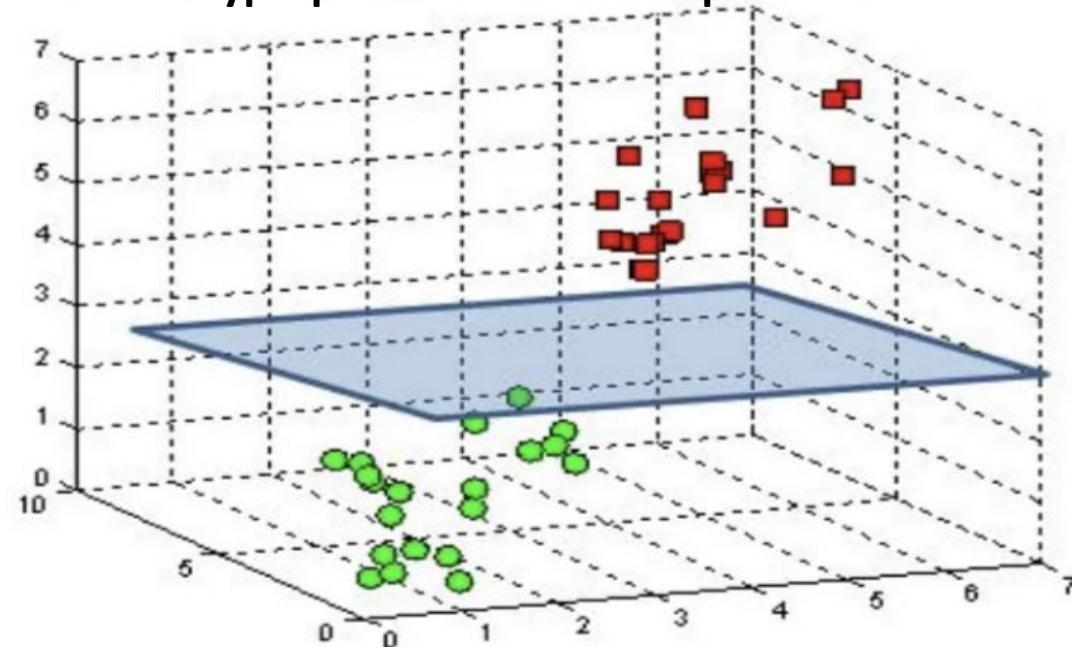
- **Machine à vecteurs de support (Support Vector Machine – SVM)**

➤ Pour séparer les deux classes de points de données, il existe de nombreux hyperplans possibles qui pourraient être choisis. Notre objectif est de trouver un plan qui a la marge maximale, c'est-à-dire la distance maximale entre les points de données des deux classes. La maximisation de la distance de marge fournit un certain renforcement afin que les futurs points de données puissent être classés avec plus de confiance.

Un hyperplan dans R^2 est une ligne



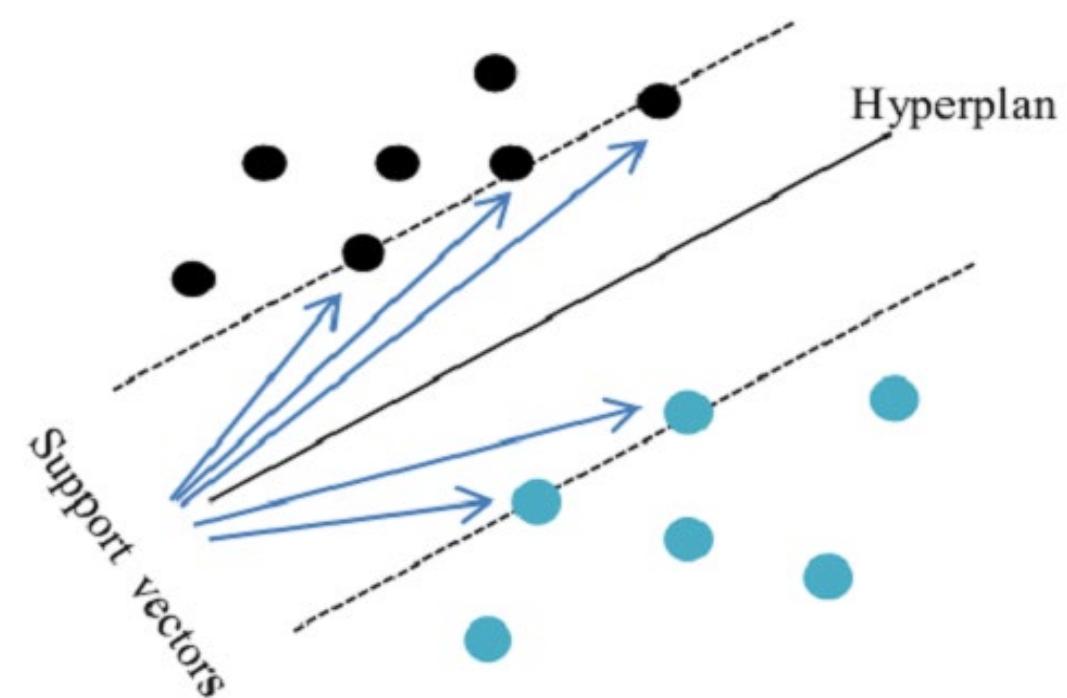
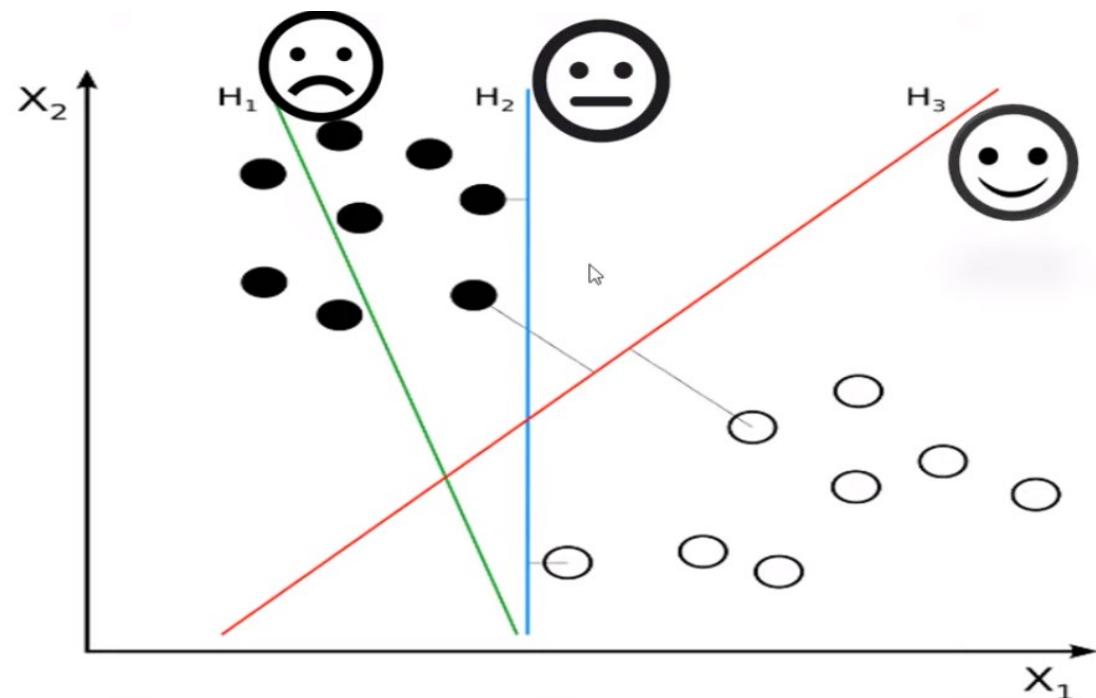
Un hyperplan dans R^3 est un plan



- **Algorithmes de Machine learning**

- **Machine à vecteurs de support (Support Vector Machine – SVM)**

- Les vecteurs de support sont les points situés sur les marges indiqués dans la figure. L'équation de l'hyperplan de séparation est définie par une équation linéaire qui sépare le jeu de données avec une marge maximale.

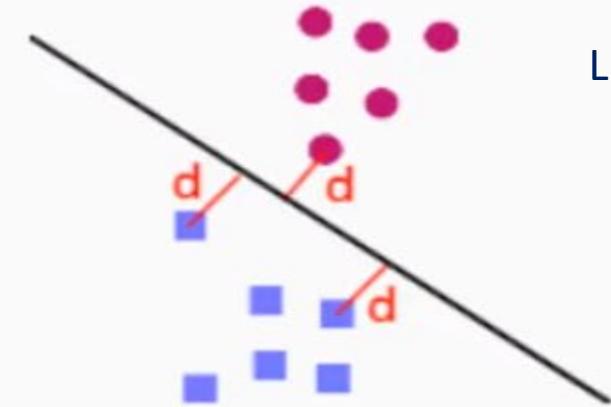
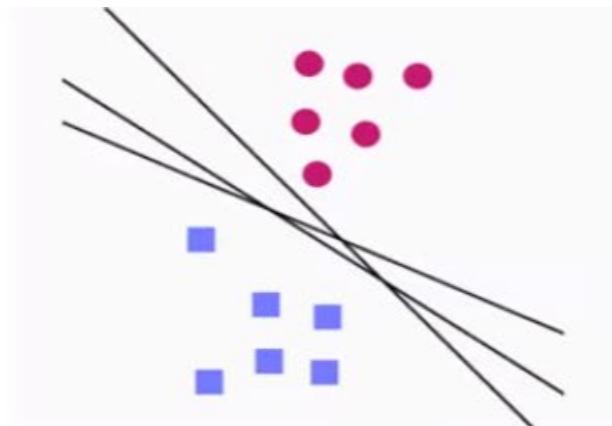


- **Algorithmes de Machine learning**

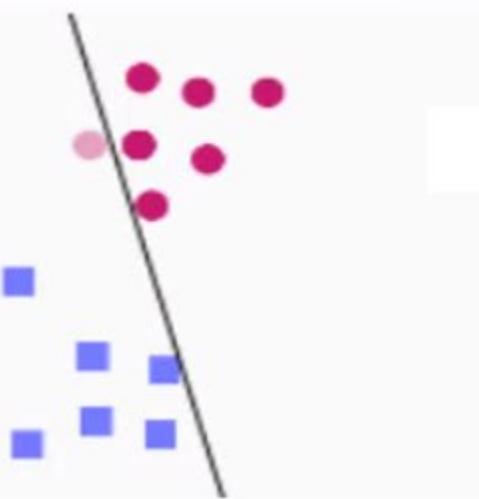
- **Machine à vecteurs de support (Support Vector Machine – SVM)**

Principe

d: distance



Linéairement séparable
(frontière)

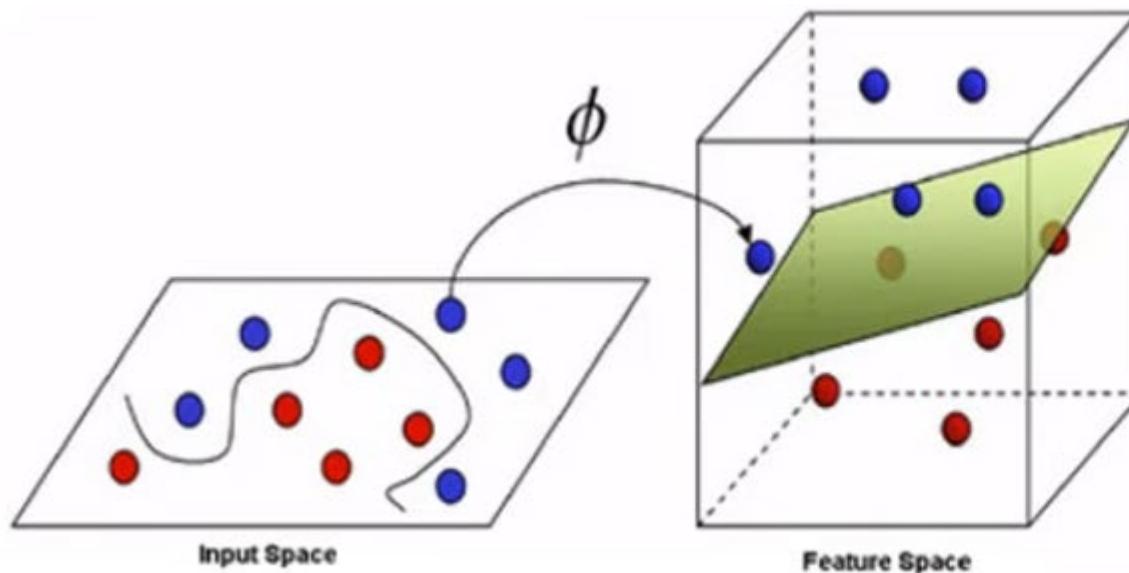


- **Algorithmes de Machine learning**

- **Machine à vecteurs de support (Support Vector Machine – SVM)**

Principe

- Cette notion de frontière suppose que les données soient linéairement séparables, ce qui est rarement le cas. Pour y pallier, les SVMs reposent souvent sur l'utilisation de « noyaux ». Ces fonctions mathématiques permettent de séparer les données en les projetant dans un **feature space** (un espace vectoriel de plus grande dimension). Ce qui permet de passer d'un problème non linéairement séparable vers un problème linéairement séparable



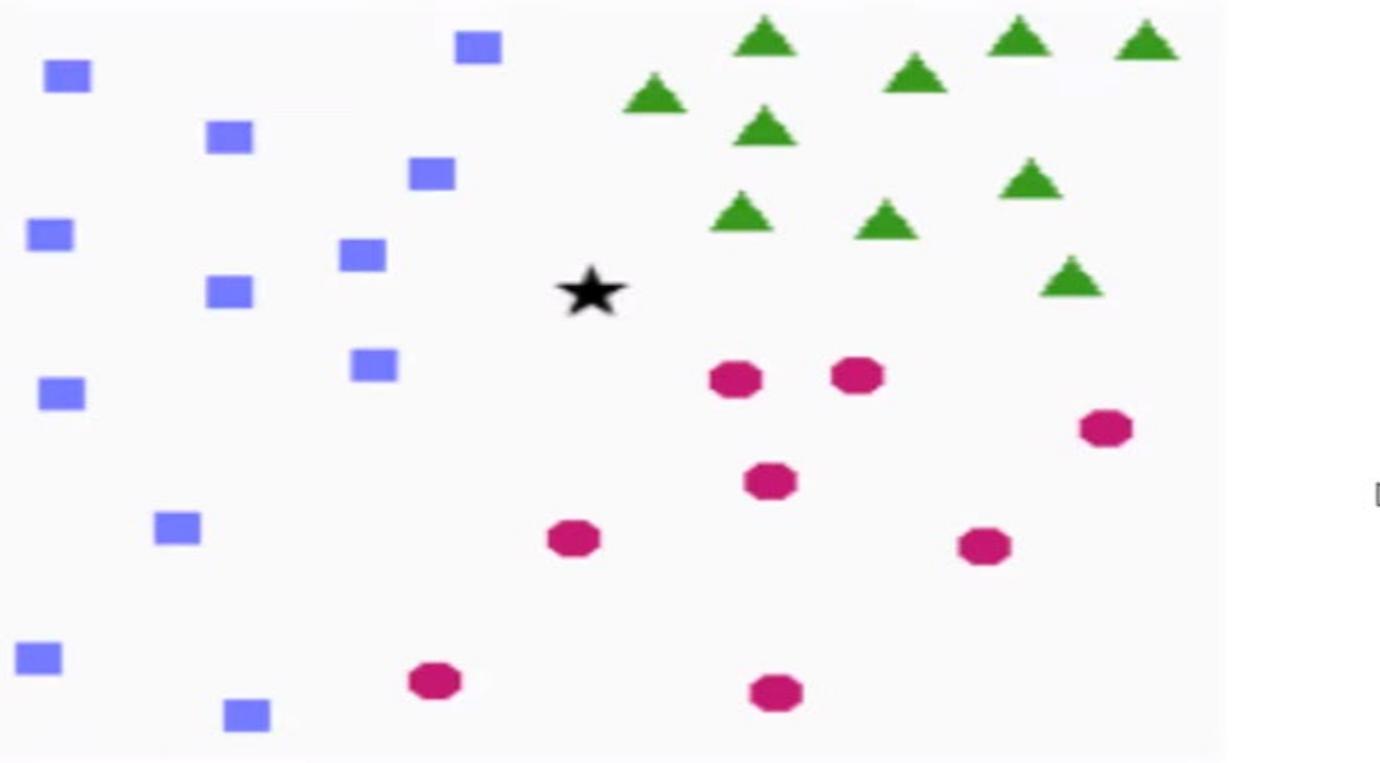
kernel trick
(Astuce du noyau)

- **Algorithmes de Machine learning**

- **Machine à vecteurs de support (Support Vector Machine – SVM)**

SVM Multi-class

Il existent plusieurs méthodes pour adapter le SVM à des problèmes Multi-class



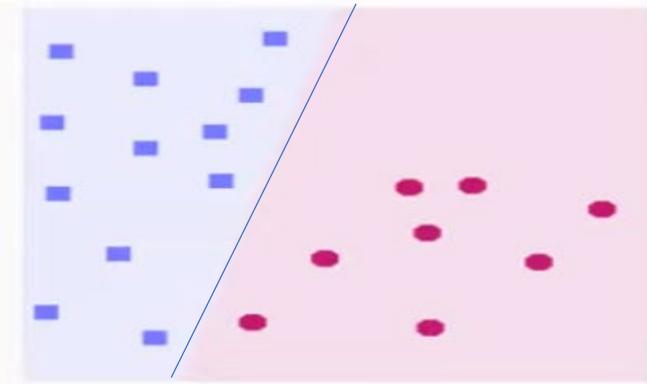
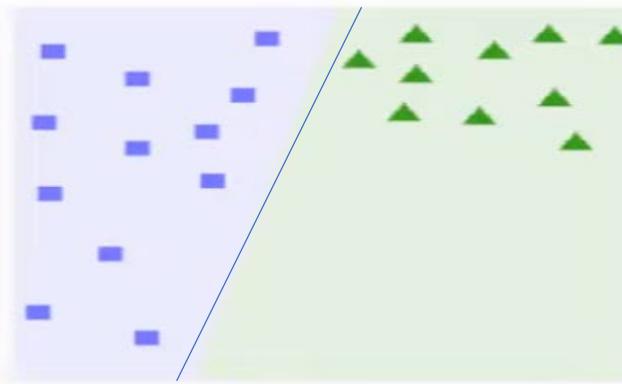
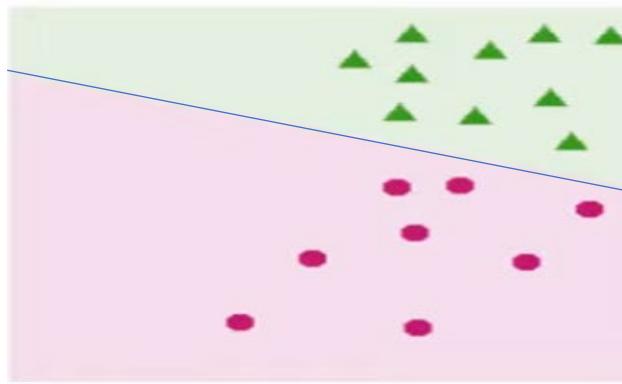
- **One vs one**
- **One vs all**

- **Algorithmes de Machine learning**

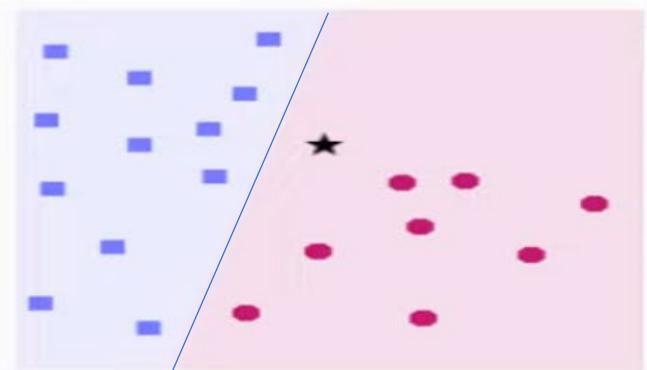
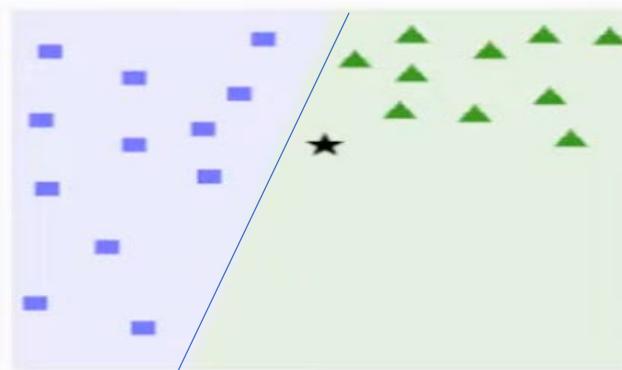
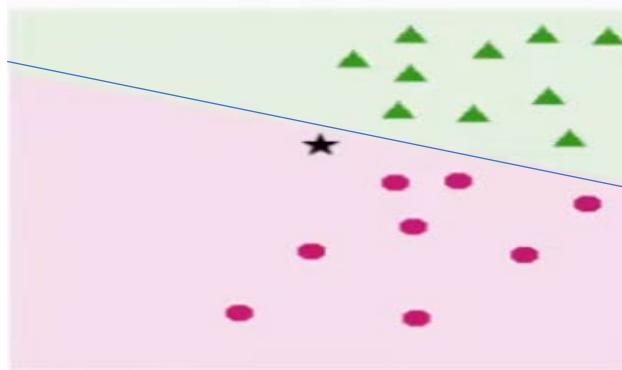
- **Machine à vecteurs de support (Support Vector Machine – SVM)**

SVM Multi-class (One vs One)

Dans cette méthode on s'intéresse à créer une frontières en prenant deux classes par deux, par exemple, **triangle** et **cercle** et trouver la frontière pour chaque cas. Pour classer une entrée on retourne à la catégorie qui aura reporté le plus de duel (ici c'est la catégorie cercle).



Nouvelle
Entrée

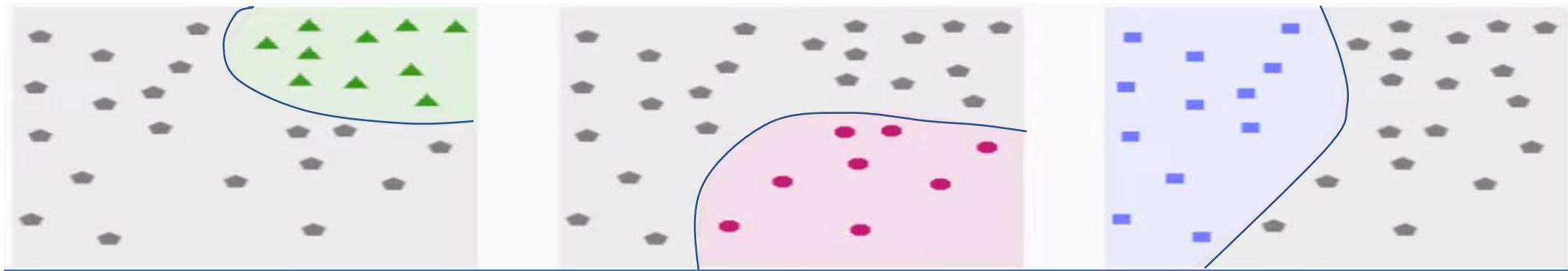


- **Algorithmes de Machine learning**

- **Machine à vecteurs de support (Support Vector Machine – SVM)**

SVM Multi-class (One vs All)

Cette méthode consiste à créer un SVM par catégorie. Pour entraîner un SVM spécialisé dans la reconnaissance des triangles, on crée deux catégories, la catégorie **triangle** qui contient toutes les entrées triangles et la catégorie **pas triangle** qui contient les autres entrées. De même pour les cercles etc... . Pour classer une entrée on regarde à quelle catégorie la nouvelle entrée est le plus probable d'appartenir.



Nouvelle
Entrée



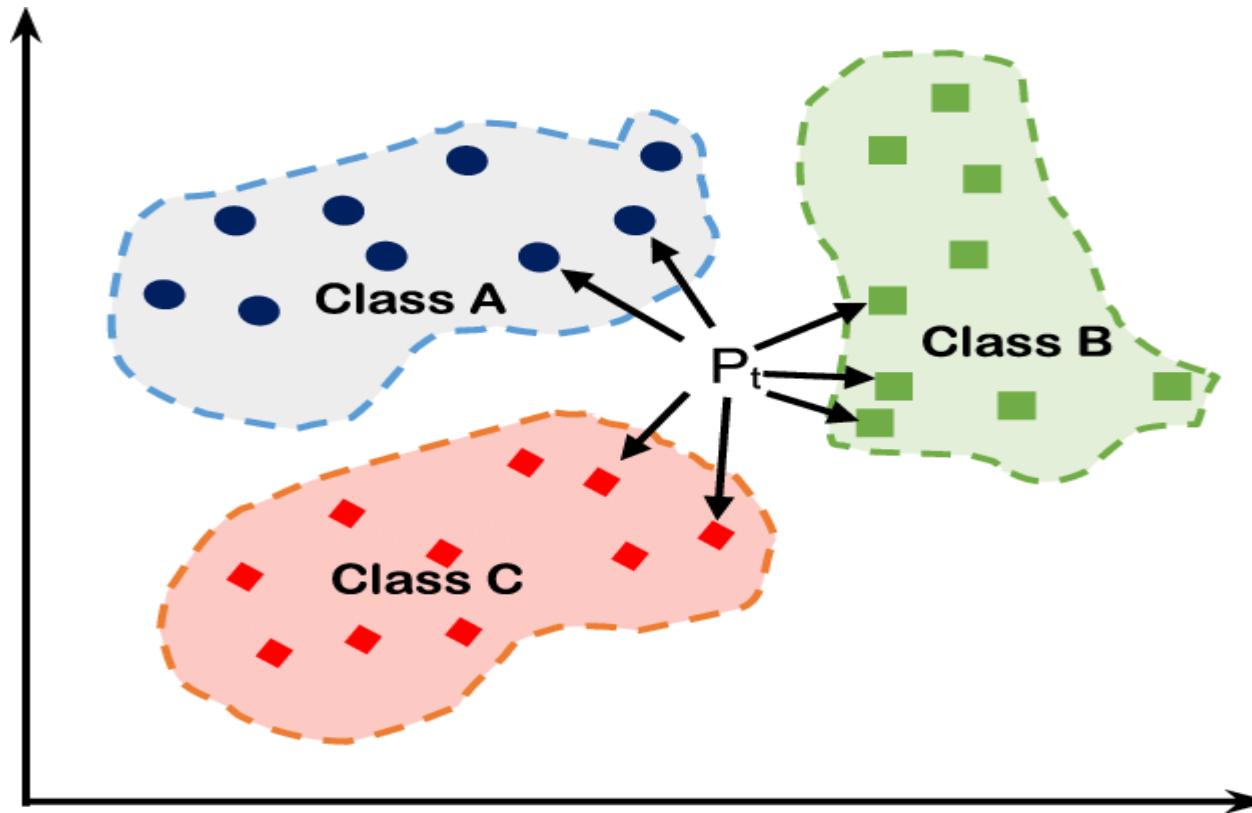
Implémentation pratique avec Python et sklearn (TP – SVM)

- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Définition

L'algorithme K-NN (K-nearest neighbors) est une méthode d'apprentissage supervisé. Il peut être utilisé aussi bien pour la régression que pour la classification. Son fonctionnement peut être assimilé à l'analogie suivante « dis moi qui sont tes voisins, je te dirais qui tu es... ». Le plus simples de tous les algorithmes de Machine Learning supervisé.



- **Algorithmes de Machine learning**

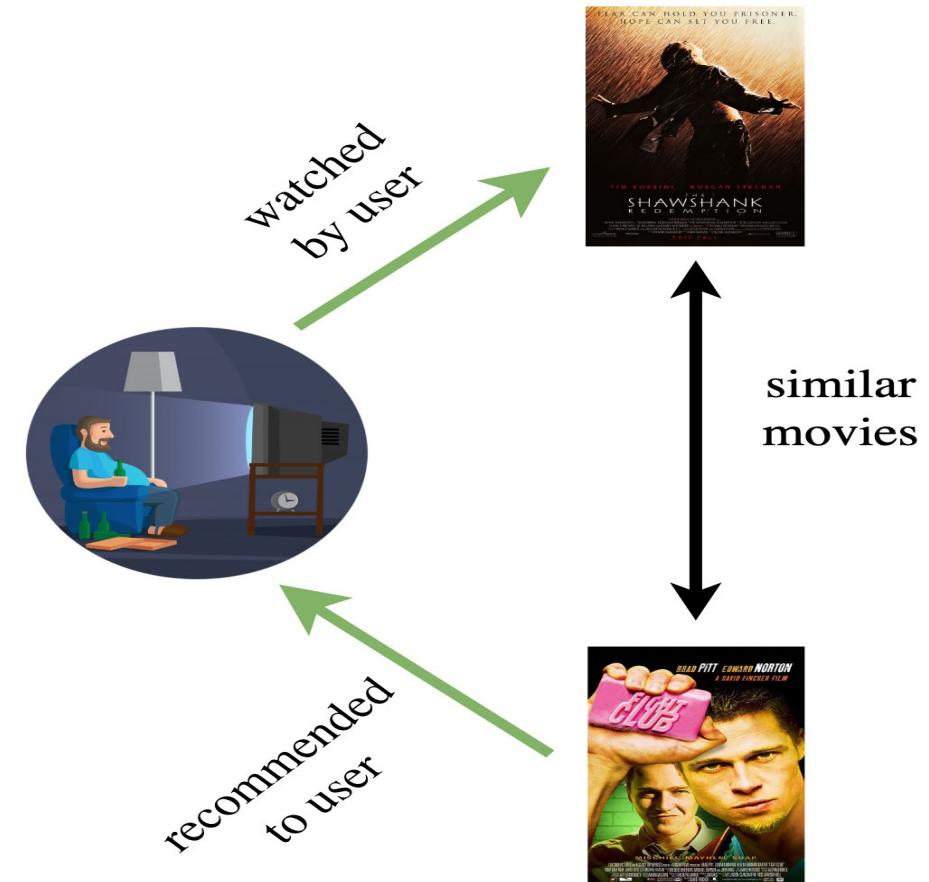
- **K plus proches voisins ou K-nearest neighbors (kNN)**

Exemple d'application

❑ L'exemple auquel vous avez tous déjà été confronté est celui de la recommandation après consultation d'un contenu. Vous regardez une vidéo en streaming et on vous propose d'autres vidéos similaires, susceptibles de vous intéresser. Comment faire ?

On attribue à chaque vidéo des caractéristiques et on lui donne un label en déterminant les autres vidéos dont elle est le plus proche.

On utilise plusieurs labels différents ("intérêt chez les moins de 18", "nombre de scènes d'action" etc.) et on détermine les plus représentatifs



- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

- Exemple d'application**

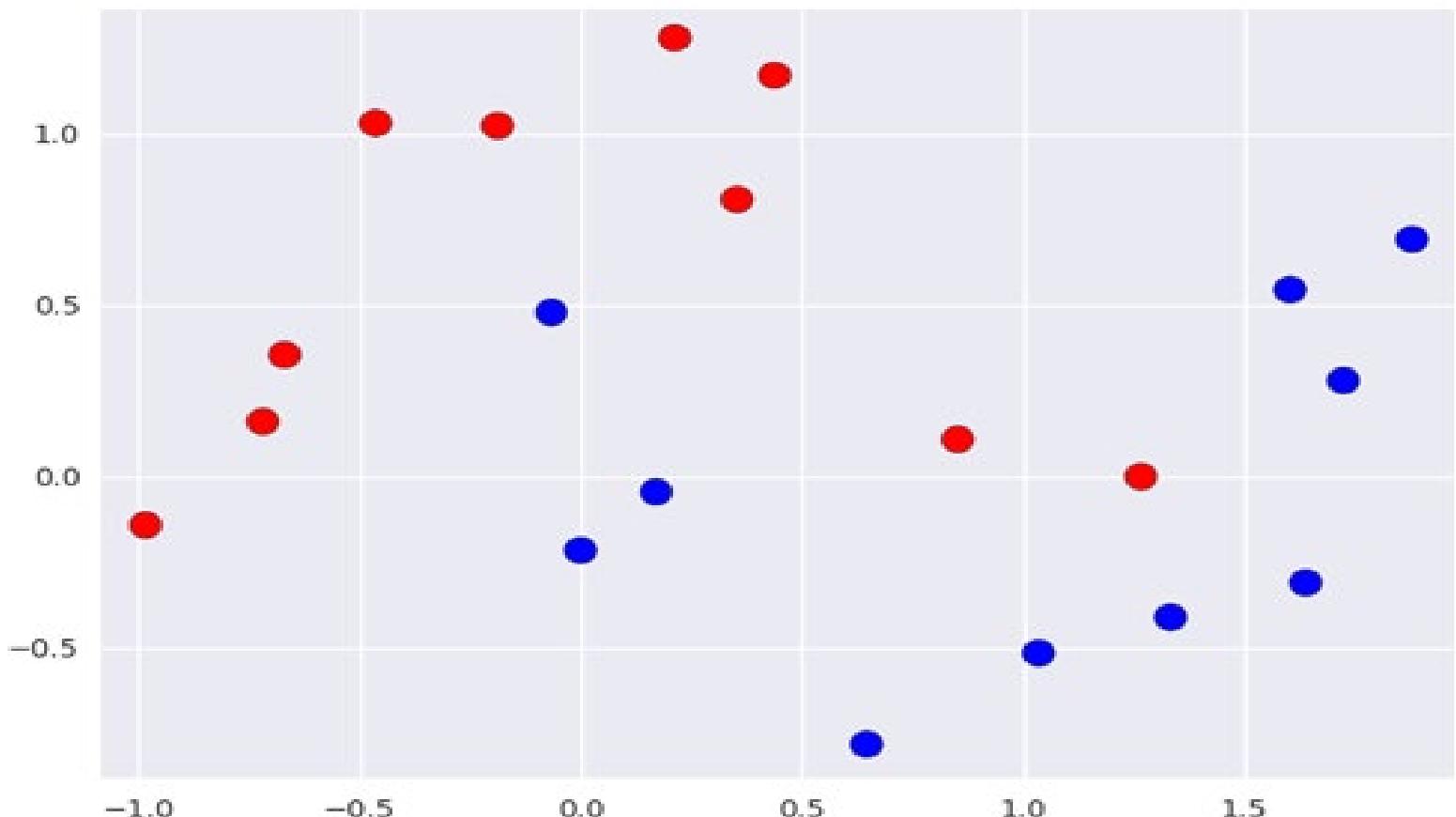
- ❑ Il peut être utilisé dans des technologies comme l'OCR (Optical Character Recognizer), qui tente de détecter l'écriture manuscrite, les images et même les vidéos.
 - ❑ Il peut être utilisé dans le domaine des notations de crédit. Il essaie de faire correspondre les caractéristiques d'un individu avec le groupe de personnes existantes afin de lui attribuer la cote de crédit. Il se verra attribuer la même note que celle accordée aux personnes correspondant à ses caractéristiques.
 - ❑ Il est utilisé pour prédire si la banque doit accorder un prêt à un particulier. Il tentera d'évaluer si l'individu donné correspond aux critères des personnes qui avaient précédemment fait défaut ou ne fera pas défaut à son prêt.

- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Fonctionnement de KNN

Avant d'utiliser un vrai jeu de données, on va prendre un petit exemple visuel afin de comprendre le fonctionnement de l'algorithme. Ci-dessous un jeu de données d'entraînement avec deux classes, rouge et bleu. L'input est donc bidimensionnel ici, et la target est la couleur à classer.

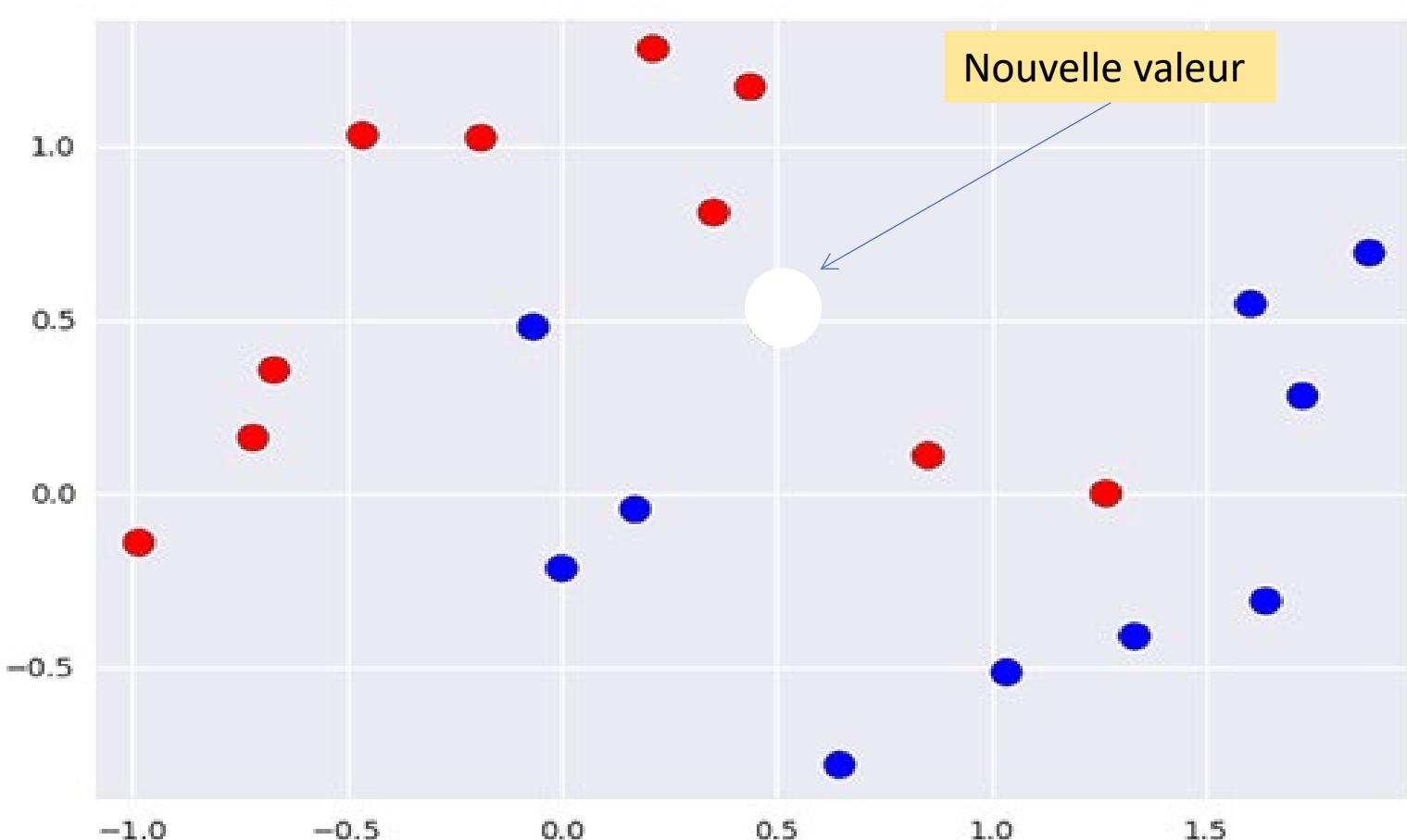


- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Fonctionnement de KNN

Si l'on a une nouvelle entrée dont on veut prédire la classe, comment pourrait-on faire ?

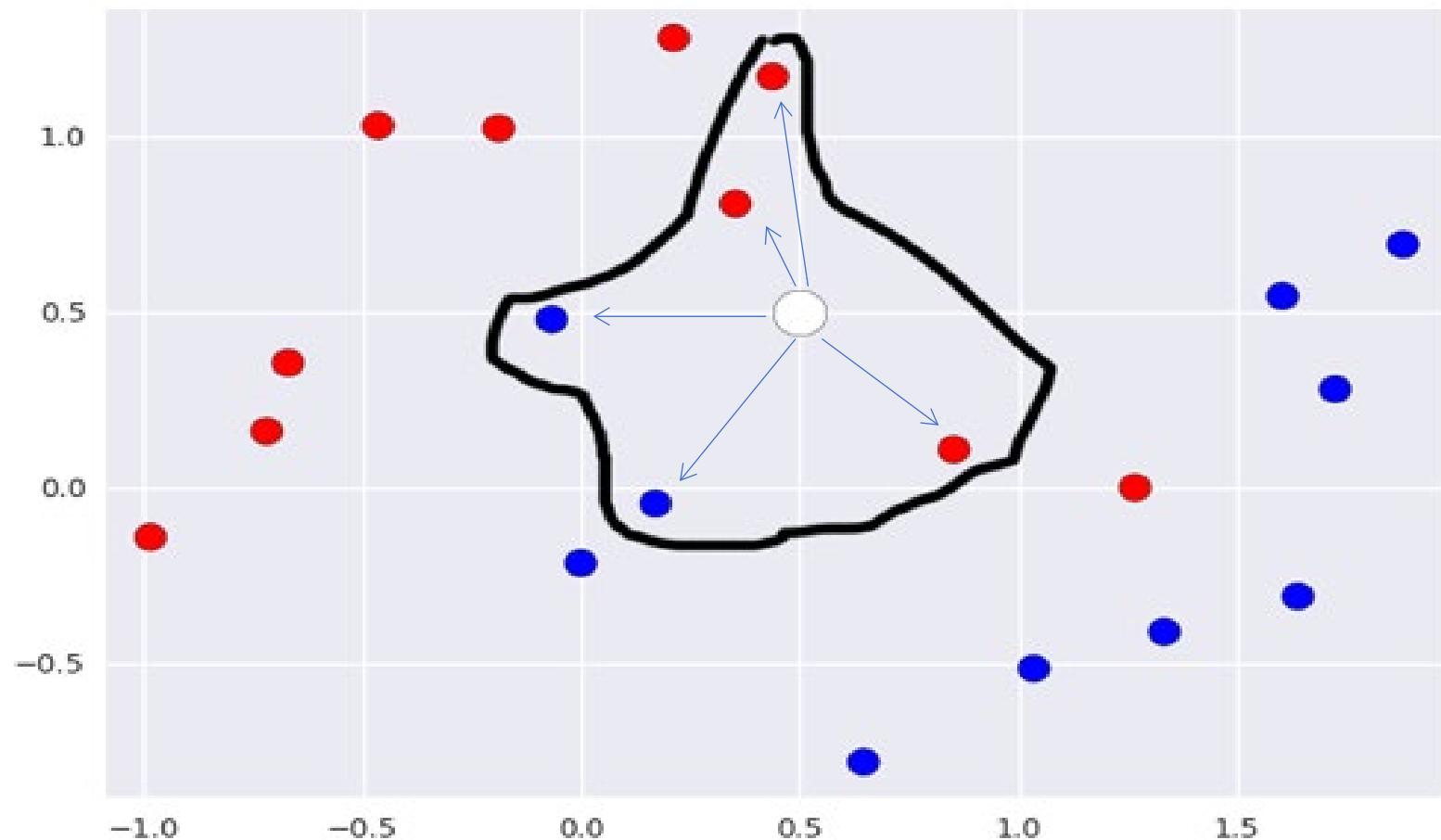


- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Fonctionnement de KNN

Et bien on va simplement regarder les k voisins les plus proches de ce point et regarder quelle classe constitue la majorité de ces points, afin d'en déduire la classe du nouveau point. Par exemple, ici, si on utilise le 5-NN, on peut prédire que la nouvelle donnée appartient à la classe rouge puisqu'elle a 3 rouges et 2 bleus dans son entourage.



- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Principe de KNN

Données en entrée :

Un ensemble de données D .

Une fonction de définition distance d.

Un nombre entier K

Comment choisir K ?
La phase la plus
importante

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

1- Calculer toutes les distances de cette observation X avec les autres observations du jeu de données D

2- Retenir les K observations du jeu de données D les proches de X en utilisant la fonction de calcul de distance d

3- Prendre les valeurs de y des K observations retenues :

- Si on effectue une régression, calculer la moyenne (ou la médiane) de y retenues

- Si on effectue une classification , calculer le mode de y retenues

4- Retourner la valeur calculée dans l'étape 3 comme étant la valeur qui a été prédite par K-NN pour l'observation X

- **Algorithmes de Machine learning**

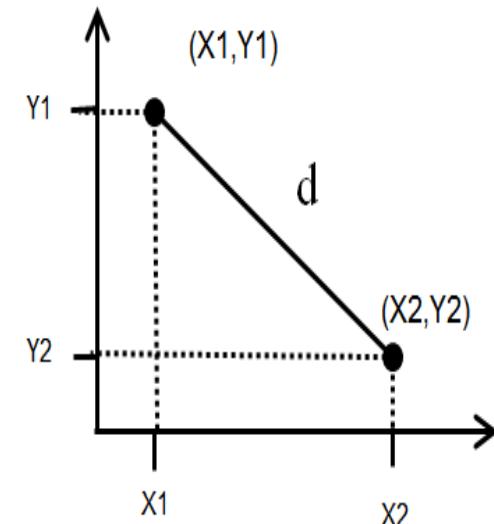
- **K plus proches voisins ou K-nearest neighbors (kNN)**

Calcul de similarité dans l'algorithme K-NN

Comme on vient de le voir dans notre écriture algorithme, K-NN a besoin d'une fonction de calcul de distance entre deux observations. Plus deux points sont proches l'un de l'autre, plus ils sont similaires et vice versa. Il existe plusieurs fonctions de calcul de distance, notamment, la distance euclidienne, la distance de Manhattan, la distance de Minkowski, celle de Jaccard, la distance de Hamming...etc. On choisit la fonction de distance en fonction des types de données qu'on manipule. Ainsi pour les données quantitatives (exemple : poids, salaires, taille, montant de panier électronique etc...) et **du même type**, la distance euclidienne est un bon candidat.

La distance Euclidienne

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$



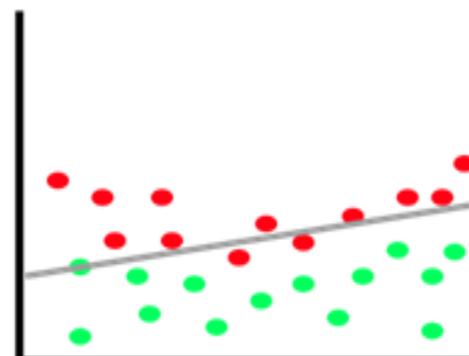
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- **Algorithmes de Machine learning**

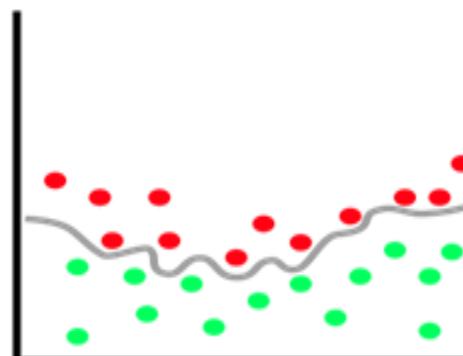
- **K plus proches voisins ou K-nearest neighbors (kNN)**

Comment choisir la valeur K ?

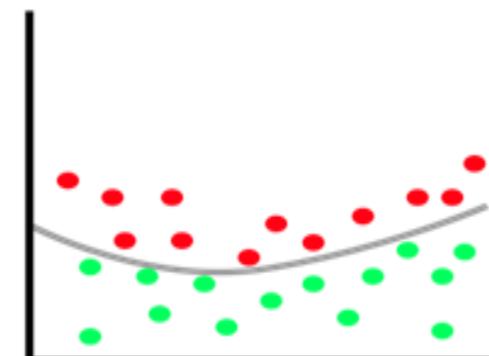
Le choix de la valeur K à utiliser pour effectuer une prédiction avec K-NN, varie en fonction du jeu de données. En règle générale, moins on utilisera de voisins (un nombre K petit) plus on sera sujette au sous-apprentissage (**underfitting**). Par ailleurs, plus on utilise de voisins (un nombre K grand)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



Underfitting



Overfitting



Balanced

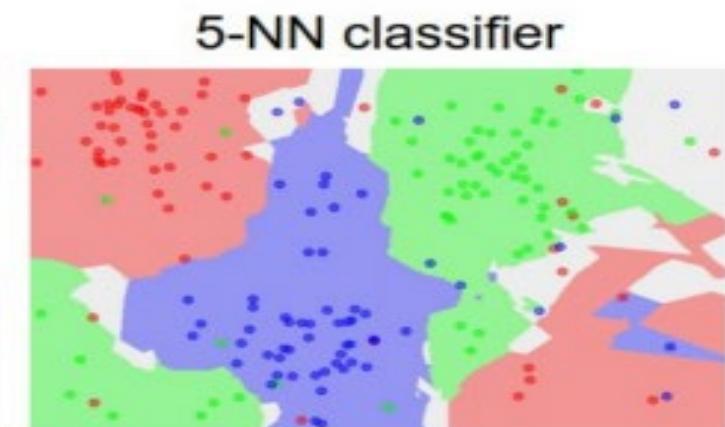
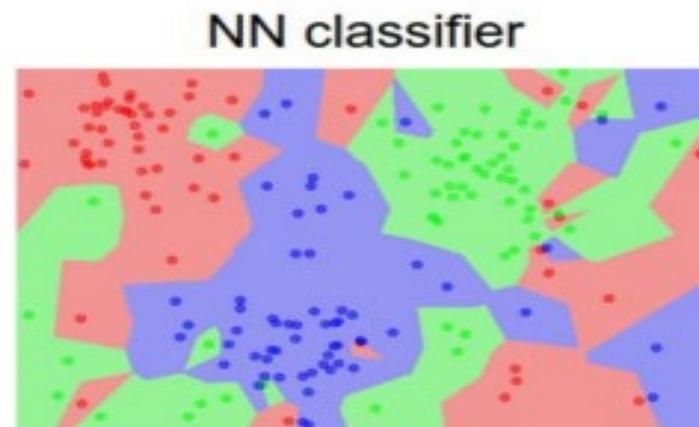
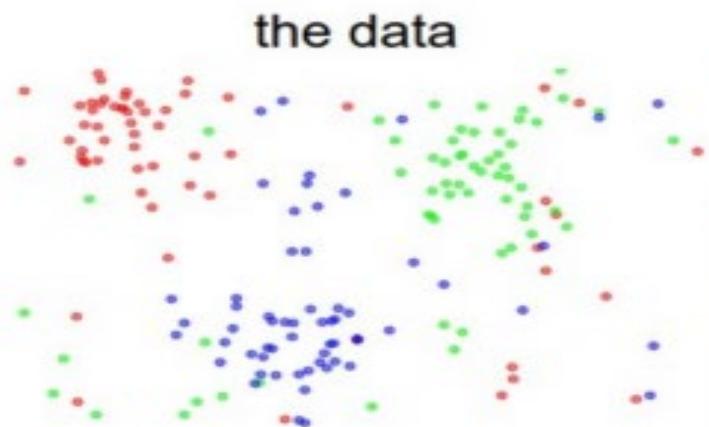
- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Comment choisir la valeur K ?

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

Pour cet exemple, on préférera le 5-NN classifier sur le NN-Classifier. Car le 5-NN classifier se généralise mieux que son opposant.

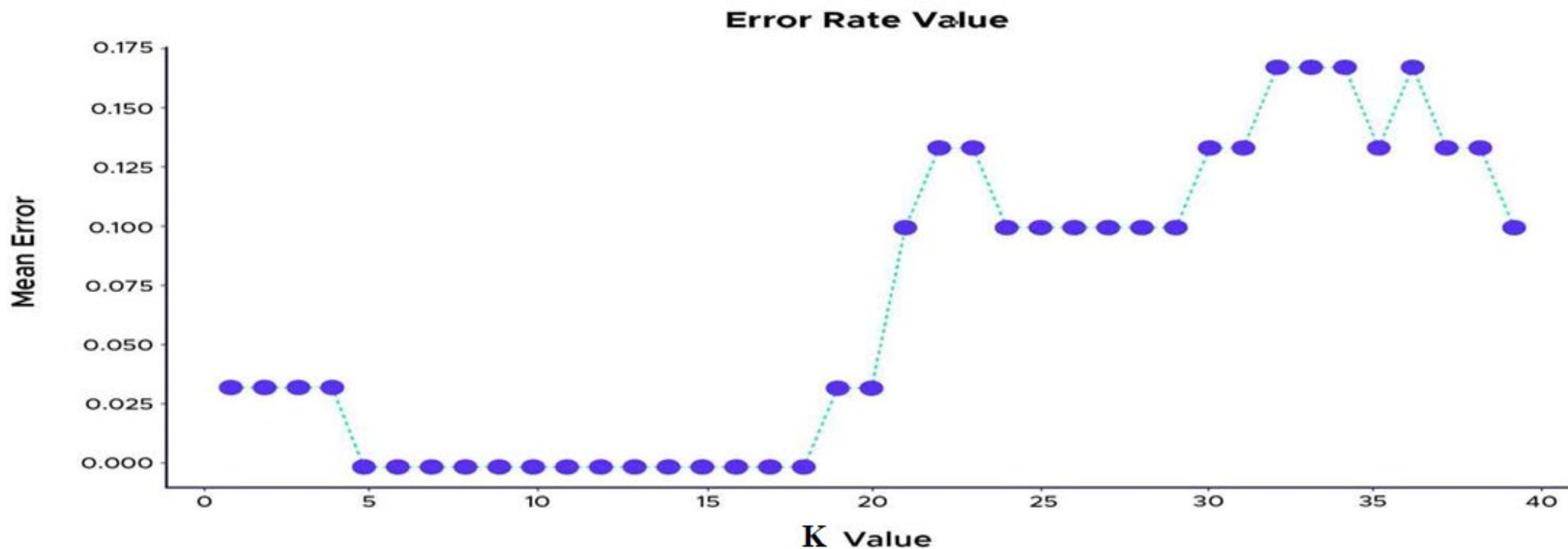


- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Comment choisir la valeur K ?

On peut également s'intéresser à un moyen de choisir le K pour lequel la classification sera la meilleure. Une façon de le trouver consiste à tracer le graphique de la valeur K et le taux d'erreur correspondant pour l'ensemble de données.



- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Exemple 1

Soit la base de données suivante D:

La question est de trouver Y dans le cas de

$$X = \{ X_1 = 3 \text{ et } X_2 = 7 \}$$

X_1	X_2	Y
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good
3	7	?

- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Exemple 1

1- On suppose K=3

2- Calculer toutes les distances de cette observation X avec les autres observations du jeu de données D

$$X = \{ X_1 = 3 \text{ et } X_2 = 7 \}$$

X₁	X₂	Distance
7	7	$\sqrt{(7 - 3)^2 + (7 - 7)^2} = 4$
7	4	$\sqrt{(7 - 3)^2 + (4 - 7)^2} = 5$
3	4	$\sqrt{(3 - 3)^2 + (4 - 7)^2} = 3$
1	4	$\sqrt{(1 - 3)^2 + (4 - 7)^2} = 3.6$

▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 1

3- Trier la distance et déterminer les voisins les plus proches en fonction de la distance minimale

Prendre les 3 plus proches voisins (K=3)

$$X = \{ X_1 = 3 \text{ et } X_2 = 7 \}$$

X_1	X_2	Distance	Rang (Rank)
7	7	$\sqrt{(7 - 3)^2 + (7 - 7)^2} = 4$	3
7	4	$\sqrt{(7 - 3)^2 + (4 - 7)^2} = 5$	4
3	4	$\sqrt{(3 - 3)^2 + (4 - 7)^2} = 3$	1
1	4	$\sqrt{(1 - 3)^2 + (4 - 7)^2} = 3.6$	2

■ Algorithmes de Machine learning

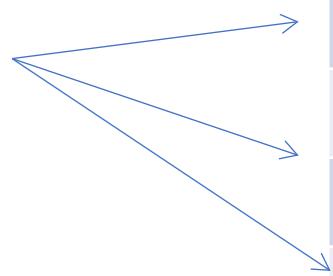
➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 1

4- Rassemble la catégorie des voisins les plus proches. Remarquez dans la deuxième ligne dernière colonne que la catégorie du plus proche voisin (y) n'est pas incluse car le rang de cette donnée est supérieur à 3 (k=3)

Prendre les 3 plus proches voisins
(K=3)

$$X = \{ X_1 = 3 \text{ et } X_2 = 7 \}$$



X_1	X_2	Distance	Rang (Rank)	Y
7	7	$\sqrt{(7 - 3)^2 + (7 - 7)^2} = 4$	3	Bad
7	4	$\sqrt{(7 - 3)^2 + (4 - 7)^2} = 5$	4	-
3	4	$\sqrt{(3 - 3)^2 + (4 - 7)^2} = 3$	1	Good
1	4	$\sqrt{(1 - 3)^2 + (4 - 7)^2} = 3.6$	2	Good

▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 1

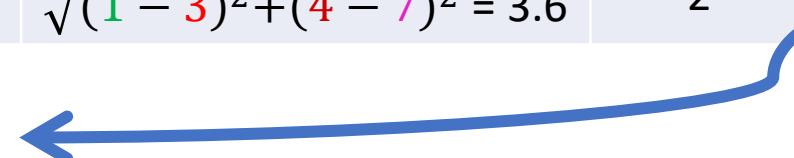
5- La classe prédictée sera la classe majoritaire des ces 3 plus proches voisins, ici deux classes Good, et une classe Bad, alors la classe de la nouvelle données X est Good.

Prendre les 3 plus proches voisins
(K=3)

$X = \{ X_1 = 3 \text{ et } X_2 = 7 \}$

X_1	X_2	Y
3	7	Good

X_1	X_2	Distance	Rang (Rank)	Y
7	7	$\sqrt{(7 - 3)^2 + (7 - 7)^2} = 4$	3	Bad
7	4	$\sqrt{(7 - 3)^2 + (4 - 7)^2} = 5$	4	-
3	4	$\sqrt{(3 - 3)^2 + (4 - 7)^2} = 3$	1	Good
1	4	$\sqrt{(1 - 3)^2 + (4 - 7)^2} = 3.6$	2	Good



▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 2

Prédire la classe de la nouvelle valeur en utilisant différent K, (K=1, K=2, K=3)

Sepal Length	Sepal Width	Classe
5.2	3.1	??

Sepal Length	Sepal Width	Classe
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Versicolor
7.3	2.9	Virginica
6.0	2.7	Versicolor
5.8	2.8	Virginica
6.3	2.3	Versicolor
5.1	2.5	Versicolor
6.3	2.5	Versicolor
5.5	2.4	Versicolor

▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 2

1-Calculer la distance

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe
5.2	3.1	??

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe	Distance
5.3	3.7	Setosa	0,608
5.1	3.8	Setosa	0,707
7.2	3.0	Virginica	2,002
5.4	3.4	Setosa	0,361
5.1	3.3	Setosa	0,224
5.4	3.9	Setosa	0,825
7.4	2.8	Virginica	2,220
6.1	2.8	Versicolor	0,949
7.3	2.9	Virginica	2,110
6.0	2.7	Versicolor	0,894
5.8	2.8	Virginica	0,671
6.3	2.3	Versicolor	1,360
5.1	2.5	Versicolor	0,608
6.3	2.5	Versicolor	1,253
5.5	2.4	Versicolor	0,762

▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 2

2-Trouver le Rang (Rank)

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe
5.2	3.1	??

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe	Distance	Rang (Rank)
5.3	3.7	Setosa	0,608	3
5.1	3.8	Setosa	0,707	6
7.2	3.0	Virginica	2,002	13
5.4	3.4	Setosa	0,361	2
5.1	3.3	Setosa	0,224	1
5.4	3.9	Setosa	0,825	8
7.4	2.8	Virginica	2,220	15
6.1	2.8	Versicolor	0,949	10
7.3	2.9	Virginica	2,110	14
6.0	2.7	Versicolor	0,894	9
5.8	2.8	Virginica	0,671	5
6.3	2.3	Versicolor	1,360	12
5.1	2.5	Versicolor	0,608	4
6.3	2.5	Versicolor	1,253	11
5.5	2.4	Versicolor	0,762	7

▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 2

3-Trouver les K plus proches voisins

Si K=1 la classe est Setosa

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe
5.2	3.1	Setosa

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe	Distance	Rang (Rank)
5.3	3.7	Setosa	0,608	3
5.1	3.8	Setosa	0,707	6
7.2	3.0	Virginica	2,002	13
5.4	3.4	Setosa	0,361	2
5.1	3.3	Setosa	0,224	1
5.4	3.9	Setosa	0,825	8
7.4	2.8	Virginica	2,220	15
6.1	2.8	Versicolor	0,949	10
7.3	2.9	Virginica	2,110	14
6.0	2.7	Versicolor	0,894	9
5.8	2.8	Virginica	0,671	5
6.3	2.3	Versicolor	1,360	12
5.1	2.5	Versicolor	0,608	4
6.3	2.5	Versicolor	1,253	11
5.5	2.4	Versicolor	0,762	7

▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 2

3-Trouver les K plus proches voisins

Si K = 2 la classe est Setosa

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe
5.2	3.1	Setosa

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe	Distance	Rang (Rank)
5.3	3.7	Setosa	0,608	3
5.1	3.8	Setosa	0,707	6
7.2	3.0	Virginica	2,002	13
5.4	3.4	Setosa	0,361	2
5.1	3.3	Setosa	0,224	1
5.4	3.9	Setosa	0,825	8
7.4	2.8	Virginica	2,220	15
6.1	2.8	Versicolor	0,949	10
7.3	2.9	Virginica	2,110	14
6.0	2.7	Versicolor	0,894	9
5.8	2.8	Virginica	0,671	5
6.3	2.3	Versicolor	1,360	12
5.1	2.5	Versicolor	0,608	4
6.3	2.5	Versicolor	1,253	11
5.5	2.4	Versicolor	0,762	7

▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 2

3-Trouver les K plus proches voisins

Si K = 3 la classe est Setosa

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe
5.2	3.1	Setosa

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe	Distance	Rang (Rank)
5.3	3.7	Setosa	0,608	3
5.1	3.8	Setosa	0,707	6
7.2	3.0	Virginica	2,002	13
5.4	3.4	Setosa	0,361	2
5.1	3.3	Setosa	0,224	1
5.4	3.9	Setosa	0,825	8
7.4	2.8	Virginica	2,220	15
6.1	2.8	Versicolor	0,949	10
7.3	2.9	Virginica	2,110	14
6.0	2.7	Versicolor	0,894	9
5.8	2.8	Virginica	0,671	5
6.3	2.3	Versicolor	1,360	12
5.1	2.5	Versicolor	0,608	4
6.3	2.5	Versicolor	1,253	11
5.5	2.4	Versicolor	0,762	7

▪ Algorithmes de Machine learning

➤ K plus proches voisins ou K-nearest neighbors (kNN)

Exemple 2

3-Trouver les K plus proches voisins

Si K = 4 la classe est Setosa

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe
5.2	3.1	Setosa

<i>Sepal Length</i>	<i>Sepal Width</i>	Classe	Distance	Rang (Rank)
5.3	3.7	Setosa	0,608	3
5.1	3.8	Setosa	0,707	6
7.2	3.0	Virginica	2,002	13
5.4	3.4	Setosa	0,361	2
5.1	3.3	Setosa	0,224	1
5.4	3.9	Setosa	0,825	8
7.4	2.8	Virginica	2,220	15
6.1	2.8	Versicolor	0,949	10
7.3	2.9	Virginica	2,110	14
6.0	2.7	Versicolor	0,894	9
5.8	2.8	Virginica	0,671	5
6.3	2.3	Versicolor	1,360	12
5.1	2.5	Versicolor	0,608	4
6.3	2.5	Versicolor	1,253	11
5.5	2.4	Versicolor	0,762	7

- **Algorithmes de Machine learning**

- **K plus proches voisins ou K-nearest neighbors (kNN)**

Avantages

L'algorithme est simple et facile à mettre en œuvre. Il n'est pas nécessaire de créer un modèle, de régler plusieurs paramètres ou de formuler des hypothèses supplémentaires.

L'algorithme est polyvalent. Il peut être utilisé pour la classification ou la régression.

Inconvénients

L'algorithme devient beaucoup plus lent à mesure que le nombre d'observation et de variables indépendantes augmente.

Implémentation pratique avec Python et sklearn (TP – KNN)

- **Algorithmes de Machine learning**

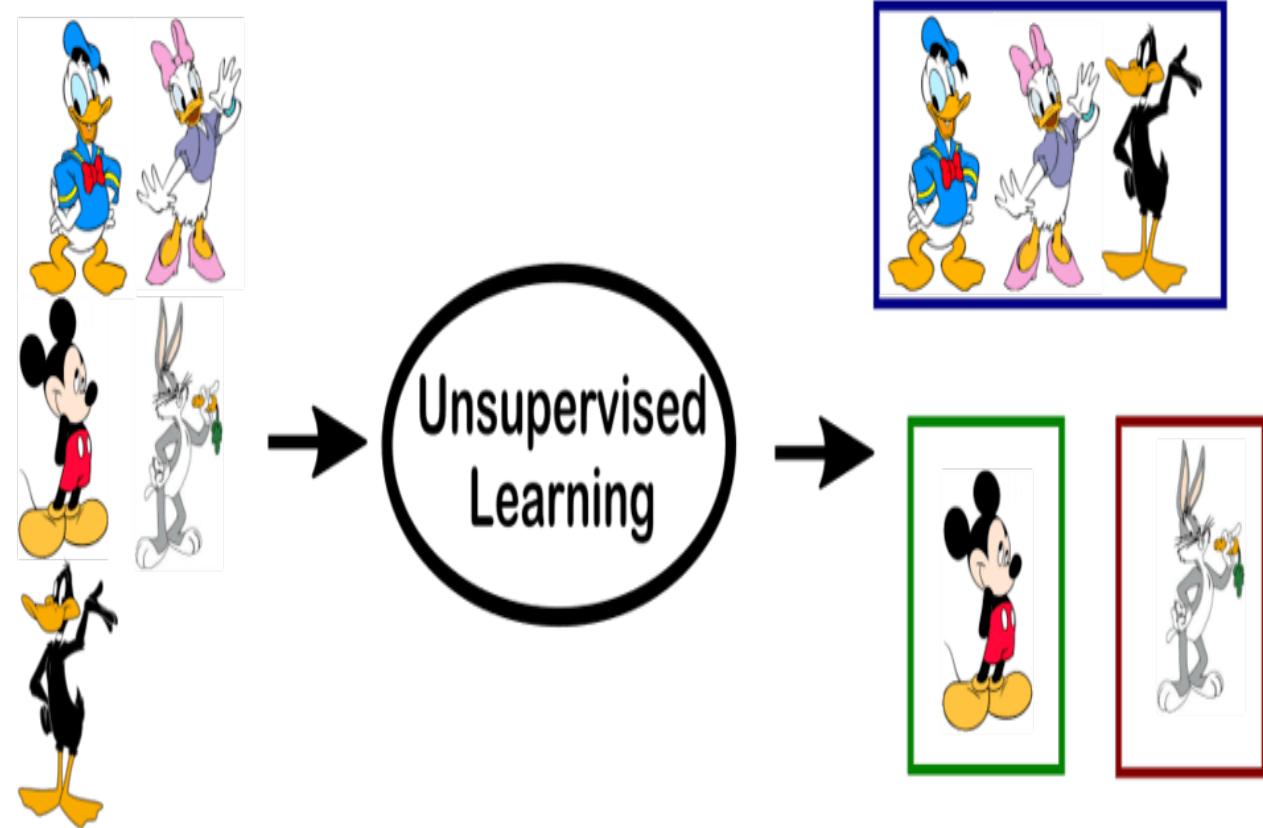
- **K-moyennes ou K-Means**

Définition

➤ K-means (k-moyennes) est un algorithme non supervisé de **clustering** (regroupement), populaire en Machine Learning.

- Le **clustering** est une méthode d'apprentissage non supervisé (unsupervised learning). Ainsi, on n'essaie pas d'apprendre une relation de corrélation entre un ensemble de features X d'une observation et une valeur à prédire Y, comme c'est le cas pour l'apprentissage supervisé.

- L'apprentissage non supervisé va plutôt trouver des patterns (motifs) dans les données. Notamment, en regroupant les choses qui se ressemblent.

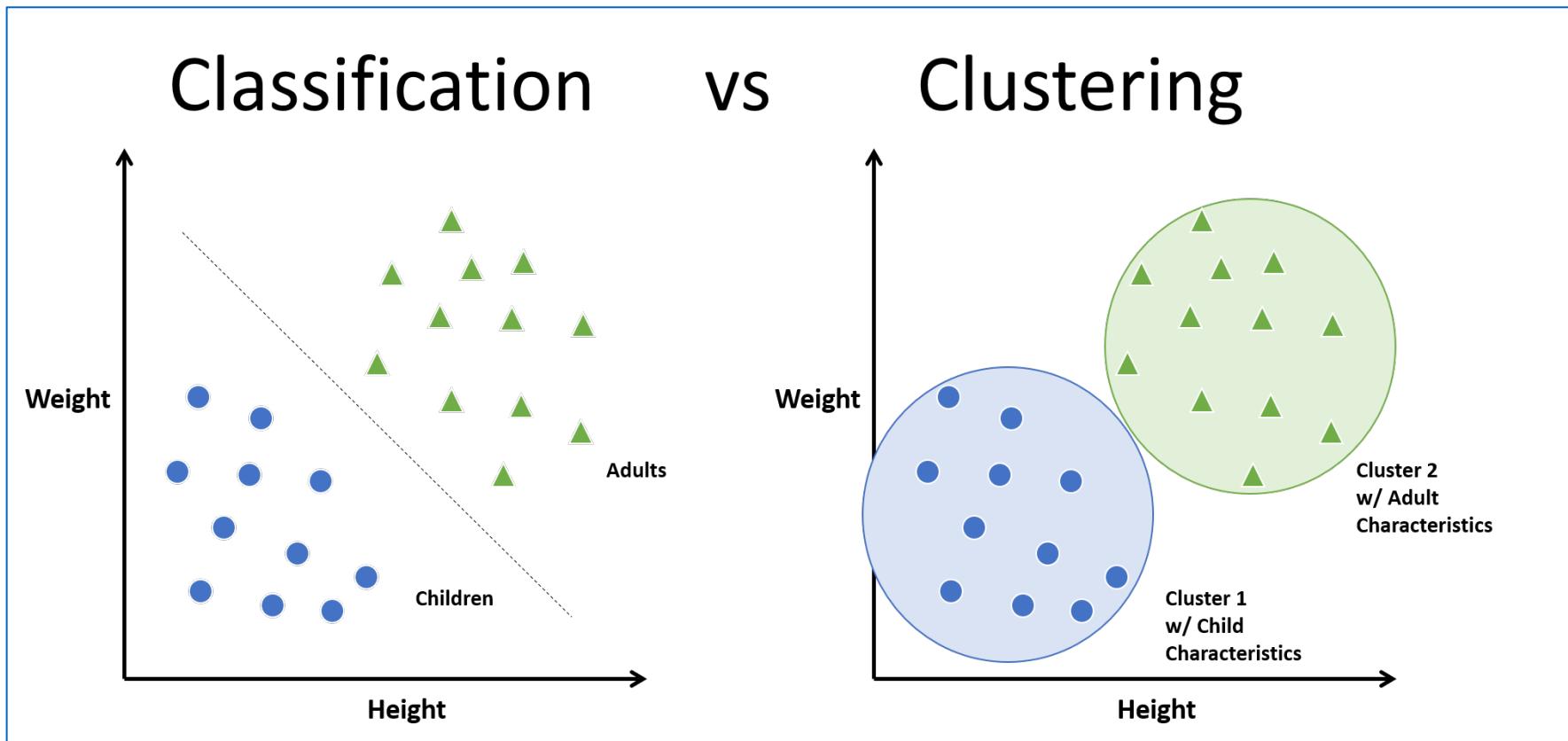


- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Définition

- Il consiste à regrouper les éléments de jeu de données en groupe appelé cluster, le but est de faire ressortir les motifs (pattern) cachés à partir des données, en regroupant les éléments qui se ressemblent.



- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Définition

- En apprentissage non supervisé, les données sont représentées comme suit :
- Chaque ligne représente un individu (une observation). A l'issu de l'application du clustering, on retrouvera ces données regroupées par ressemblance. Le clustering va regrouper en plusieurs familles (clusters) les individus/objets en fonction de leurs caractéristiques. Ainsi, les individus se trouvant dans un même cluster sont similaires et les données se trouvant dans un autre cluster ne le sont pas.
- K-means permet de regrouper en K clusters distincts les observations du dataset. Ainsi les données similaires se retrouveront dans un même cluster. Par ailleurs, une observation ne peut se retrouver que dans un cluster à la fois (exclusivité d'appartenance). Une même observation, ne pourra donc, appartenir à deux clusters différents.

Attributs (<i>features</i>)			
$x_{1,1}$	$x_{1,2}$...	$x_{1,p}$
$x_{2,1}$	$x_{2,2}$...	$x_{2,p}$
...
$x_{n,1}$	$x_{n,2}$...	$x_{n,p}$

- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Définition

- K-means cherche à créer des groupes d'individus homogènes. Le cas d'usage le plus classique de clustering est la segmentation client, comprendre et synthétiser une population et segmenter une image.



Créer des groupes d'individus homogènes

Des groupes dont lesquels
les individus se ressemblent

Des groupes qui se distinguent le plus
possible les uns des autres

- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Notion de similarité

Pour pouvoir regrouper un jeu de données en K cluster distincts, l'algorithme K-Means a besoin d'un moyen de comparer le degré de similarité entre les différentes observations. Ainsi, deux données qui se ressemblent, auront une distance de dissimilarité réduite, alors que deux objets différents auront une distance de séparation plus grande.

Les littératures mathématiques et statistiques regorgent de définitions de distance, les plus connues pour les cas de clustering est la distance **Euclidienne**. Soit une matrice X à n variables quantitatives. Dans l'espace vectoriel E^n . La distance euclidienne d entre deux observations x et y se calcule comme suit :

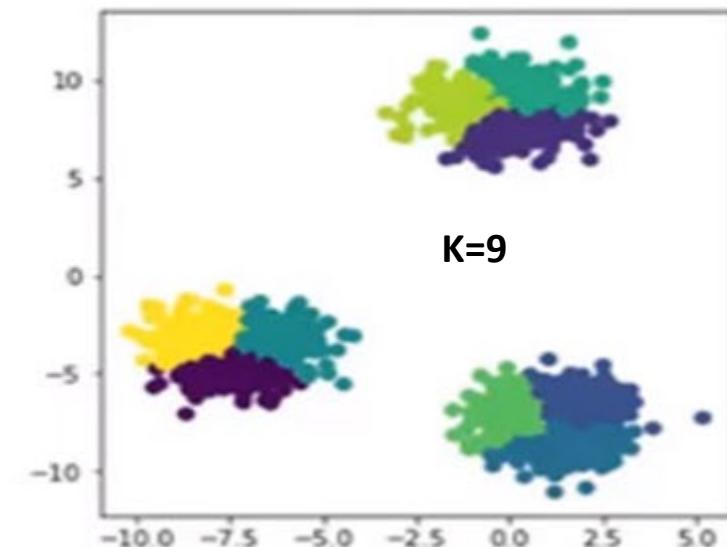
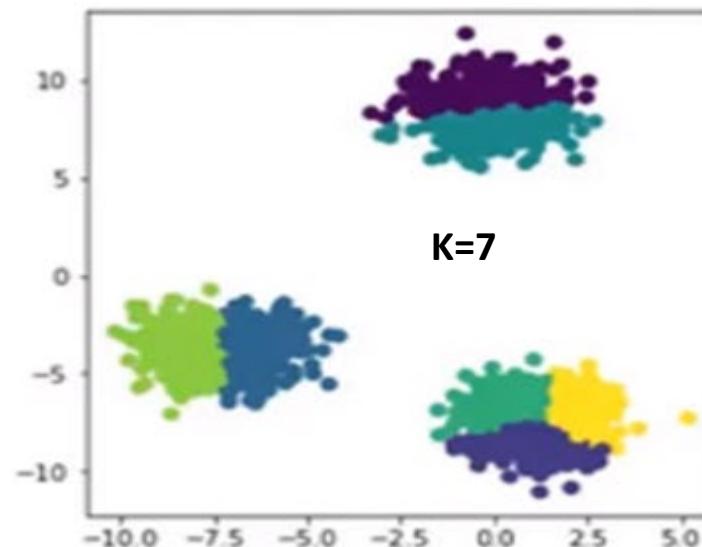
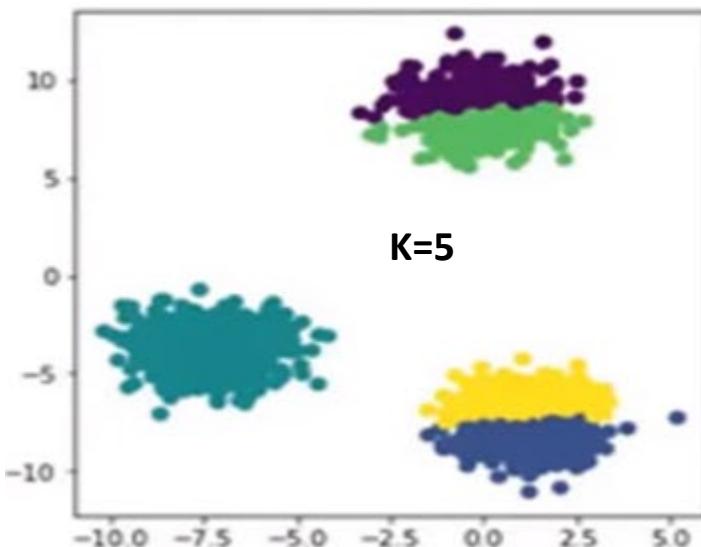
$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Choisir K: le nombre de clusters

- Choisir un nombre de cluster K n'est pas forcément intuitif. Spécialement quand le jeu de données est grand et qu'on n'ait pas un a priori ou des hypothèses sur les données. Un nombre K grand peut conduire à un partitionnement trop fragmenté des données. Ce qui empêchera de découvrir des patterns intéressants dans les données. Par contre, un nombre de clusters trop petit, conduira à avoir, potentiellement, des cluster trop généralistes contenant beaucoup de données.

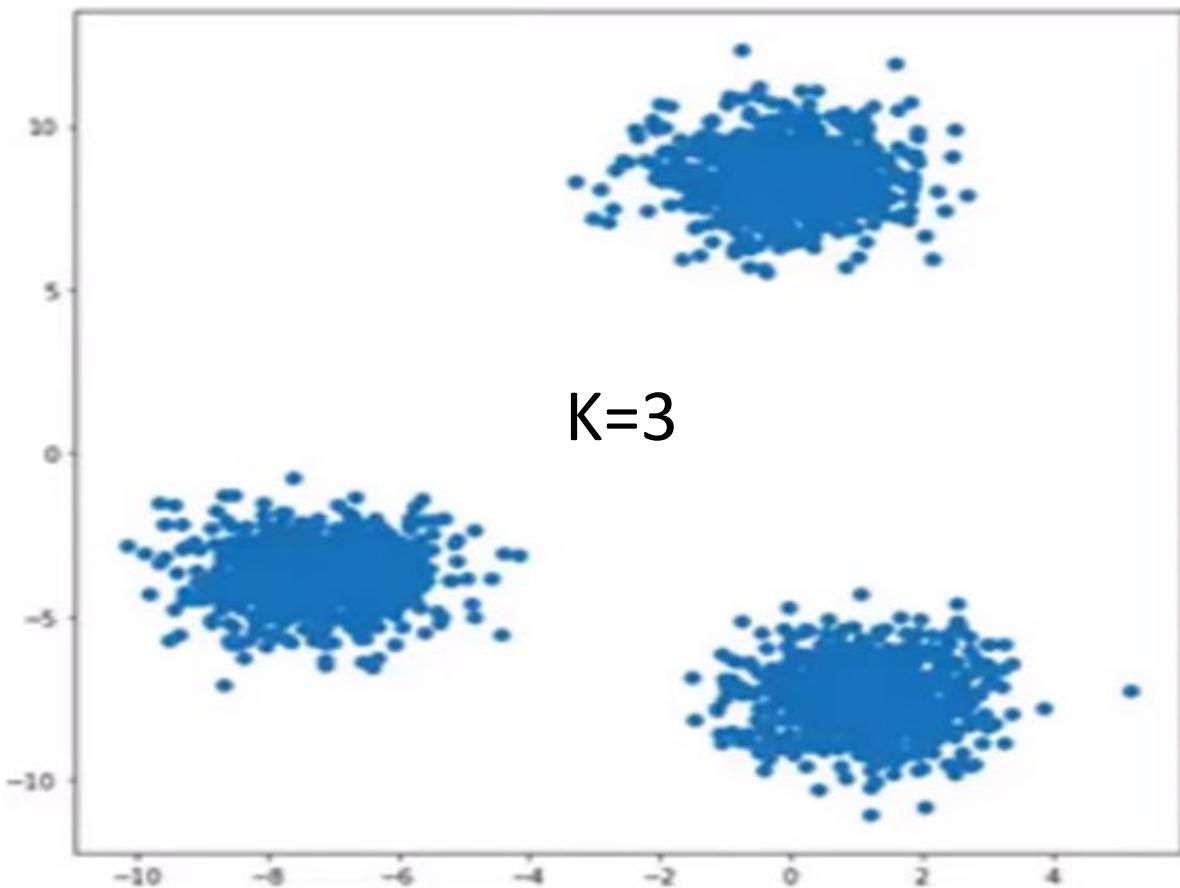


- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Choisir K: le nombre de clusters

- La méthode la plus usuelle pour choisir le nombre de clusters est de lancer K-Means avec différentes valeurs de K et de calculer la variance des différents clusters. La variance est la somme des distances entre chaque centroid d'un cluster et les différentes observations incluses dans le même cluster.
- Ainsi, on cherche à trouver un nombre de clusters K de telle sorte que les clusters retenus minimisent la distance entre leurs centres (centroids) et les observations dans le même cluster. On parle de minimisation de la distance intra-classe.

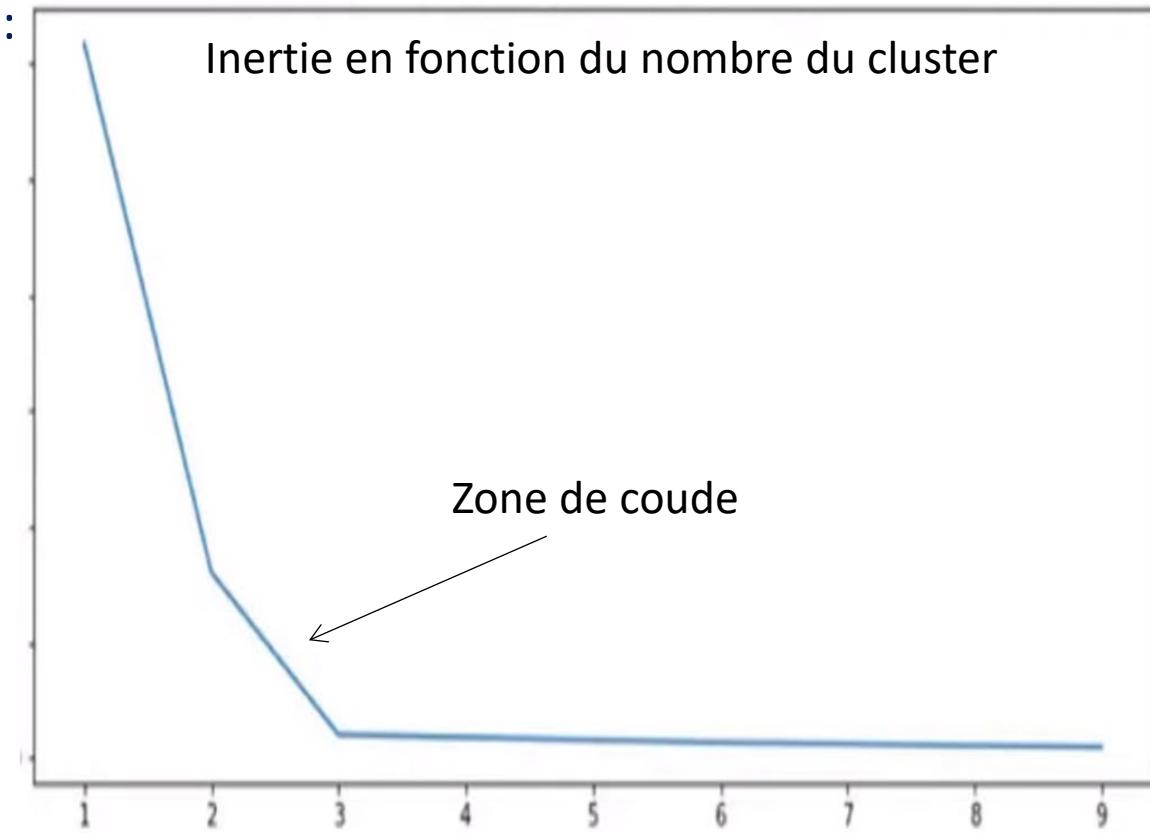


- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Choisir K: le nombre de clusters

- Généralement, en mettant dans un graphique les différents nombres de clusters K en fonction de la variance, on retrouve un graphique similaire à celui-ci :
- On remarque sur ce graphique, Le nombre optimal de clusters est le point représentant le **coude**. Ici le coude peut être représenté par K valant 2 ou 3. C'est le nombre optimal de clusters. Généralement, le point du coude est celui du nombre de clusters à partir duquel la variance ne se réduit plus significativement. En effet, la "chute" de la courbe de variance (distortion) entre 1 et 3 clusters est significativement plus grande que celle entre 5 clusters et 9 clusters. Le fait de chercher le point représentant le coude, a donné nom à cette méthode : La méthode **Elbow**



- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Cas d'utilisation K-means

- K-Means en particulier et les algorithmes de clustering de façon générale ont tous un objectif commun : Regrouper des éléments similaires dans des clusters. Ces éléments peuvent être tous et n'importe quoi, du moment qu'ils sont encodés dans une matrice de données.

Les champs d'application de K-Means sont nombreux, il est notamment utilisé en :

- La segmentation de la clientèle en fonction d'un certain critère (démographique, habitude d'achat etc....)
- Utilisation du clustering en Data Mining lors de l'exploration de données pour déceler des individus similaires. Généralement, une fois ces populations détectées, d'autres techniques peuvent être employées en fonction du besoin.
- Clustering de documents (regroupement de documents en fonction de leurs contenus. Pensez à comment Google Actualités regroupe des documents par thématiques.)

- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Principe

Entrée :

K le nombre de cluster à former

Le Training Set (matrice de données)

DEBUT

Choisir aléatoirement K points (une ligne de la matrice de données). Ces points sont les centres des clusters (nommé centroïd).

REPETER

Affecter chaque point (élément de la matrice de donnée) au groupe dont il est le plus proche au son centre

Recalculer le centre de chaque cluster et modifier le centroïde

JUSQU'A CONVERGENCE

OU (stabilisation de l'inertie totale de la population)

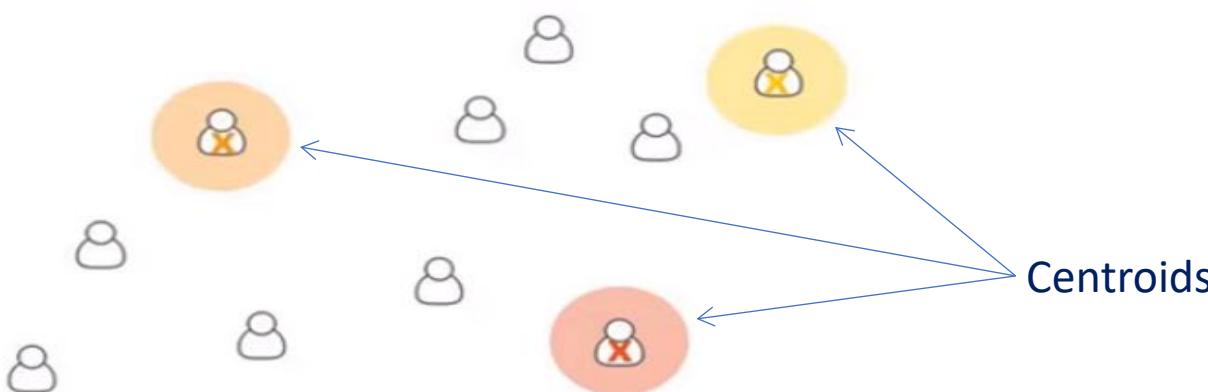
FIN

Stabilisation des centres de clusters (les centroids ne bougent plus lors des itérations).

- **Algorithmes de Machine learning**

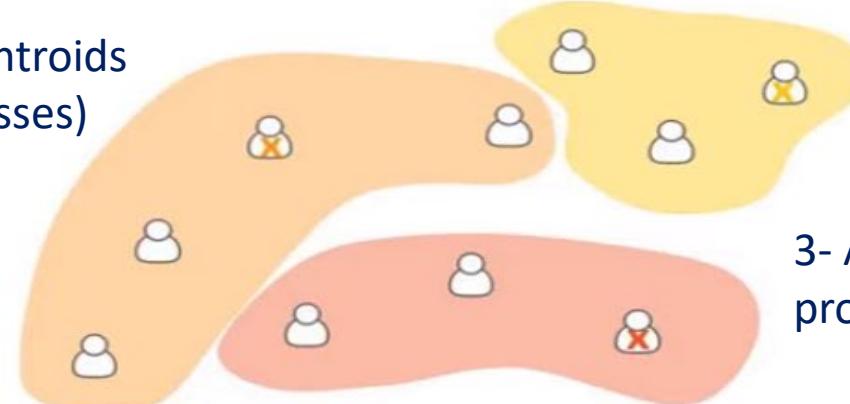
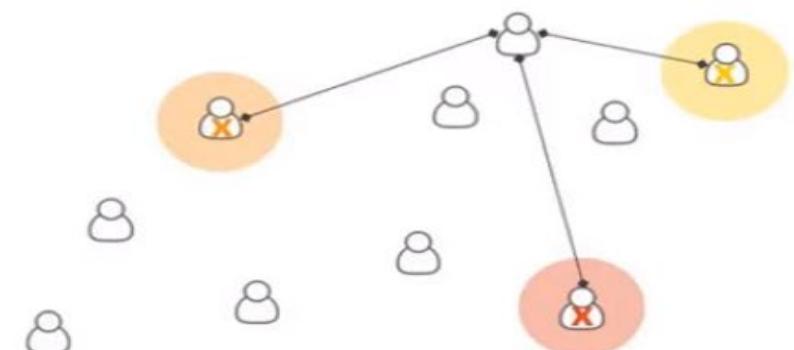
- **K-moyennes ou K-Means**

Exemple On souhaite créer trois groupes



1-On tire aléatoirement trois centroids
(centres initiaux de nos trois classes)

2-On calcule la distance entre les individus
Et chaque centre

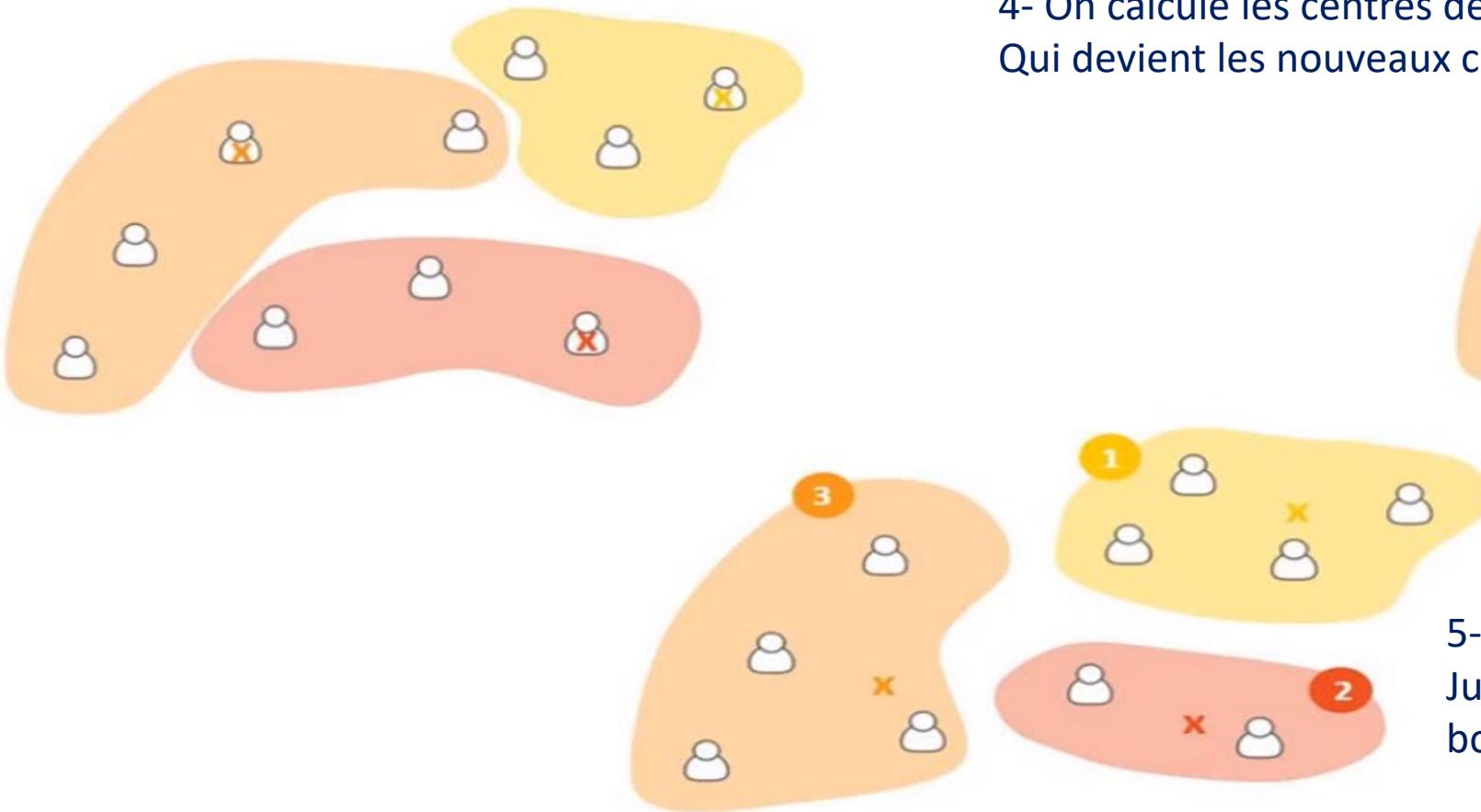


3- Affecter chaque individus au centre le plus proche

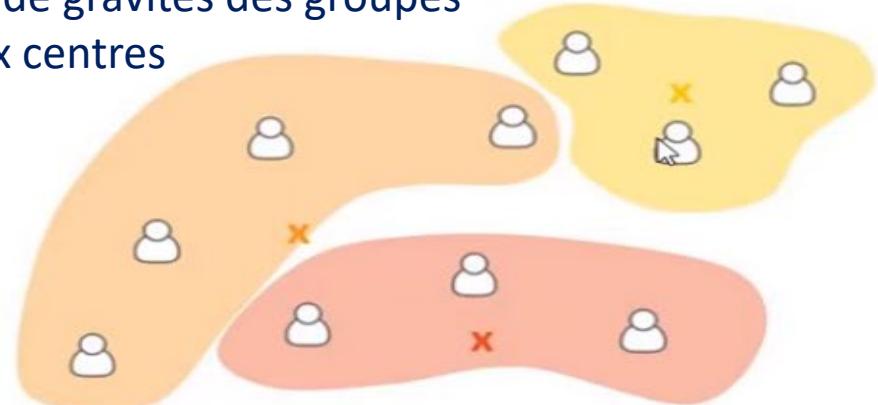
- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Exemple On souhaite créer trois groupes



4- On calcule les centres de gravités des groupes
Qui devient les nouveaux centres



5- On recommence les étapes 2 et 3
Jusqu'à ce que les nouveaux centroids ne bougent plus

- **Algorithmes de Machine learning**

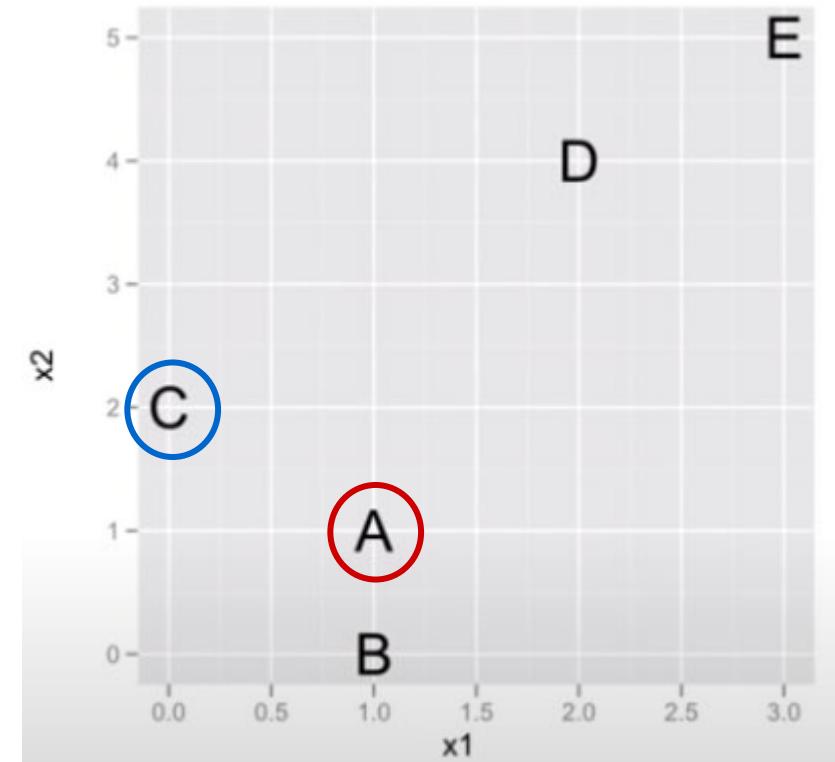
- **K-moyennes ou K-Means**

Exemple 1

Soit la base de données suivantes, on suppose K=2

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

1- On suppose A et C sont choisis au hasard comme centres initiaux



▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 1

Soit la base de données suivantes, on suppose K=2

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

2- Calculer la distance entre chaque centre et chaque points

i	<i>Centre 1</i>	<i>Centre 2</i>
A	$\sqrt{(1 - 1)^2 + (1 - 1)^2} = 0$	$\sqrt{(1 - 0)^2 + (1 - 2)^2} = 1.41$
B	$\sqrt{(1 - 1)^2 + (0 - 1)^2} = 1$	$\sqrt{(1 - 0)^2 + (0 - 2)^2} = 2.23$
C	$\sqrt{(0 - 1)^2 + (2 - 1)^2} = 1.41$	$\sqrt{(0 - 0)^2 + (2 - 2)^2} = 0$
D	$\sqrt{(2 - 1)^2 + (4 - 1)^2} = 3.16$	$\sqrt{(2 - 0)^2 + (4 - 2)^2} = 2.82$
E	$\sqrt{(3 - 1)^2 + (5 - 1)^2} = 4.47$	$\sqrt{(3 - 0)^2 + (5 - 2)^2} = 4.24$

C11

A	1	1
---	---	---

C12

C	0	2
---	---	---

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 1

Soit la base de données suivantes,
on suppose K=2

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

3- Affecter chaque point au groupe
dont il est le plus proche au son centre

i	Centre 1	Centre 2	Cluster
A	$\sqrt{(1 - 1)^2 + (1 - 1)^2} = 0$	$\sqrt{(1 - 0)^2 + (1 - 2)^2} = 1.41$	1
B	$\sqrt{(1 - 1)^2 + (0 - 1)^2} = 1$	$\sqrt{(1 - 0)^2 + (0 - 2)^2} = 2.23$	1
C	$\sqrt{(0 - 1)^2 + (2 - 1)^2} = 1.41$	$\sqrt{(0 - 0)^2 + (2 - 2)^2} = 0$	2
D	$\sqrt{(2 - 1)^2 + (4 - 1)^2} = 3.16$	$\sqrt{(2 - 0)^2 + (4 - 2)^2} = 2.82$	2
E	$\sqrt{(3 - 1)^2 + (5 - 1)^2} = 4.47$	$\sqrt{(3 - 0)^2 + (5 - 2)^2} = 4.24$	2

C11

A	1	1
---	---	---

C12

C	0	2
---	---	---

■ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 1

4- Calculer le centre de gravité de chaque cluster

i	Centre 1	Centre 2	Cluster
A	0	1.41	1
B	1	2.23	1
C	1.41	0	2
D	3.16	2.82	2
E	4.47	4.24	2

i	X ₁	X ₂	Centre de chaque cluster
A	1	1	$(\frac{1+1}{2}, \frac{1+0}{2}) = (1, 0.5)$
B	1	0	
C	0	2	$(\frac{0+2+3}{3}, \frac{2+4+5}{3}) = (1.66, 3.66)$
D	2	4	
E	3	5	

C ₁ 1	1	0.5
------------------	---	-----

C ₂ 2	1.66	3.66
------------------	------	------

Les nouveaux centres sont

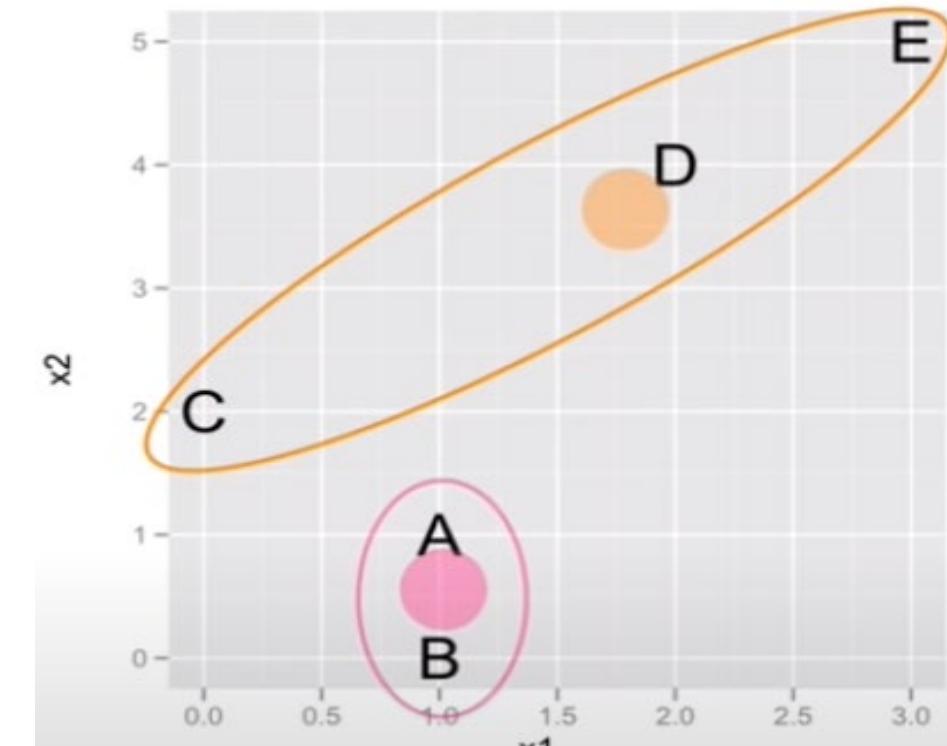
- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Exemple 1

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

Les nouveaux centres sont



C ₁	1	0.5
----------------	---	-----

C ₂	1.66	3.66
----------------	------	------

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 1

Calculer la distance entre chaque centre et chaque points

Affecter chaque point au groupe dont il est le plus proche au son centre

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

Les nouveaux centres sont

i	Centre 1	Centre 2	Cluster
A	$\sqrt{(1 - 1)^2 + (1 - 0.5)^2} = 0.5$	$\sqrt{(1 - 1.66)^2 + (1 - 3.66)^2} = 2.74$	1
B	$\sqrt{(1 - 1)^2 + (0 - 0.5)^2} = 0.5$	$\sqrt{(1 - 1.66)^2 + (0 - 3.66)^2} = 3.71$	1
C	$\sqrt{(0 - 1)^2 + (2 - 0.5)^2} = 1.80$	$\sqrt{(0 - 1.66)^2 + (2 - 3.66)^2} = 2.34$	1
D	$\sqrt{(2 - 1)^2 + (4 - 0.5)^2} = 3.64$	$\sqrt{(2 - 1.66)^2 + (4 - 3.66)^2} = 0.48$	2
E	$\sqrt{(3 - 1)^2 + (5 - 0.5)^2} = 4.92$	$\sqrt{(3 - 1.66)^2 + (5 - 3.66)^2} = 1.89$	2

C21	1	0.5
-----	---	-----

C22	1.66	3.66
-----	------	------

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 1

Calculer le centre de gravité de chaque cluster

i	Centre 1	Centre 2	Cluster
A	0	1.41	1
B	1	2.23	1
C	1.41	0	1
D	3.16	2.82	2
E	4.47	4.24	2

i	X_1	X_2	Centre de chaque cluster
A	1	1	$(\frac{1+1+0}{3}, \frac{1+0+2}{3}) = (0.66, 1)$
B	1	0	
C	0	2	
D	2	4	$(\frac{2+3}{2}, \frac{4+5}{2}) = (2.5, 4.5)$
E	3	5	

C21	0.66	1
-----	------	---

C22	2.5	4.5
-----	-----	-----

Les nouveaux centres sont

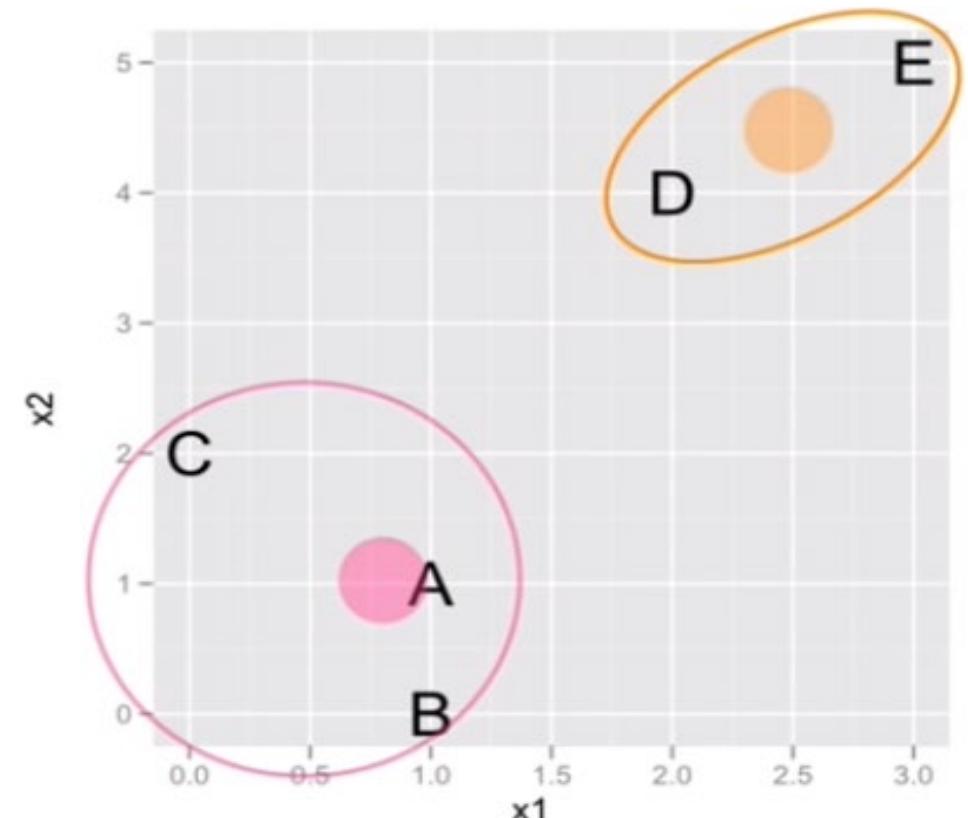
- **Algorithmes de Machine learning**

- **K-moyennes ou K-Means**

Exemple 1

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

Les nouveaux centres sont



C21	0.66	1
-----	------	---

C22	2.5	4.5
-----	-----	-----

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 1

Calculer la distance entre chaque centre et chaque points

Affecter chaque point au groupe dont il est le plus proche au son centre

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

i	Centre 1	Centre 2	Cluster
A	$\sqrt{(1 - 0.66)^2 + (1 - 1)^2} = 0.34$	$\sqrt{(1 - 2.5)^2 + (1 - 4.5)^2} = 3.80$	1
B	$\sqrt{(1 - 0.66)^2 + (0 - 1)^2} = 1.05$	$\sqrt{(1 - 2.5)^2 + (0 - 4.5)^2} = 4.74$	1
C	$\sqrt{(0 - 0.66)^2 + (2 - 1)^2} = 1.19$	$\sqrt{(0 - 2.5)^2 + (2 - 4.5)^2} = 3.53$	1
D	$\sqrt{(2 - 0.66)^2 + (4 - 1)^2} = 3.28$	$\sqrt{(2 - 2.5)^2 + (4 - 4.5)^2} = 0.70$	2
E	$\sqrt{(3 - 0.66)^2 + (5 - 1)^2} = 4.63$	$\sqrt{(3 - 2.5)^2 + (5 - 4.5)^2} = 0.70$	2

Les clusters ne changent pas implique fin de l'algorithme

Cluster 1 (A, B, C)
Cluster 2 (D, E)

C21	0.66	1
-----	------	---

C22	2.5	4.5
-----	-----	-----

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 2

Soit la base de données suivantes, on suppose K=2

1- On suppose 1 et 4 sont choisis au hasard comme centres initiaux

i	X_1	X_2
1	1	1
2	1.5	2
3	3	4
4	5	7
5	3.5	5
6	4.5	5
7	3.5	4.5

C11	1	1
-----	---	---

C12	5	7
-----	---	---

i	centre 1	centre 2	cluster
1	0,00	7,21	1
2	1,12	6,10	1
3	3,61	3,61	1
4	7,21	0,00	2
5	4,72	2,50	2
6	5,32	2,06	2
7	4,30	2,92	2

2- Calculer la distance entre chaque centre et chaque point

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 2

3- Calculer le centre de gravité de chaque cluster

i	X_1	X_2	Cluster	Centre de chaque cluster
1	1	1	1	
2	1.5	2	1	$(\frac{1+1.5+3}{3}, \frac{1+2+4}{3}) = (1.83, 2.33)$
3	3	4	1	
4	5	7	2	
5	3.5	5	2	$(\frac{5+3.5+4.5+3.5}{4}, \frac{7+5+5+4.5}{4}) = (4.13, 5.38)$
6	4.5	5	2	
7	3.5	4.5	2	

Les nouveaux centres sont

C21	1.83	2.33
-----	------	------

C22	4.13	5.38
-----	------	------

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 2

Les nouveaux centres sont

i	X_1	X_2
1	1	1
2	1.5	2
3	3	4
4	5	7
5	3.5	5
6	4.5	5
7	3.5	4.5

C21	1.83	2.33
C22	4.13	5.38

i	centre 1	centre 2	cluster
1	1,57	5,38	1
2	0,47	4,28	1
3	2,03	1,78	2
4	5,64	1,85	2
5	3,14	0,73	2
6	3,77	0,53	2
7	2,73	1,08	2

2- Calculer la distance entre chaque centre et chaque points

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 2

3- Calculer le centre de gravité de chaque cluster

i	X_1	X_2	Cluster	Centre de chaque cluster
1	1	1	1	$(\frac{1+1.5}{2}, \frac{1+2}{2}) = (1.25, 1.50)$
2	1.5	2	1	
3	3	4	1	
4	5	7	2	
5	3.5	5	2	$(\frac{3+5+3.5+4.5+3.5}{5}, \frac{4+7+5+5+4.5}{5}) = (3.90, 5.10)$
6	4.5	5	2	
7	3.5	4.5	2	

Les nouveaux centres sont

C31	1.25	1.50
-----	------	------

C32	3.90	5.10
-----	------	------

▪ Algorithmes de Machine learning

➤ K-moyennes ou K-Means

Exemple 2

i	X_1	X_2
1	1	1
2	1.5	2
3	3	4
4	5	7
5	3.5	5
6	4.5	5
7	3.5	4.5

Les nouveaux centres sont

C21	1.25	1.50
-----	------	------

C22	3.90	5.10
-----	------	------

i	centre 1	centre 2	cluster
1	0,56	5,02	1
2	0,56	3,92	1
3	3,05	1,42	2
4	6,66	2,20	2
5	4,16	0,41	2
6	4,78	0,61	2
7	3,75	0,72	2

2- Calculer la distance entre chaque centre et chaque points

Les clusters ne changent pas implique fin de l'algorithme

Cluster 1 (1, 2)

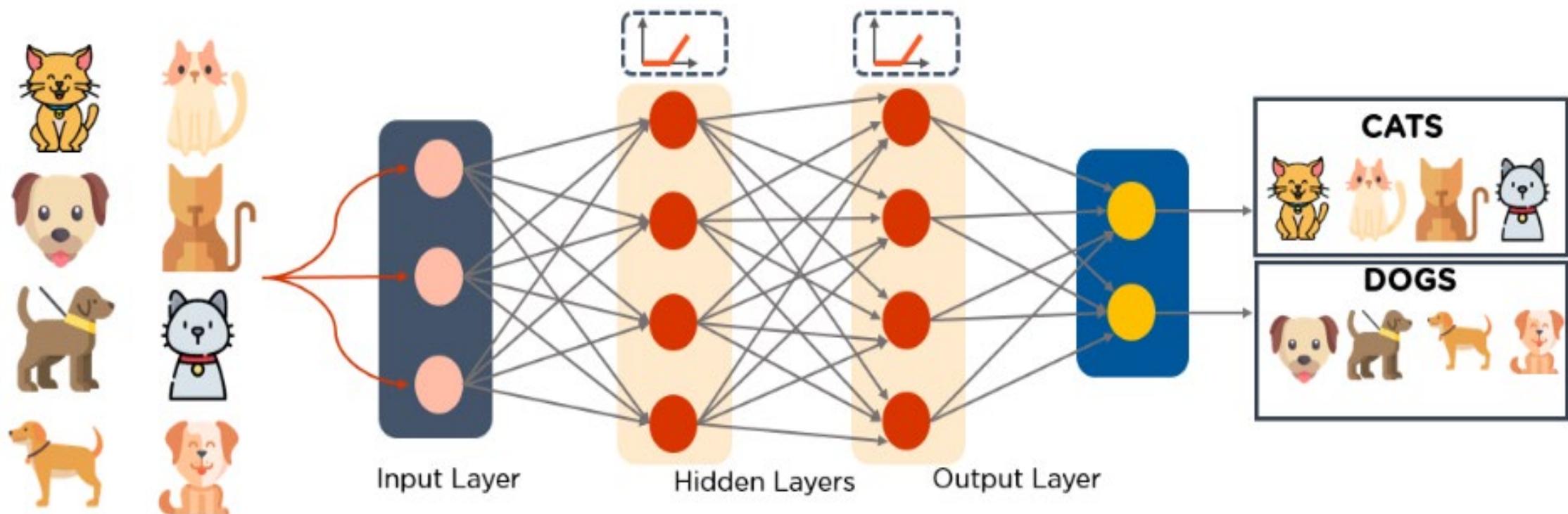
Cluster 2 (3, 4, 5, 6, 7)

Implémentation pratique avec Python et sklearn (TP – K-means)

Deep learning

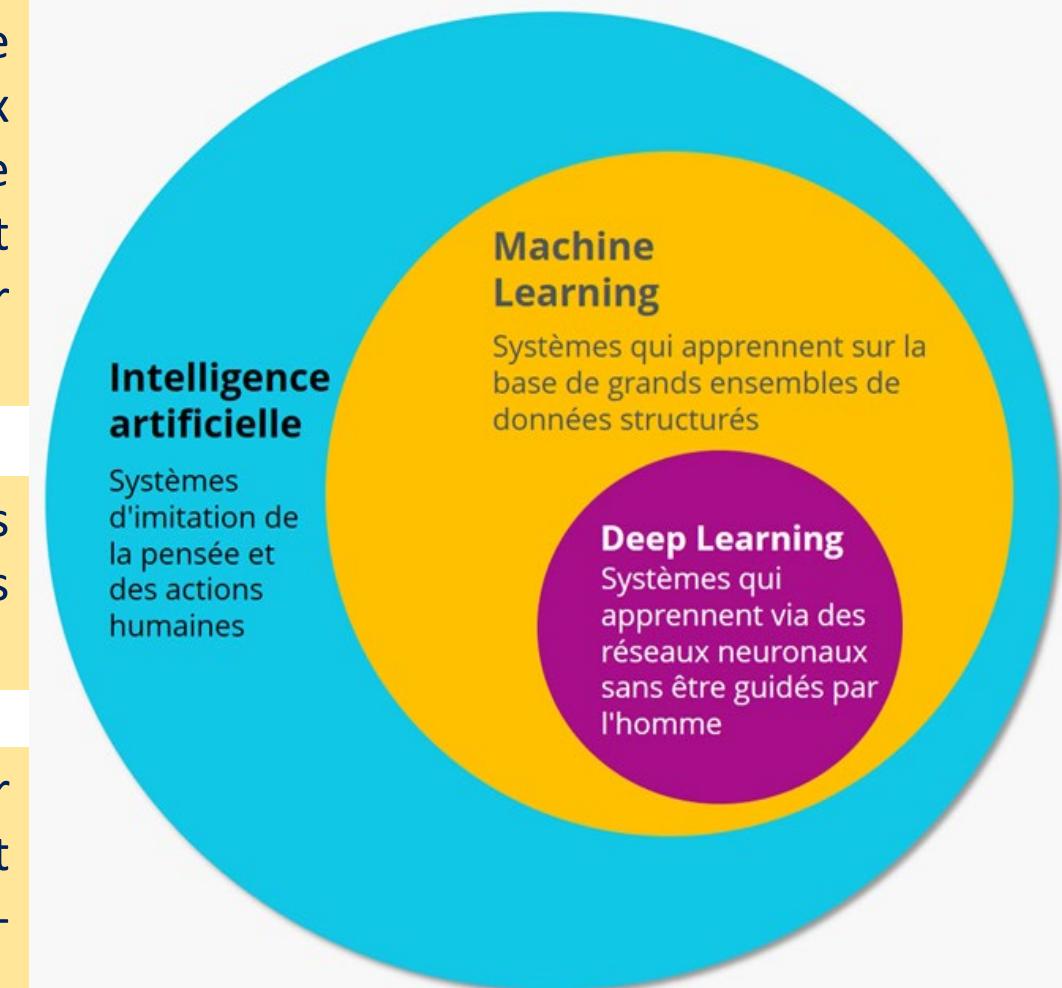
Deep learning (Définition)

- ❖ Deep learning (Apprentissage profond) est un algorithme d'abstraction de haut niveau qui permet de modéliser les données à partir de grands ensembles de données apprises à base de réseaux de neurones artificiels.



Machine learning Vs Deep learning

- Le Machine learning et le Deep learning font partie de l'intelligence artificielle. Ces approches ont toutes deux pour résultat de donner aux ordinateurs la capacité prendre des décisions intelligentes. Cependant, le Deep learning est une sous-catégorie du Machine learning, car il s'appuie sur un apprentissage sans surveillance.
- Machine learning utilise des algorithmes pour analyser les données, apprendre de ces données et prendre des décisions éclairées en fonction de ce qu'il a appris.
- Le Deep learning structure des algorithmes en couches pour créer un « réseau de neurones artificiels » qui peut apprendre et prendre des décisions intelligentes par lui-même.



Différences majeures

1-La mise en place d'une solution basée sur ML implique l'existence de données structurées. Le système est ensuite alimenté par des données structurées et catégorisées lui permettant de comprendre comment classer de nouvelles données similaires alors le DL n'a pas besoin de données structurées. Le système fonctionne à partir de plusieurs couches de réseaux neuronaux, qui combinent différents algorithmes en s'inspirant du cerveau humain.

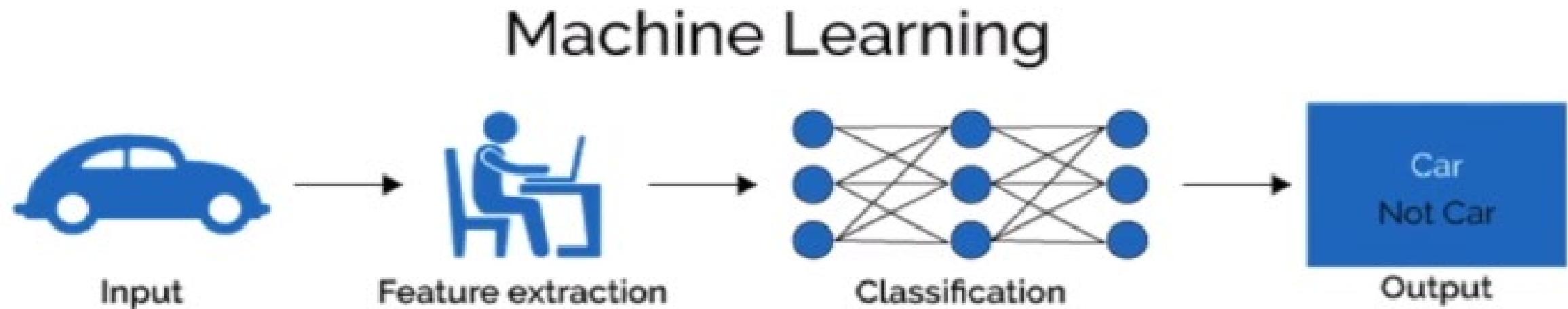
2-Le ML fonctionne à partir d'une base de données contrôlable, alors que le DL a besoin d'un volume de données bien plus considérable. Le système doit disposer de plus de données d'entrées pour obtenir des résultats fiables. Par ailleurs, le DL nécessite une technologies plus sophistiquée. Elle exige plus de ressources IT et s'avère nettement plus coûteuse que le ML.

Différences majeures

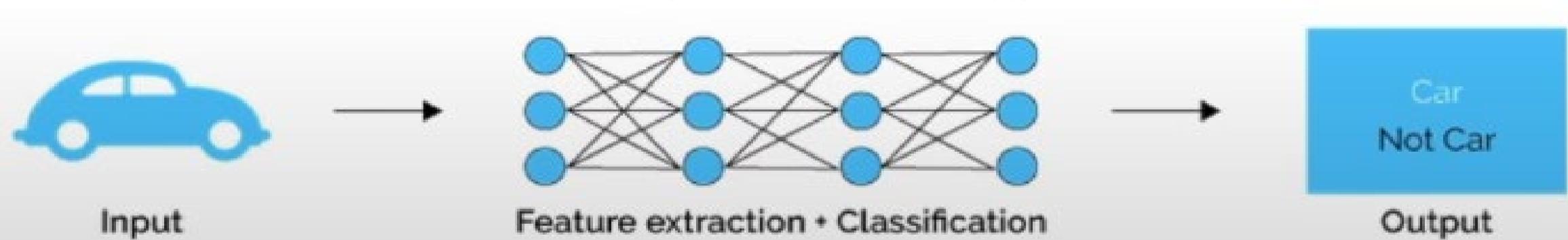
3- Avec le DL, le système identifie lui-même les caractéristiques discriminantes de données, sans avoir d'une catégorisation préalable. Le système n'a pas besoin d'être entraîné par un développeur. Il évalue lui-même le besoin de modifier le classement ou de créer des catégories inédites en fonction des nouvelles données.

4-Si un algorithme de ML renvoie une prédiction inexacte, une intervention est nécessaire pour effectuer des ajustements. Avec un modèle de DL, un algorithme peut déterminer par lui-même si une prédiction est précise ou non via son propre réseau de neurones.

Différences majeures



Deep Learning



Machine Learning

Régression linéaire

Régression logistique

Les arbres de décision

Les K-plus proche voisins

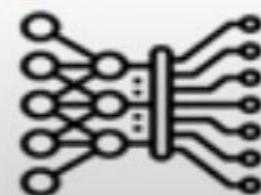
K-means

SVM

ACP

...

Deep Learning



Réseau de neurones

Différences majeures



- Petites quantités de données.
- Puissance de calcul normale..
- Caractéristiques créées par les utilisateurs.
- Peu de temps.

Données



Matériels



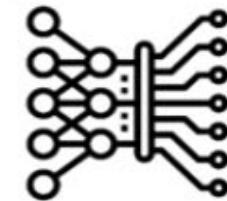
Personnalisation



Temps d'exécution



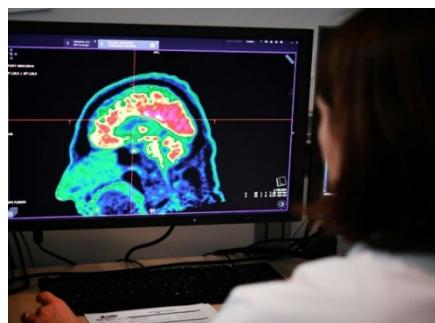
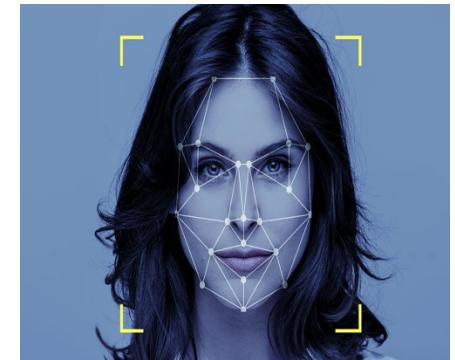
- Grandes quantités de données.
- Puissance de calcul élevée.
- Caractéristiques de façon autonome.
- Temps assez long.



Domaine d'application

❖ Le Deep Learning est utilisé dans de nombreux domaines:

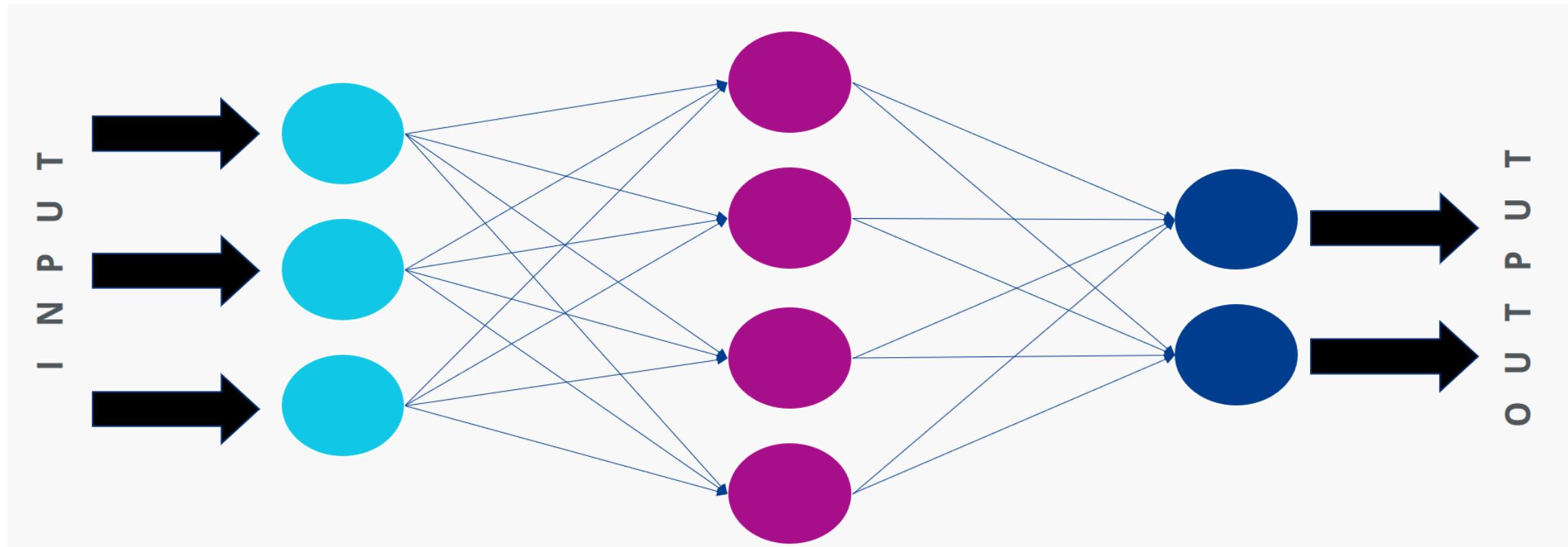
- ✓ Reconnaissance faciale et d'image,
- ✓ Traduction automatique,
- ✓ Voiture autonome,
- ✓ Diagnostic médical,
- ✓ Recommandations personnalisées,
- ✓ Modération automatique des réseaux sociaux,
- ✓ Prédiction financière et trading automatisé,
- ✓ Identification de pièces défectueuses,
- ✓ Détection de fraudes,
- ✓ Chatbots (agents conversationnels),
- ✓ Robots intelligents.



FRAUD DETECTION

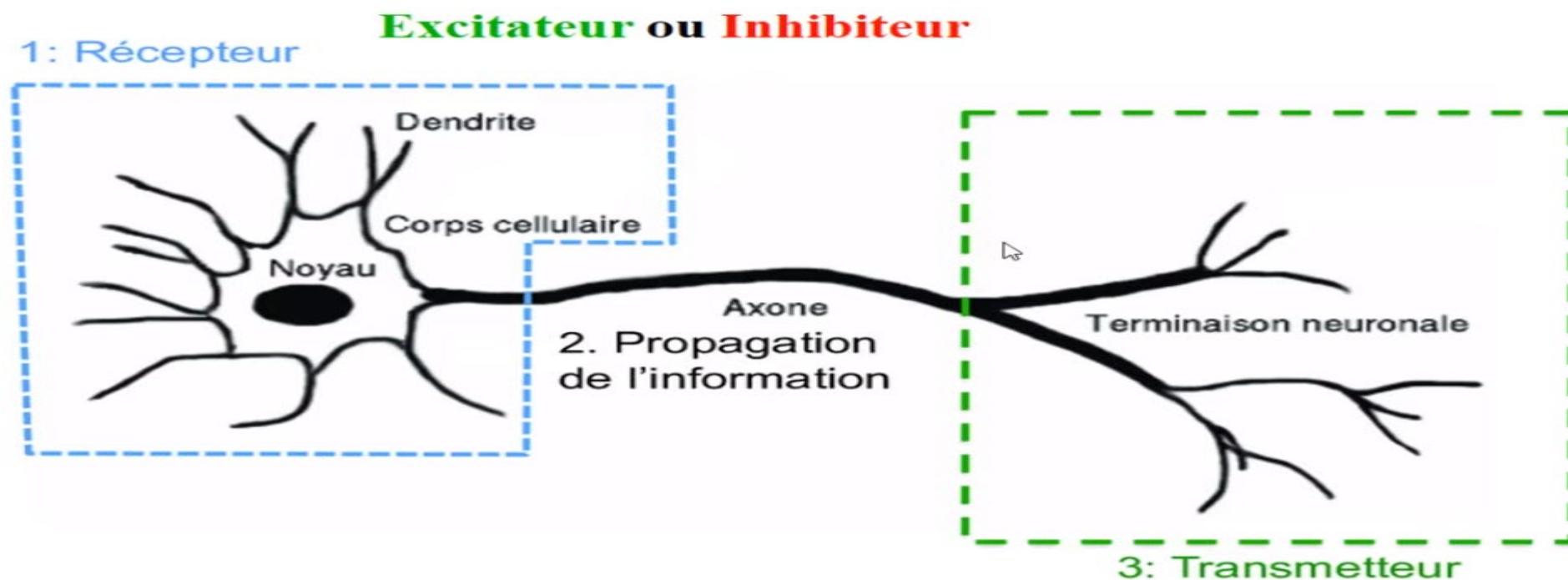
Réseau de neurones artificiels

Les réseaux de neurones artificiels (RNA) est un algorithme qui s'inspire de la biologie et plus particulièrement du fonctionnement du cerveau humain. Il s'agit de neurones en activités, ces neurones sont reliés entre eux par des liens de connexion et ont un poids. Ils se multiplient avec le signal transmis dans le réseau. La sortie de chaque neurone est déterminée en utilisant une fonction d'activation.



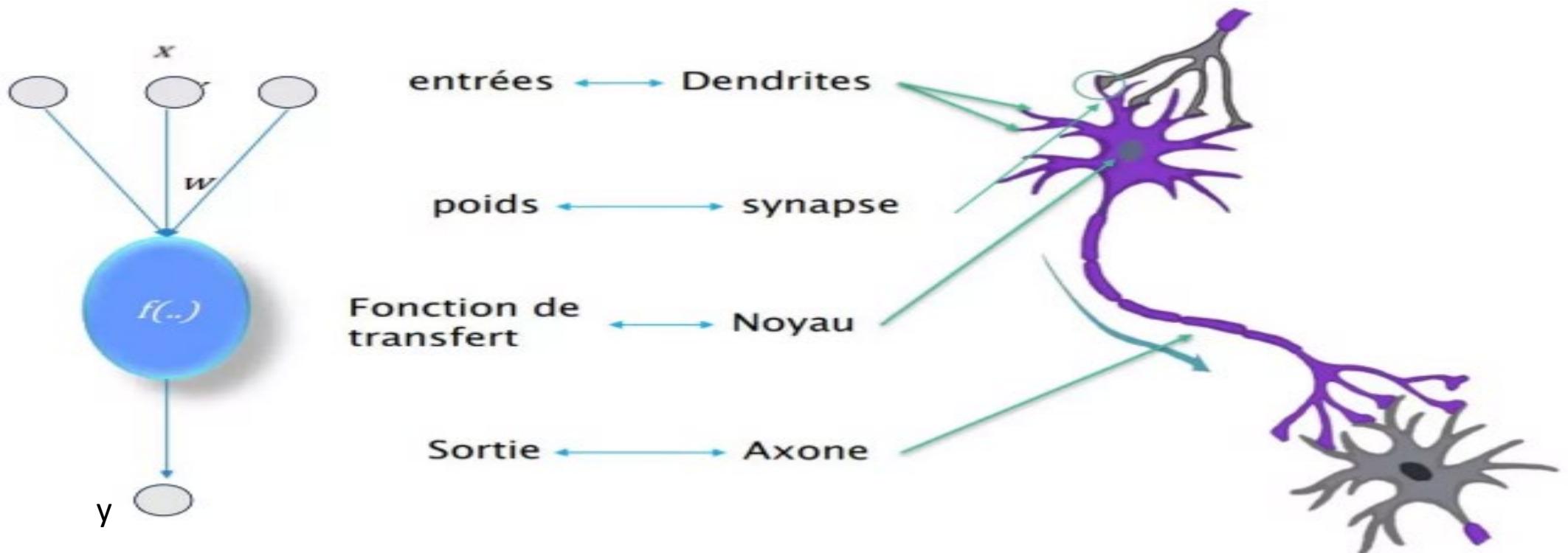
Réseau de neurones artificiels

Les neurones sont des cellules excitables, connectées les une aux autres et ayant un rôle de transmettre des informations dans notre système nerveux, chaque neurone se compose de plusieurs dendrites, d'un corps cellulaire et d'un axone. Les dendrites reçoivent au niveau des synapses des signaux des neurones qui le précèdent, des signaux de type Exciteur ou Inhibiteur, lorsque la somme de ces signaux dépasse un seuil, le neurone s'active (Activation) et il produit un signal électrique qui circule au long de l'Axone en directions des terminaison pour être envoyé à son tour vers d'autres neurones connectés.



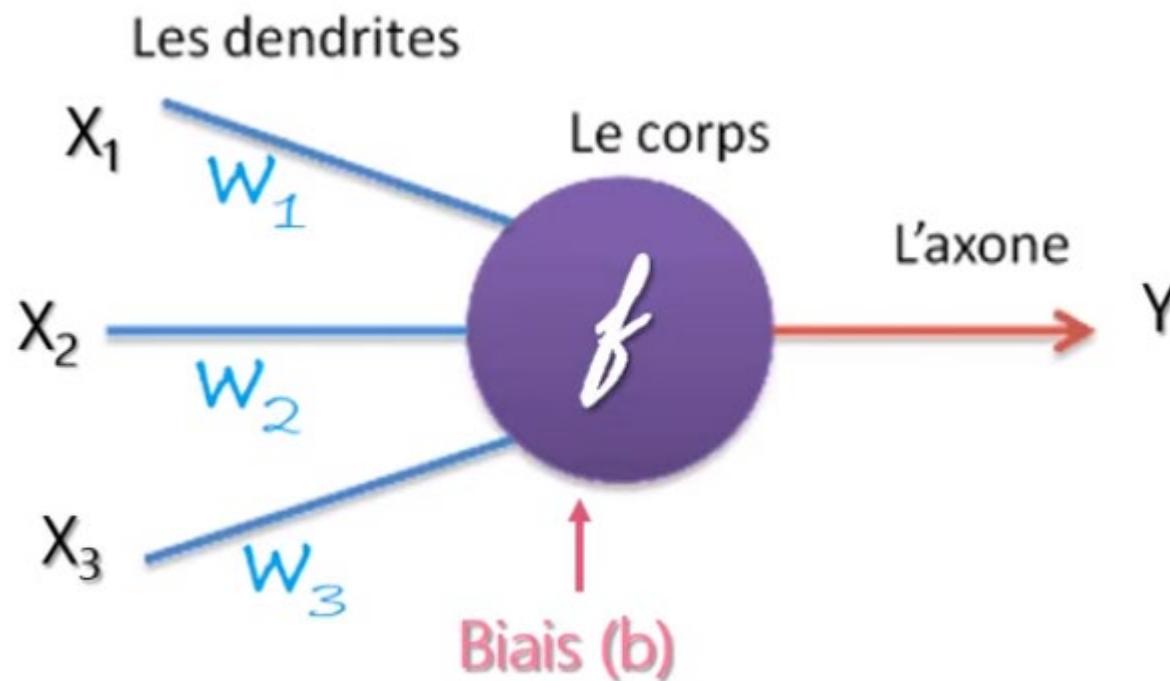
Réseau de neurones artificiels

A la suite de cette observation biologique, les scientifiques ont schématiser un neurone en quatre parties, les dendrites, synapse, noyau, axone



Réseau de neurones artificiels

Pour bien illustrer ce fonctionnement on le schématisé comme suit: x_1, x_2, x_3 présentent les entrées d'une fonction dite de transfert et qui retourne une sortie y qui permet d'activer ou non le neurone selon la force des signaux (Exciteur ou inhibiteur) suivant un traitement à effectuer au niveau de la fonction F . La fonction F se compose de deux phases consécutives, la première et l'**agrégation** ou la somme. La deuxième est l'**activation**.



Agrégation

$$z = w_1x_1 + w_2x_2 + w_3x_3 + b$$

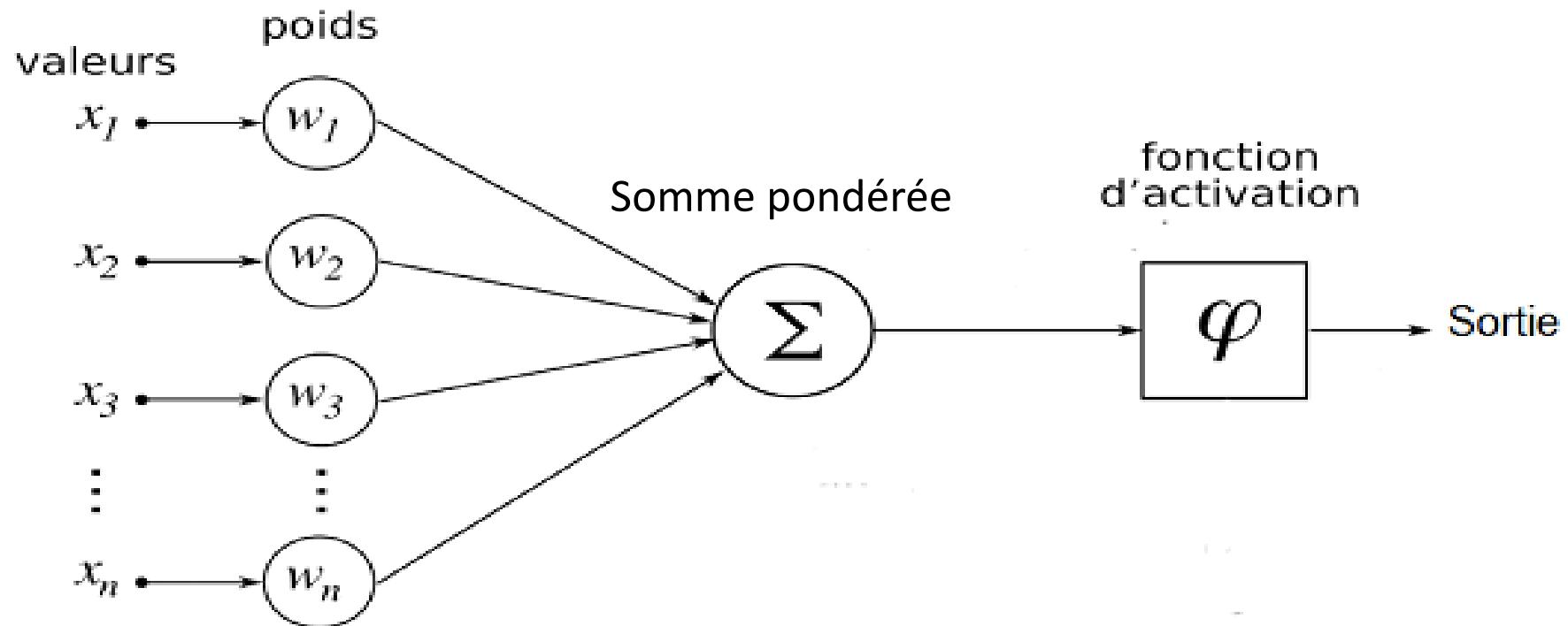
Activation

$$\begin{aligned} a &= 1 && \text{si } z \geq \text{seuil} \\ a &= 0 && \text{Sinon} \end{aligned}$$

$$y = a(w_1x_1 + w_2x_2 + w_3x_3 + b) = a(z)$$

Réseau de neurones artificiels

Modèle de réseaux de neurones artificiels

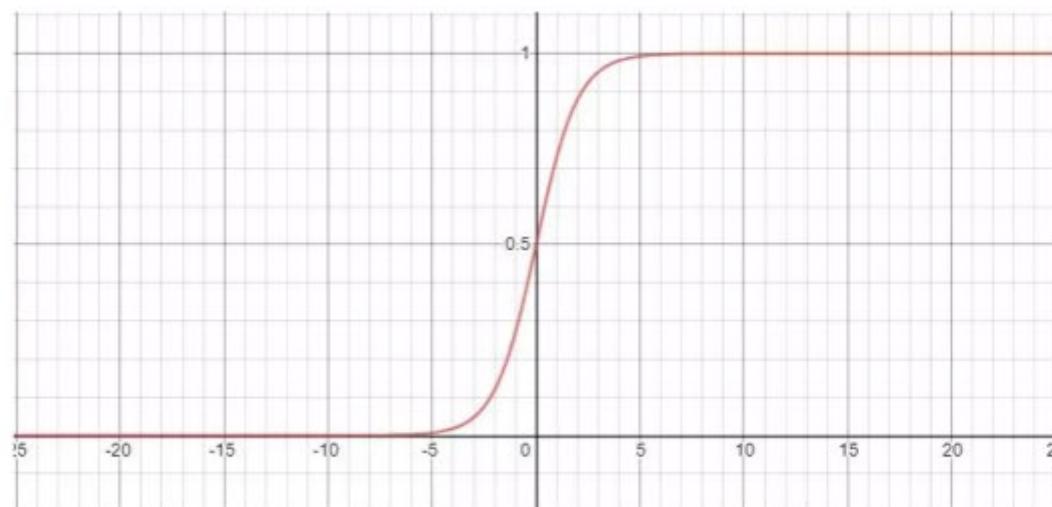


Fonctions d'activation

Sigmoid / Logistique

$$S : \mathbb{R} \rightarrow [0, 1]$$

$$x \rightarrow \frac{1}{1 + e^{-x}}$$



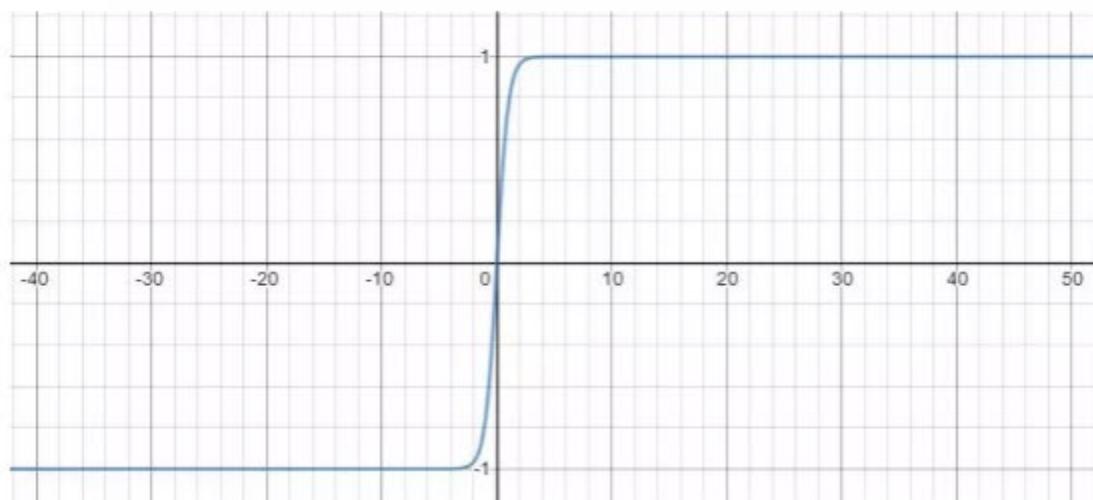
- Réduit la valeur de pré-activation des neurones entre 0 et 1
- Toujours Positive
- Bornée

Fonctions d'activation

TanH / tangente hyperbolique

$$\text{Tanh} : \mathfrak{R} \rightarrow [-1, 1]$$

$$x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}} - 1$$



- Réduit la valeur de pré-activation des neurones entre -1 et 1
- Sortie centrée en 0
- Bornée

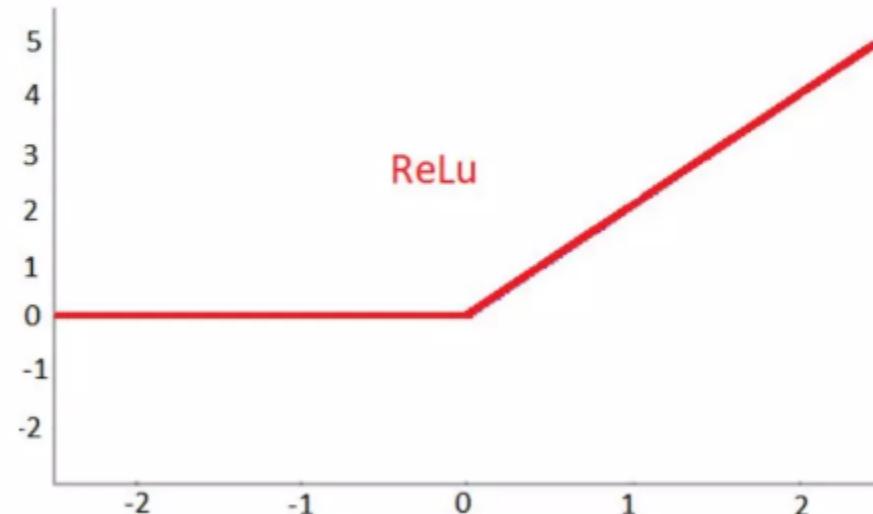
Fonctions d'activation

ReLU / Rectified Linear Unit

Il s'agit de l'une des fonctions les plus puissantes et les plus utiles

- Elle est rapide → simple à calculer
- Elle casse la linéarité des valeurs négatives
- Elle désactive certains neurones

$$f(x) = \max(0, x)$$



Fonctions d'activation

Softmax

Soit :

$$\vec{x} = [x_1, x_2, \dots, x_n]$$

On a : $f_i(\vec{x}) : \Re \rightarrow [0, 1]$

$$\vec{x} \rightarrow \frac{e^{x_i}}{\sum_k e^{x_k}}$$

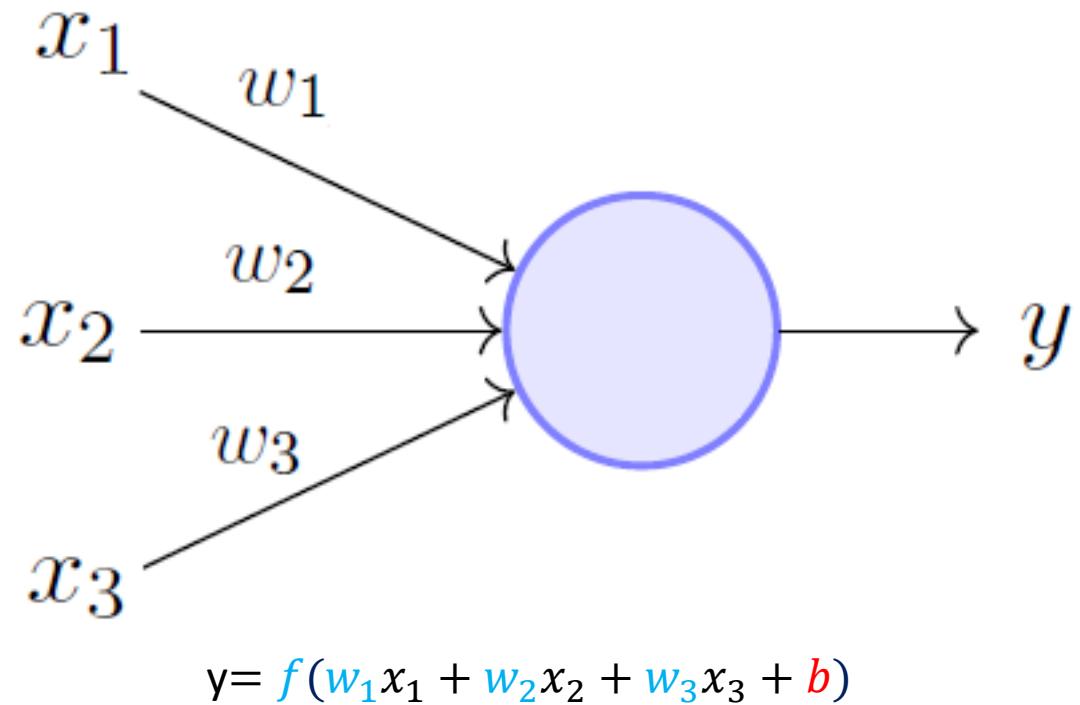
- Elle prend en argument tout le vecteur.
- Réduit la valeur de pré-activation des neurones entre 0 et 1.
- La somme des résultats fournis est toujours égale à 1.

Probabilités

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

Un réseau à propagation avant à couche unique (Perceptron mono-couche)



1957



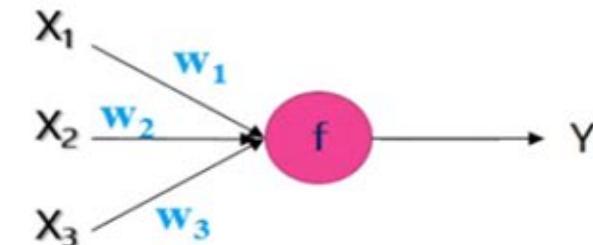
Frank Rosenblatt

Il dispose d'un algorithme d'apprentissage pour déterminer les poids w

Réseau de neurones artificiels

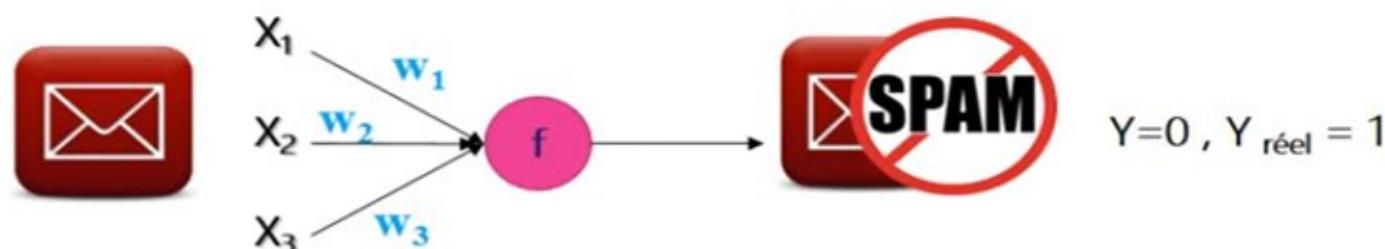
Perceptron mono-couche (Simple Layer Perceptron)

Formule de mise à jour des poids



$$\mathbf{W} = \mathbf{W} + \alpha (y_{\text{réel}} - y) \mathbf{X}$$

Sortie de référence
Vitesse d'apprentissage
Entrée du neurone
Sortie produite par le neurone



Si $y_{\text{réel}} = 1$ et $y = 0$ alors

$$\mathbf{W} = \mathbf{W} + \alpha \times$$

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

Algorithme :

α est une constante positive. // Taux d'apprentissage

Initialiser les poids $w_i = 0$ // w_0 correspond à b (biais)

Répéter : // jusqu'à nb. itérations max. ou nb. erreurs est 0

- Prendre un exemple (x, t) dans D
- Calculer la sortie o du réseau pour l'entrée x : $a(Y)$
- Si $t \neq o$ Alors // Mise à jour des poids w_i et biais

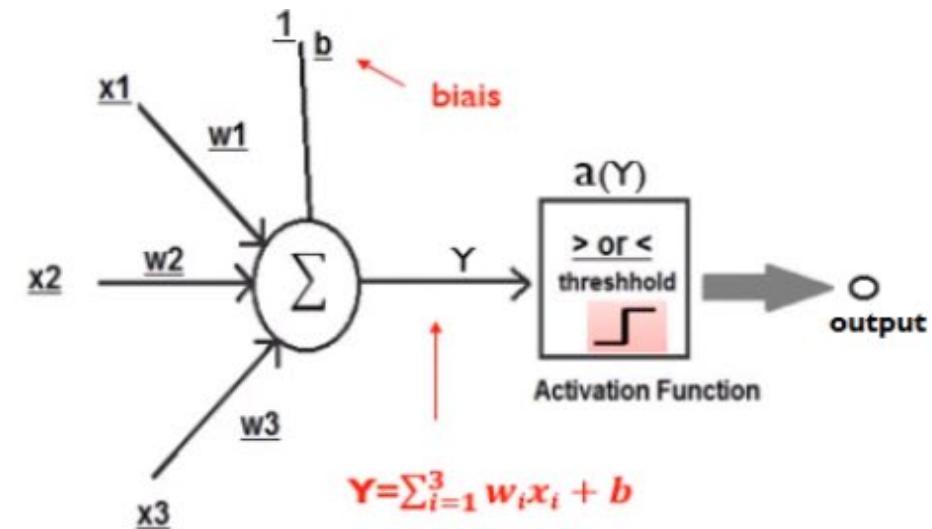
Pour i de 0 à n :

$w_i = w_i + \alpha * (t - a(y)) * x_i$ // Règle d'apprentissage

FinPour

FinSi

FinRépéter



Dataset (données d'entraînement)

D	x	x_1	x_2	x_3	$t(x)$
$\mathbf{x}^{(1)}$	0	0	0	0	+1
$\mathbf{x}^{(2)}$	0	1	0	0	+1
$\mathbf{x}^{(3)}$	1.5	0	-1.5	0	+1
$\mathbf{x}^{(4)}$	1.5	1	-1.5	0	+1
$\mathbf{x}^{(5)}$	1.5	0	0	0	-1
$\mathbf{x}^{(6)}$	1.5	1	0	0	-1
$\mathbf{x}^{(7)}$	0	0	-1.5	0	-1
$\mathbf{x}^{(8)}$	0	1	-1.5	0	-1

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

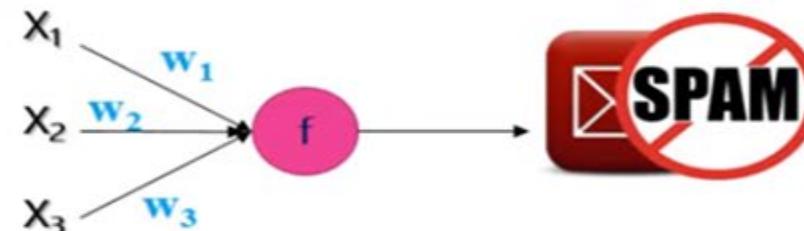
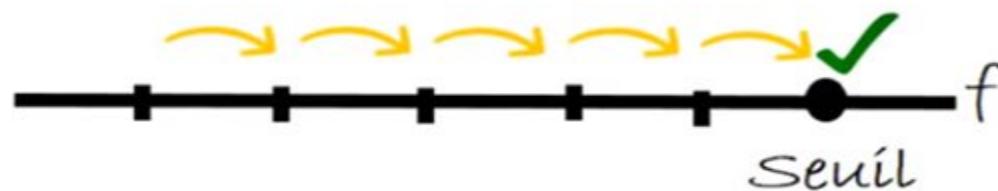
$$w = w + \alpha (y_{\text{réel}} - y) \times$$

$$w = w + \alpha \times$$

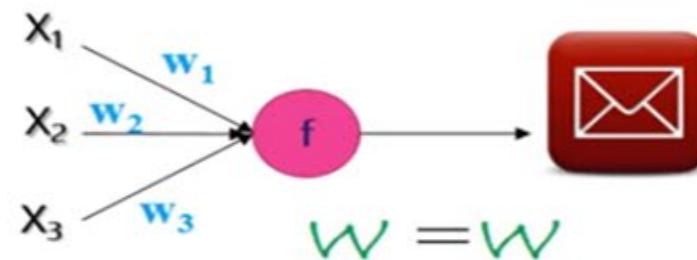
$$x_1 = 1 \rightarrow w_1 = w_1 + \alpha$$

$$x_2 = 0 \rightarrow w_2 = w_2$$

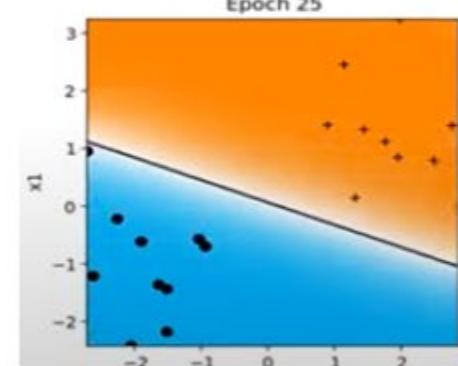
$$x_3 = 0 \rightarrow w_3 = w_3$$



$Y=0, Y_{\text{réel}}=1$



$Y=1, Y_{\text{réel}}=1$



Perceptron mono-couche pour la classification des données linéairement séparables

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

Exemple 1

Soit la base de données suivante :

No	x1	x2	y
1	1	1	1
2	2	0.4	0
3	0	1	1

$$On\ donne: x_0 = 1, \quad w_0 = 0, \quad w_1 = 0, \quad w_2 = 0,$$

$$b = 0, \quad \alpha = 0.1$$

Agrégation

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b$$

Activation

$$a = 0 \quad si \ z \leq 0$$

$$a = 1 \quad Sinon$$

$$y = a(w_0 x_0 + w_1 x_1 + w_2 x_2 + b) = a(z)$$

$$w = w + \alpha(y - a(z)) * x$$

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

Exemple 1

On donne: $x_0 = 1, w_0 = 0, w_1 = 0, w_2 = 0, b = 0, \alpha = 0.1$

No	x1	x2	y
1	1	1	1
2	2	0.4	0
3	0	1	1

Etape 1: entrée (1)

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b = 0 * 1 + 0 * 1 + 0 * 1 + 0 = 0$$

$$a(z) = 0 \neq (y = 1) \rightarrow \text{modification des points}$$

$$w_0 = 0 + 0.1(1 - 0) * 1 = 0.1$$

$$w_1 = 0 + 0.1(1 - 0) * 1 = 0.1$$

$$w_2 = 0 + 0.1(1 - 0) * 1 = 0.1$$

Agrégation

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b$$

Activation

$$a = 0 \quad \text{si } z \leq 0$$

$$a = 1 \quad \text{Sinon}$$

$$y = a(w_0 x_0 + w_1 x_1 + w_2 x_2 + b) = a(z)$$

$$w = w + \alpha(y - a(z)) * x$$

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

Exemple 1

On donne: $x_0 = 1, w_0 = 0.1, w_1 = 0.1, w_2 = 0.1, b = 0, \alpha = 0.1$

No	x1	x2	y
1	1	1	1
2	2	0.4	0
3	0	1	1

Agrégation

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b$$

Activation

$$a = 0 \quad \text{si } z \leq 0$$

$$a = 1 \quad \text{Sinon}$$

Etape 2: entrée (2)

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b = 0.1 * 1 + 0.1 * 2 + 0.1 * 0.4 = 0.34$$

$$a(z) = 1 \neq (y = 0) \rightarrow \text{modification des points}$$

$$y = a(w_0 x_0 + w_1 x_1 + w_2 x_2 + b) = a(z)$$

$$w_0 = 0.1 + 0.1(0 - 1) * 1 = 0$$

$$w = w + \alpha(y - a(z)) * x$$

$$w_1 = 0.1 + 0.1(0 - 1) * 2 = -0.1$$

$$w_2 = 0.1 + 0.1(0 - 1) * 0.4 = 0.06$$

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

Exemple 1

On donne: $x_0 = 1, w_0 = 0, w_1 = -0.1, w_2 = 0.06, b = 0, \alpha = 0.1$

No	x1	x2	Y
1	1	1	1
2	2	0.4	0
3	0	1	1

Etape 3: entrée (3)

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b = 0 * 1 - 0.1 * 0 + 0.06 * 1 = 0.06$$

$$a(z) = 1 = (y = 1) \rightarrow \text{Pas de modification des points}$$

$$y = a(0 * x_0 - 0.1 * x_1 + 0.06 * x_2 + 0) = a(z)$$

Agrégation

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b$$

Activation

$$a = 0 \quad \text{si } z \leq 0$$

$$a = 1 \quad \text{Sinon}$$

$$y = a(w_0 x_0 + w_1 x_1 + w_2 x_2 + b) = a(z)$$

$$w = w + \alpha(y - a(z)) * x$$

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

Exemple 2

On donne: $x_0 = 1, w_0 = 0, w_1 = 1, w_2 = -1, b = 0, \alpha = 1$

No	x1	x2	Y (OR)
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

Etape 1: entrée (1)

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b = 0 * 1 + 1 * 0 - 1 * 0 + 0 = 0$$

$$a(z) = 0 \text{ et } (y = 0) \rightarrow \text{Pas de modification des points}$$

Etape 2: entrée (2)

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b = 0 * 1 + 1 * 0 - 1 * 1 = -1$$

$$a(z) = 0 \text{ et } (y = 1) \rightarrow \text{modification des points}$$

$$w_0 = 0 + 1(1 - 0) * 1 = 1$$

$$w_1 = 1 + 1(1 - 0) * 0 = 1$$

$$w_2 = -1 + 1(1 - 0) * 1 = 0$$

Agrégation

$$z = \sum w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + b$$

Activation

$$a = 0 \quad \text{si } z \leq 0$$

$$a = 1 \quad \text{Sinon}$$

$$y = a(w_0 x_0 + w_1 x_1 + w_2 x_2 + b) = a(z)$$

$$w = w + \alpha(y - a(z)) * x$$

Réseau de neurones artificiels

Perceptron mono-couche (Simple Layer Perceptron)

Exemple 2

De même et pour 10 itérations on trouve les résultats suivants :

Étape	w_0	w_1	w_2	Entrée	$\sum_{i=0}^2 w_i x_i$	$a(z)$	y	w_0	w_1	w_2
init								0	1	-1
1	0	1	-1	100	0	0	0	$0+0 \times 1$	$1+0 \times 0$	$-1+0 \times 0$
2	0	1	-1	101	-1	0	1	$0+1 \times 1$	$1+1 \times 0$	$-1+1 \times 1$
3	1	1	0	110	2	1	1	1	1	0
4	1	1	0	111	2	1	1	1	1	0
5	1	1	0	100	1	1	0	$1+(-1) \times 1$	$1+(-1) \times 0$	$0+(-1) \times 0$
6	0	1	0	101	0	0	1	$0+1 \times 1$	$1+1 \times 0$	$0+1 \times 1$
7	1	1	1	110	2	1	1	1	1	1
8	1	1	1	111	3	1	1	1	1	1
9	1	1	1	100	1	1	0	$1+(-1) \times 1$	$1+(-1) \times 0$	$1+(-1) \times 0$
10	0	1	1	101	1	1	1	0	1	1

Donc : $w_0 = 0$; $w_1 = 1$; $w_2 = 1$

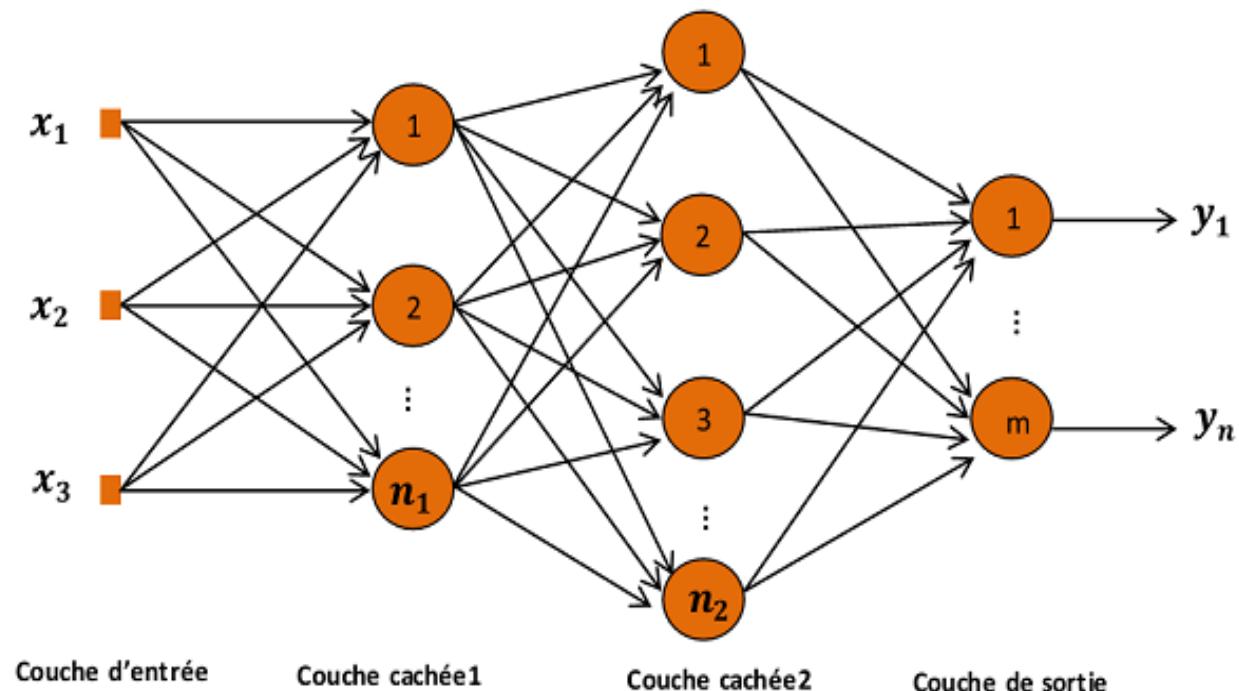
Ce perceptron calcule le OU logique pour tout couple $(x_1 ; x_2)$

Algorithmes de deep learning

Réseau de neurones artificiels

Perceptron multi-couche (Multi Layer Perceptron)

1957



Geoffrey Hinton

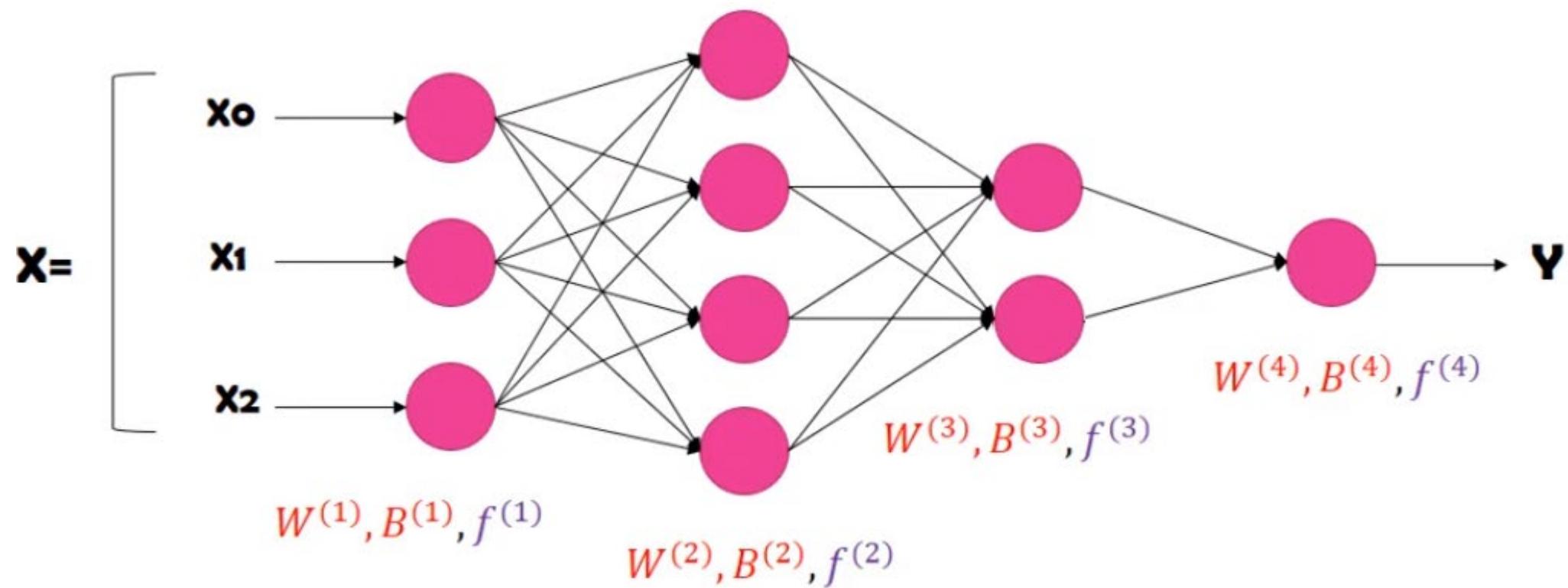
Plus les couches cachées sont nombreuses plus le réseau est **profond** d'où l'appellation **apprentissage profond** ou **Deep Learning**

Perceptron multi-couche pour la classification des données non linéairement séparables

Algorithmes de deep learning

Réseau de neurones artificiels

Perceptron multi-couche (Multi Layer Perceptron)

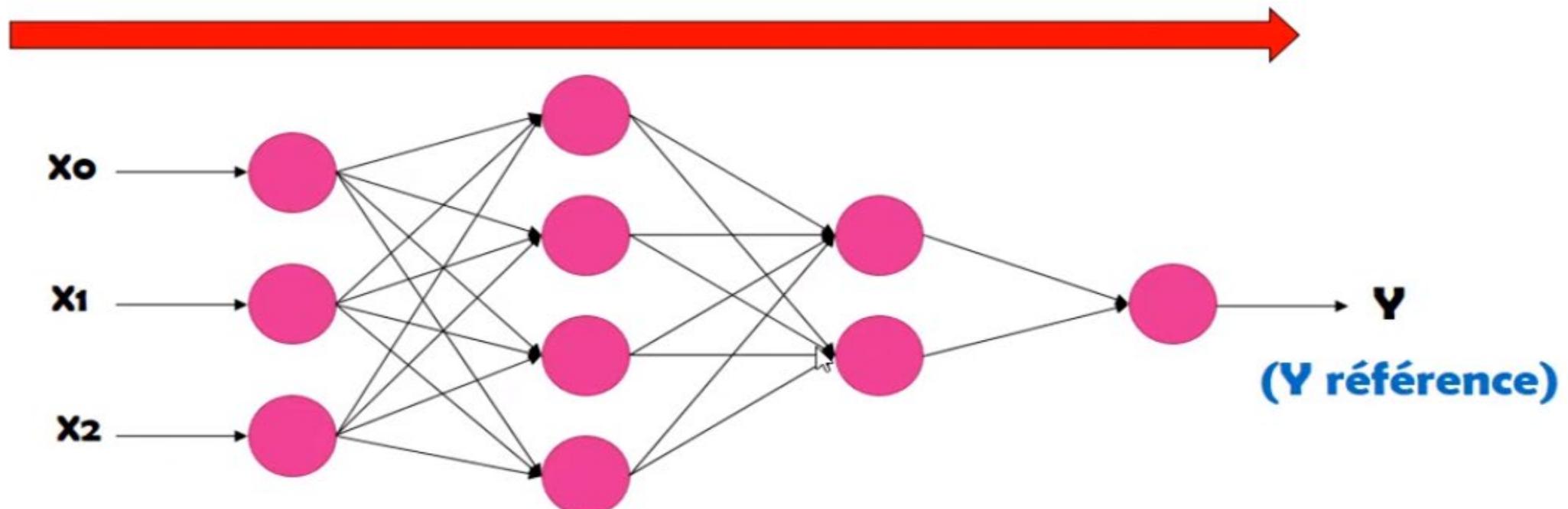


Algorithmes de deep learning

Réseau de neurones artificiels

Perceptron multi-couche (Multi Layer Perceptron)

Algorithm



1.

**Forward
propagation**

2.

**Calculer
l'erreur**

3.

**Back
propagation**

4.

**Descente de
Gradient**

Implémentation pratique avec Python et sklearn (TP – MLP)

Réseaux de neurones convolutifs

Une image n'est qu'une représentation analogique d'un être ou d'une chose qu'on cherche souvent à traiter

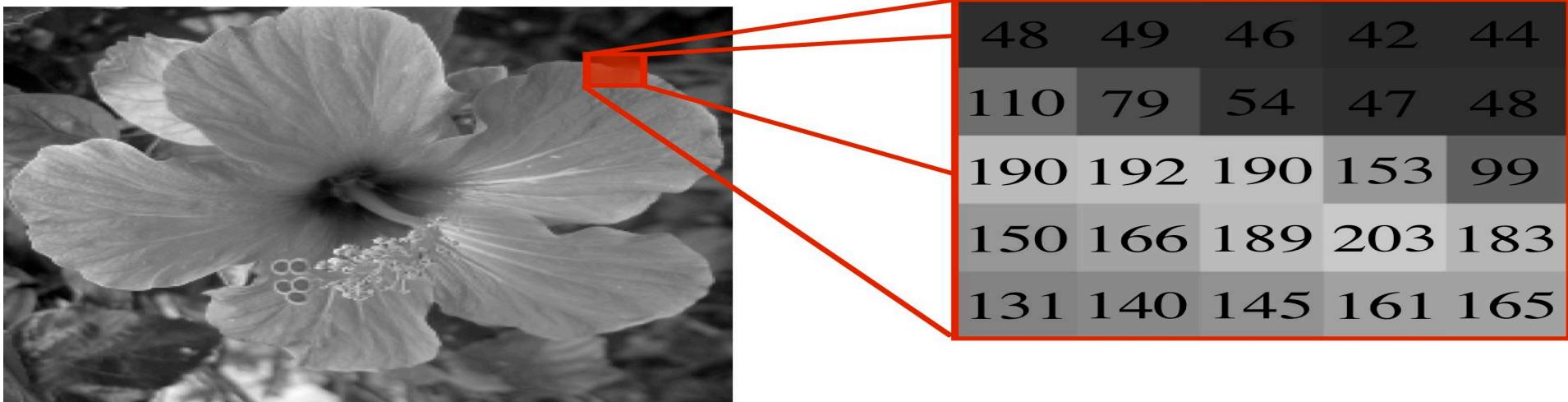
Le traitement d'image s'effectue pour plusieurs raisons, entre autre:

- Pour améliorer la qualité de l'image
- Pour détecter quelques formes, certains contours ou certaines textures de modèle connu
- Pour réduire les informations contenues dans une image

Aussi, le traitement d'image concerne l'analyse de l'image qui:

Cherche à extraire des informations contenues dans les divers objets

Les techniques de base utilisées sont essentiellement l'extraction d'attributs et la segmentation de l'image en zone homogène



Algorithmes de deep learning

Réseaux de neurones convolutifs

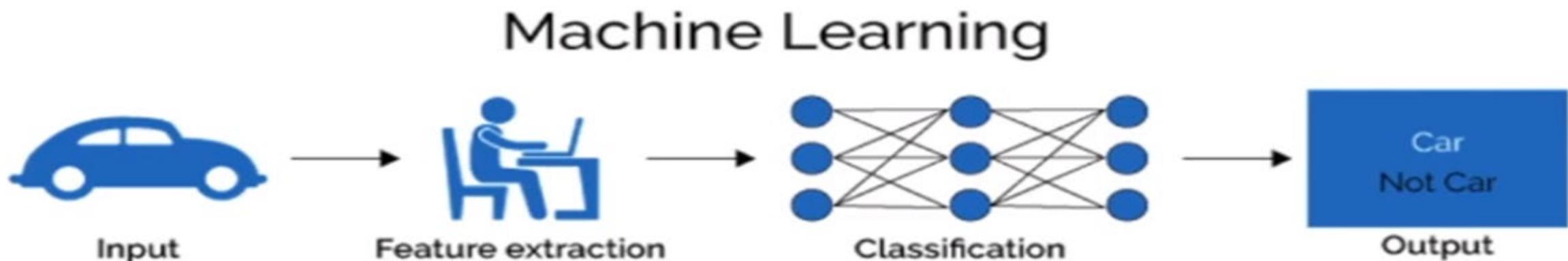
La classification d'une image en utilisant un modèle de réseaux de neurones suit les phases suivantes :

Phase d'extraction des caractéristiques de l'image (Features extraction):

Phase d'apprentissage permet d'ajuster les paramètres de réseau en minimisant l'erreur entre la sortie désirée et la sortie calculée (méthode de descente de gradient)

Phase de généralisation c'est la phase de classification

Inconvénient : le résultat de la classification dépend de la qualité des caractéristiques présentés à l'entrée du réseau est un problème fondamental en vision par ordinateur



Algorithmes de deep learning

Réseaux de neurones convolutifs



64*64*3

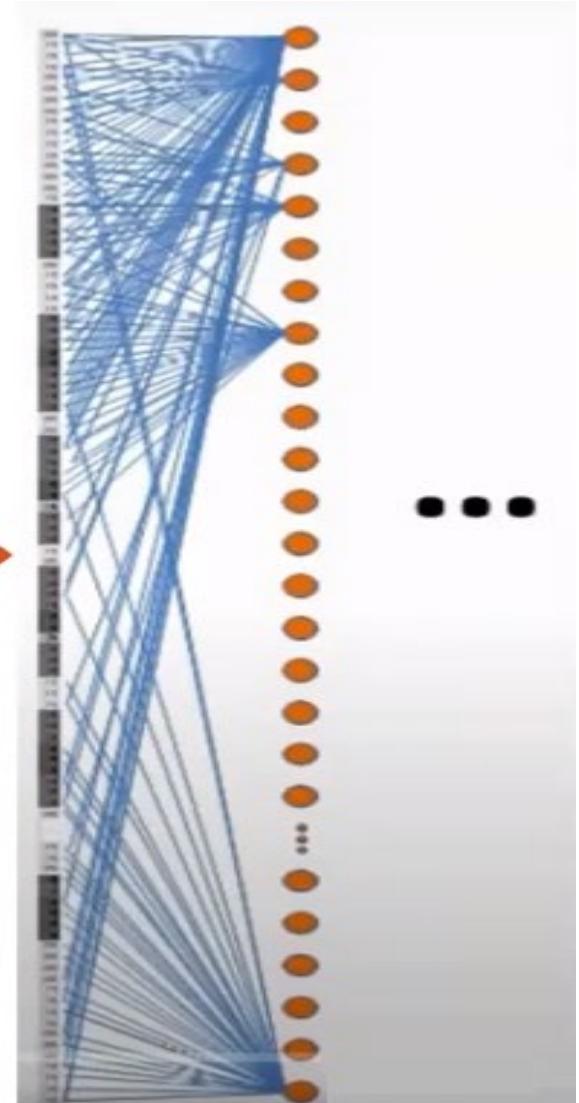
→ Cat? (0/1)

12228 entrées



1000*1000*3

3 Million entrées



...

- Perte de la sémantique de l'image lors de sa transformation en vecteur
- Fléau de la dimensionnalité

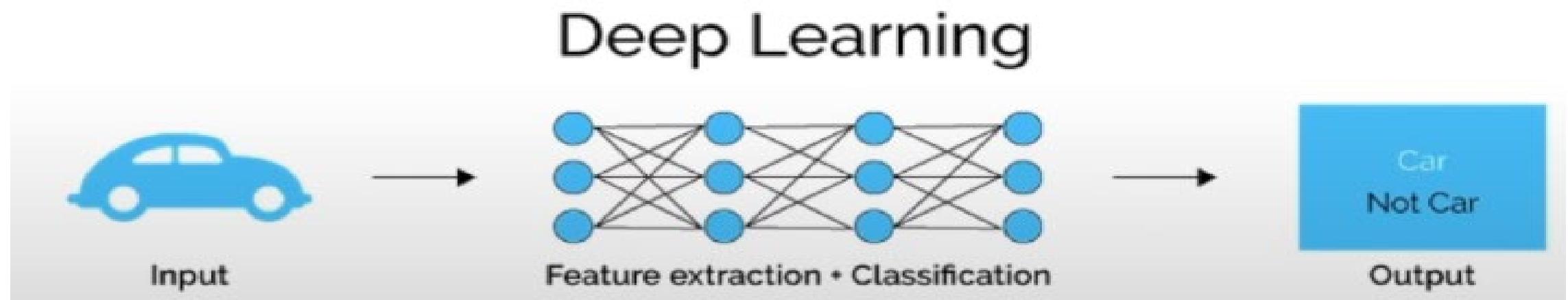
Algorithmes de deep learning

Réseaux de neurones convolutifs

En 2012, une révolution se produit en Deep learning, il s'agit d'un réseau de neurones convolutif appelé AlexNet, mais il est largement inspiré du réseau de neurones convolutifs LetNet, développé en 1998 par le chercheur français **Yann Lecun**

Un RN qui consiste à appliquer des filtres (noyau ou kernel) sur l'image à analyser et qui permet à la fois :

- ✓ D'extraire les caractéristiques de l'image
- ✓ De classer l'image

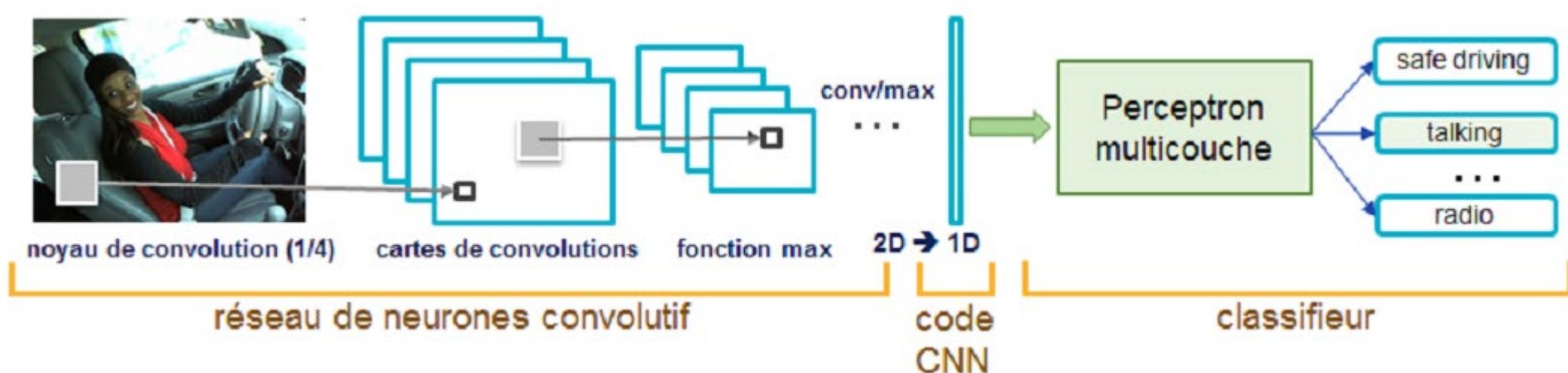


Algorithmes de deep learning

Réseaux de neurones convolutifs

Définition (CNN)

Les réseaux de neurones convolutionnels sont à ce jour les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network, ils comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu]. La première partie d'un CNN est la partie convulsive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN



Réseaux de neurones convolutifs

Définition (CNN)

Les réseaux de neurones convolutionnels sont basés sur le perceptron multicouche(MLP), et inspirés du comportement du cortex visuel des vertébrés. Bien qu'efficaces pour le traitement d'images, les MLP ont beaucoup de mal à gérer des images de grande taille, ce qui est dû à la croissance exponentielle du nombre de connexions avec la taille de l'image.

Par exemple, si on prend une image de taille 32x32x3 (32 de large, 32 de haut, 3 canaux de couleur), un seul neurone entièrement connecté dans la première couche cachée du MLP aurait 3072 entrées ($32 \times 32 \times 3$). Une image 200x200 conduirait ainsi à traiter 120 000 entrées par neurone ce qui, multiplié par le nombre de neurones, devient énorme.

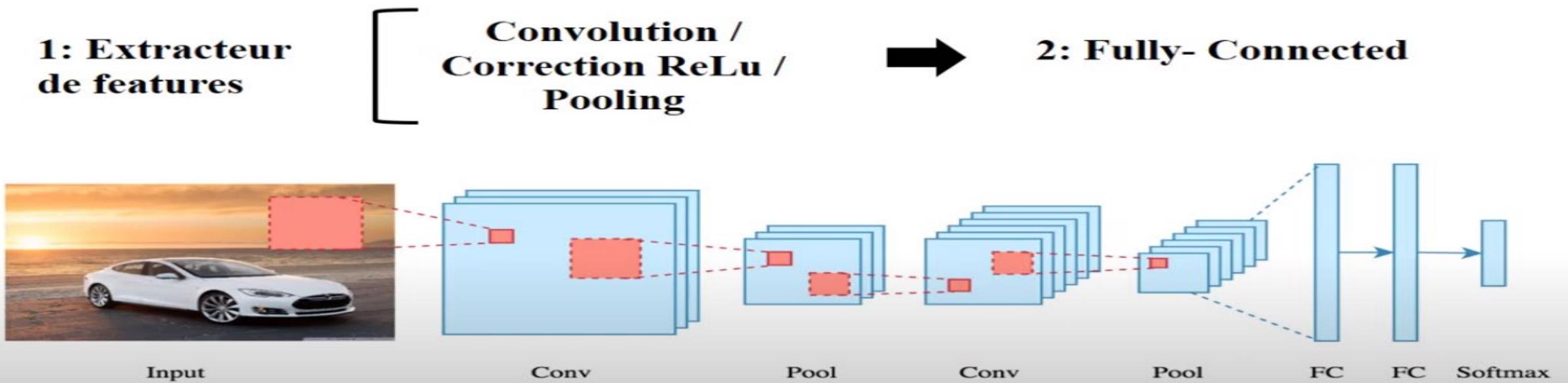
Algorithmes de deep learning

Réseaux de neurones convolutifs

Définition (CNN)

Une architecture CNN est formée par un empilement de couches de traitement indépendantes :

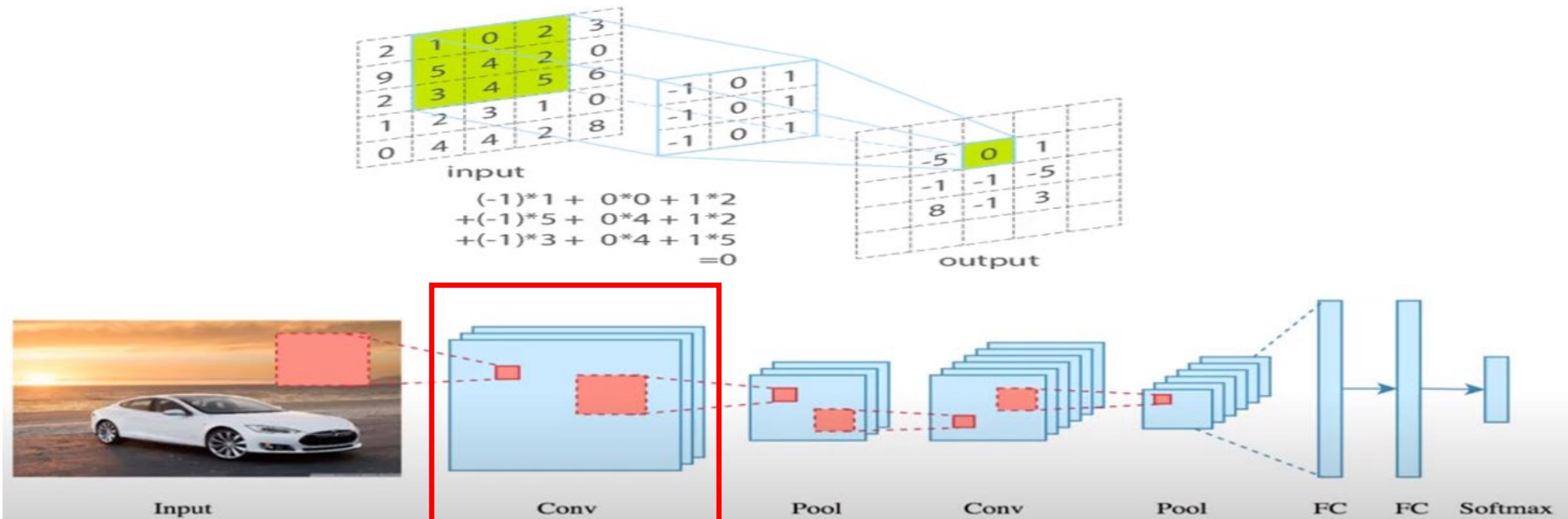
- La couche de **convolution** (CONV) qui traite les données d'un champ récepteur.
- La couche de **pooling** (POOL), qui permet de compresser l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage).
- La couche de **correction** (ReLU), souvent appelée par abus 'ReLU' en référence à la fonction d'activation (Unité de rectification linéaire).
- La couche "entièrement connectée" (FC), qui est une couche de type perceptron.
- La couche de perte (LOSS).



Réseaux de neurones convolutifs

La couche de convolution

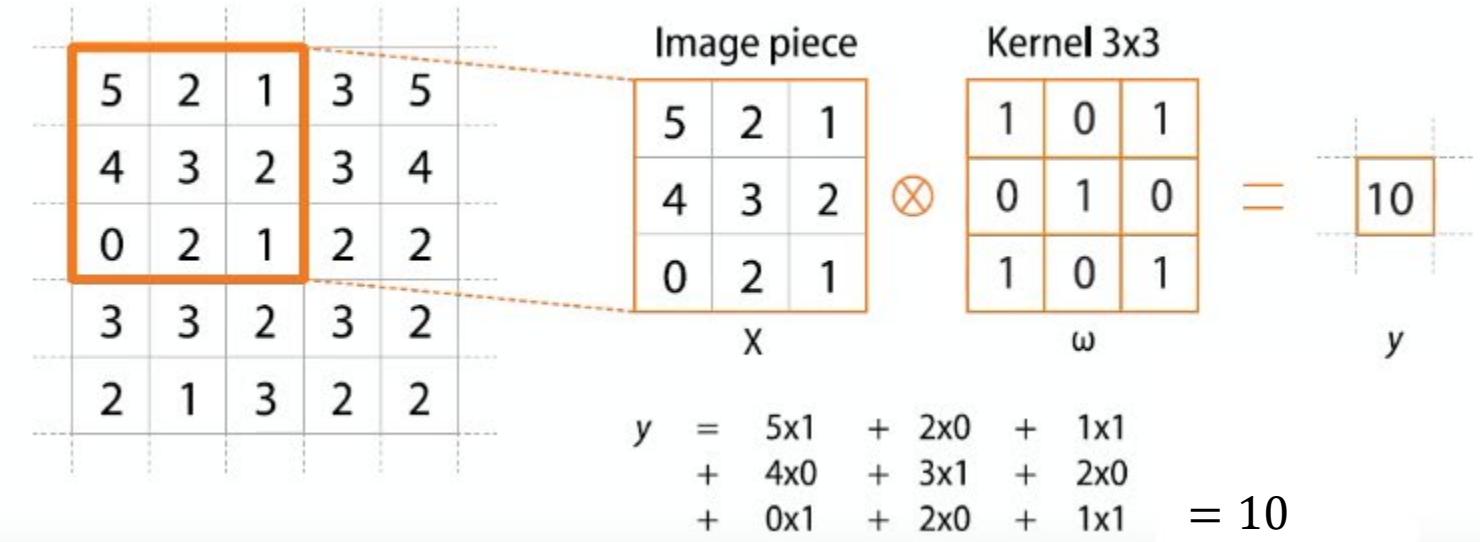
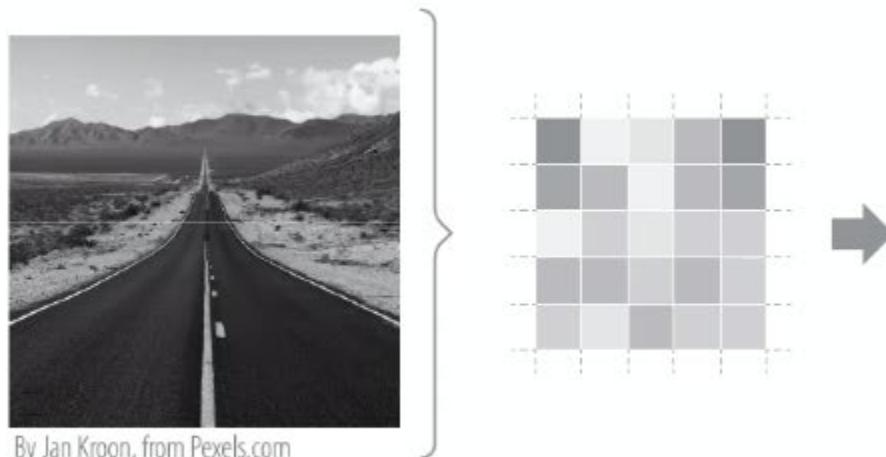
La convolution, ou produit de convolution, est une généralisation du filtre moyenneur où l'on considère cette fois une moyenne pondérée. La fenêtre glissante est alors elle-même une image qui contient les coefficients de pondération. On l'appelle généralement noyau de convolution ou masque (kernel ou mask en anglais)



Algorithmes de deep learning

Réseaux de neurones convolutifs

La couche de convolution



Convolution 2D (image sans couleur)

$$y = \sum_{i=1}^n \sum_{j=1}^m x_{i,j} \cdot w_{i,j} \quad \text{avec} \begin{cases} n & \text{largeur de noyau(kernel)} \\ m & \text{hauteur de noyau(kernel)} \end{cases}$$

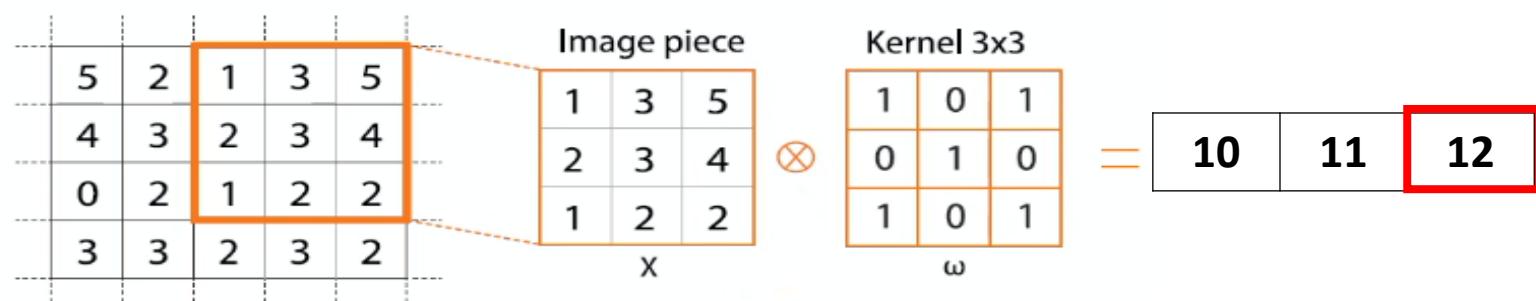
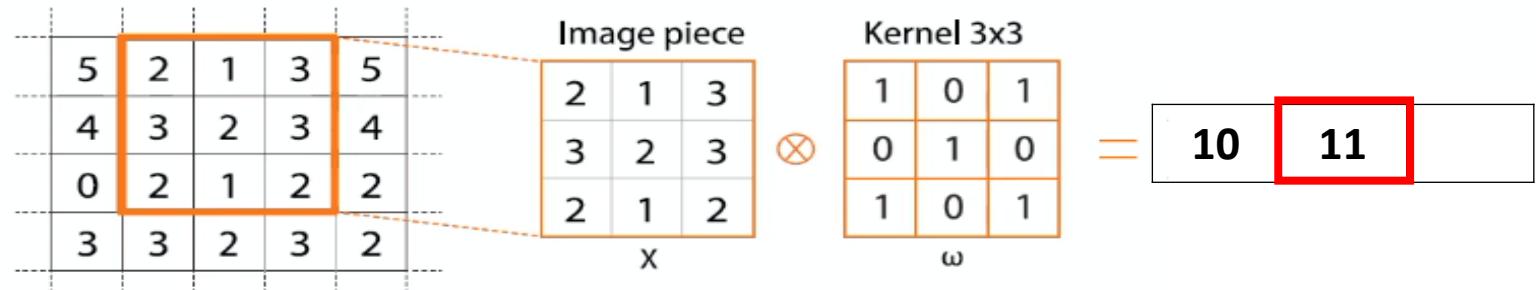
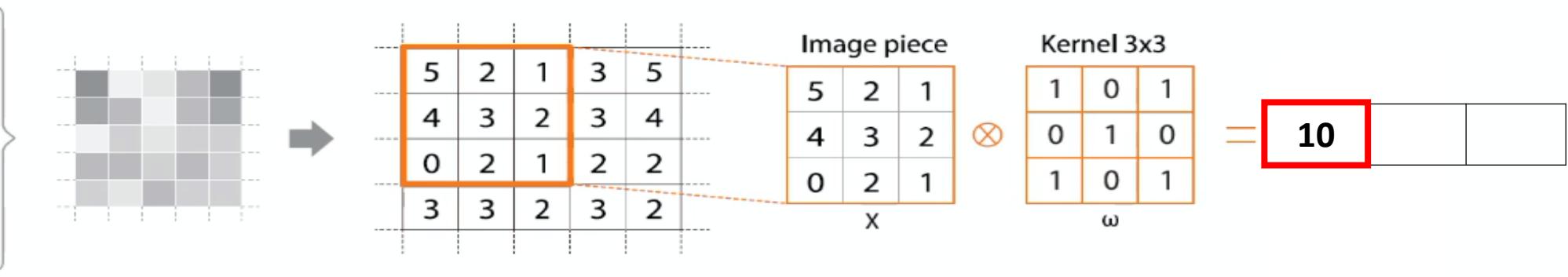
Algorithmes de deep learning

Réseaux de neurones convolutifs

La couche de convolution



By Jan Kroon, from Pexels.com

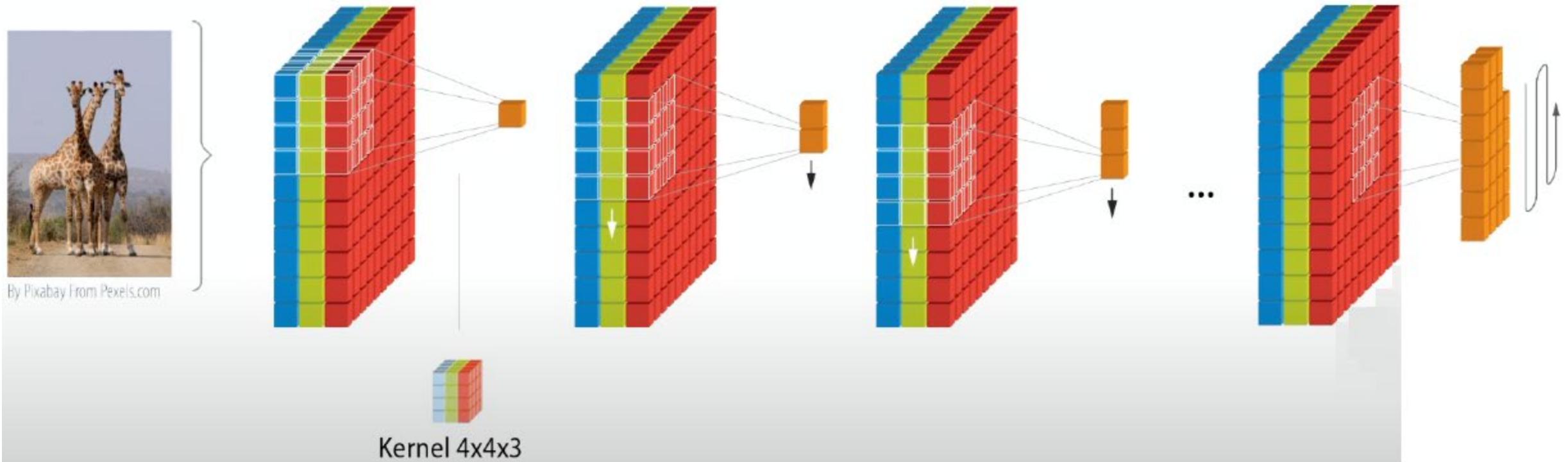


Convolution 2D

Algorithmes de deep learning

Réseaux de neurones convolutifs

La couche de convolution



Convolution 3D (image avec couleur)

3 = les trois couleurs (RGB), Red, Green, Blue

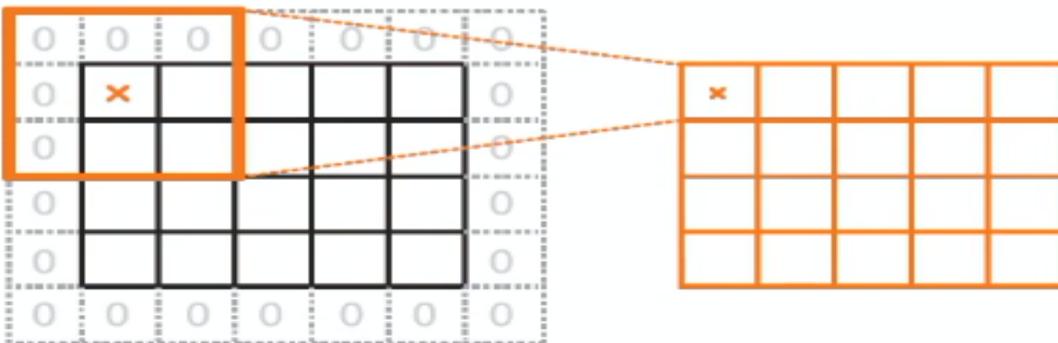
Algorithmes de deep learning

Réseaux de neurones convolutifs

La couche de convolution

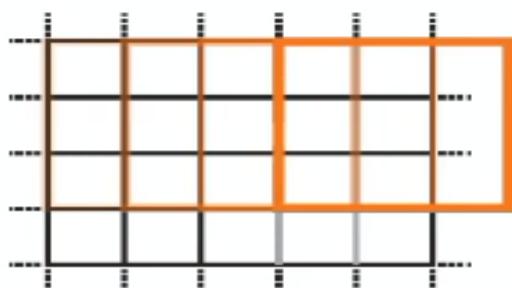
padding

(Rembourrage)



strides

(Pas)



Strides = (dx, dy)
dx = décalage horizontal
dy = décalage vertical
Ici **dx=1**

padding = 'same'

L'utilisation de padding permet de conserver la même taille de l'image

padding = 'valid'

Pas de padding permet de renvoyer une image de taille différente

Strides (2,2) déplace l'image de deux pixels, revoie une convolution deux fois plus petite que l'image de départ

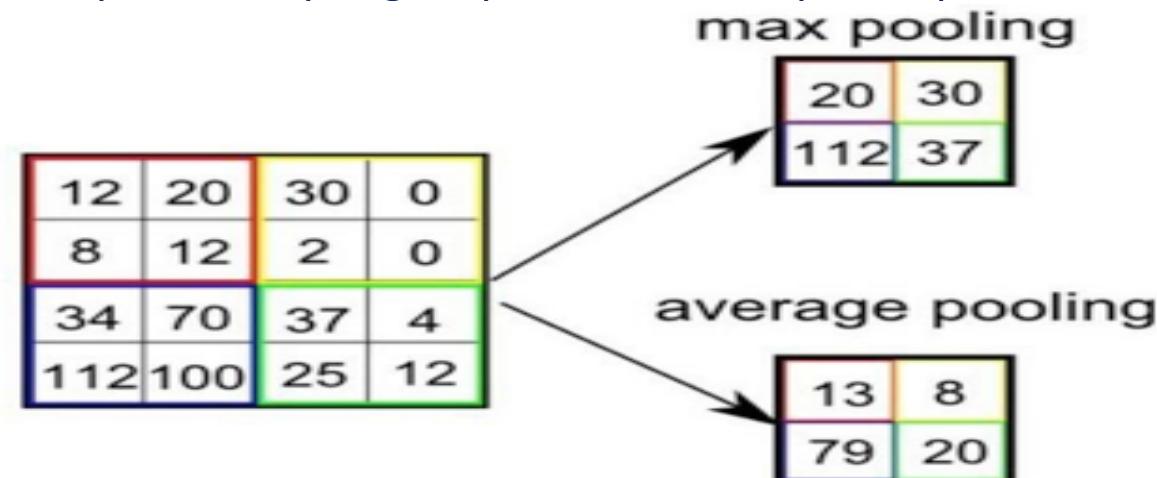
Réseaux de neurones convolutifs

Couche de pooling

La mise en commun est principalement la réduction d'échelle d'une image acquise à partir de couches précédentes tout en **préservant leurs caractéristiques importantes**. La mise en commun maximale (**max-pooling**) est un type de mise en commun populaire utilisé par beaucoup. Par exemple, vous plantez pour mettre en commun avec un rapport de deux. Il permet de réduire de moitié la largeur et la hauteur de votre image. Par conséquent, vous comprimez les pixels (un sur quatre) en une grille de 2 par 2, puis vous la mappez sur un nouveau pixel.

Vous devez prendre la plus grande valeur des quatre pixels pour obtenir un maximum de regroupement. Ainsi, un seul nouveau pixel représente essentiellement quatre anciens en utilisant la plus grande valeur des quatre pixels. Ce processus se produit pour chaque groupe contenant quatre pixels tout autour de l'image

On utilise souvent des cellules carrées pour ne pas avoir un chevauchement des pixels

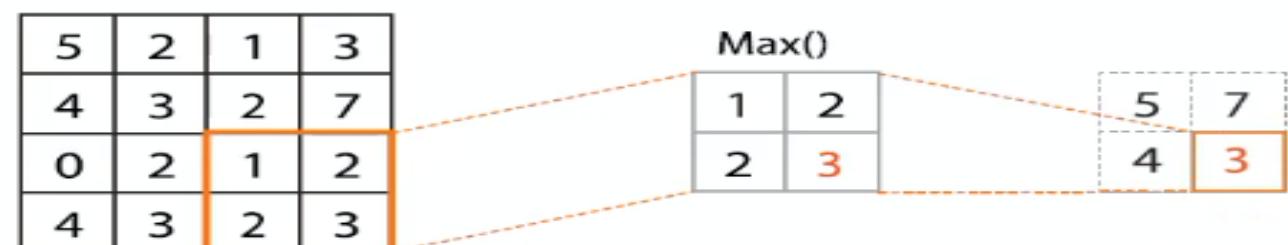
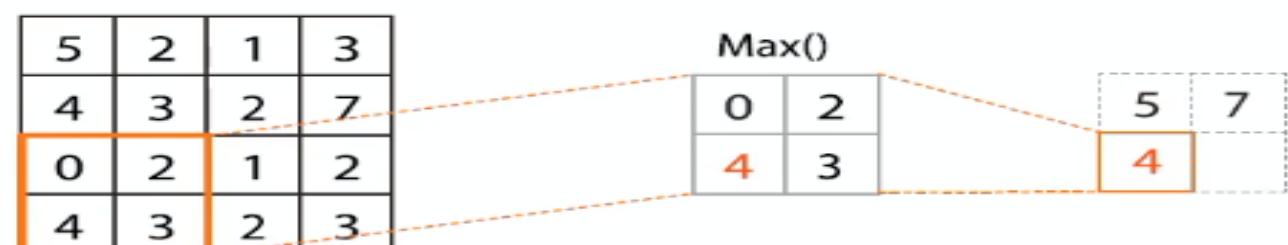
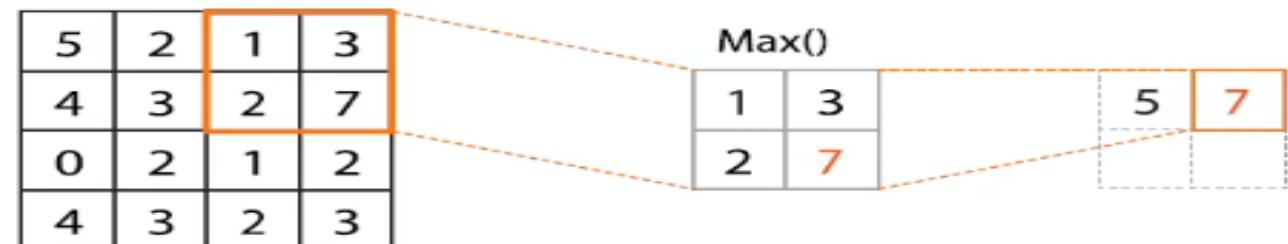
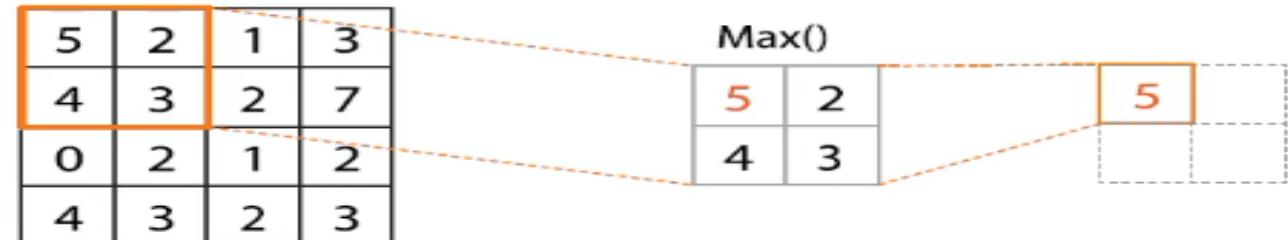


Algorithmes de deep learning

Réseaux de neurones convolutifs

Couche de pooling

Par défaut, le **strides** correspond à la taille de la fenêtre



Réseaux de neurones convolutifs

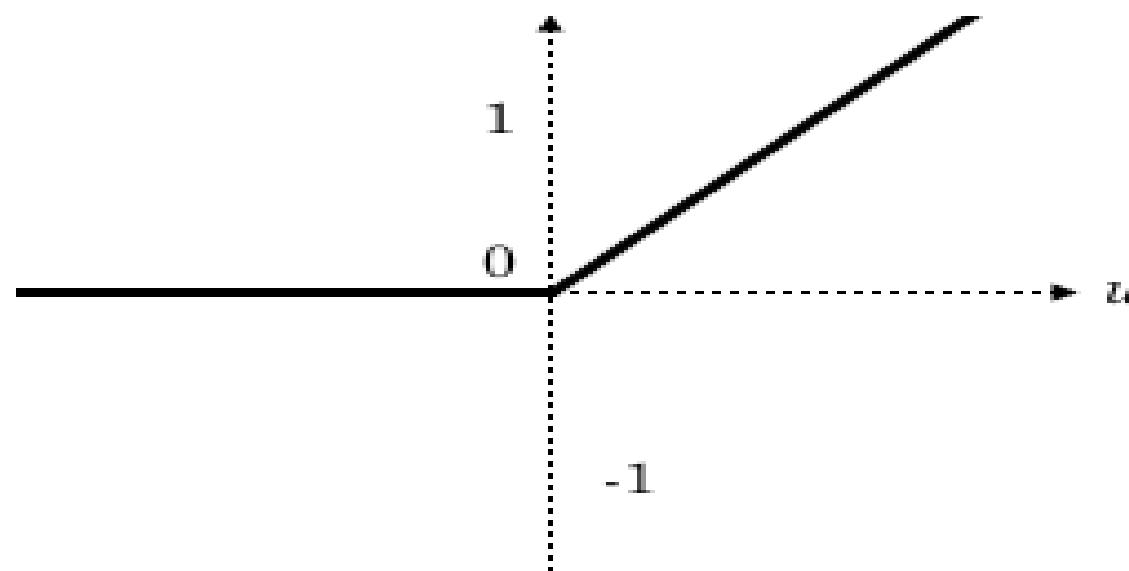
Couches de correction (ReLU)

Il est possible d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie.

La fonction **ReLU** (abréviation de Unités Rectifiée linéaires) : $F(x) = \max(0, x)$ Cette fonction force les neurones à retourner des valeurs positives.

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation

$$f(u) = \max(0, u)$$



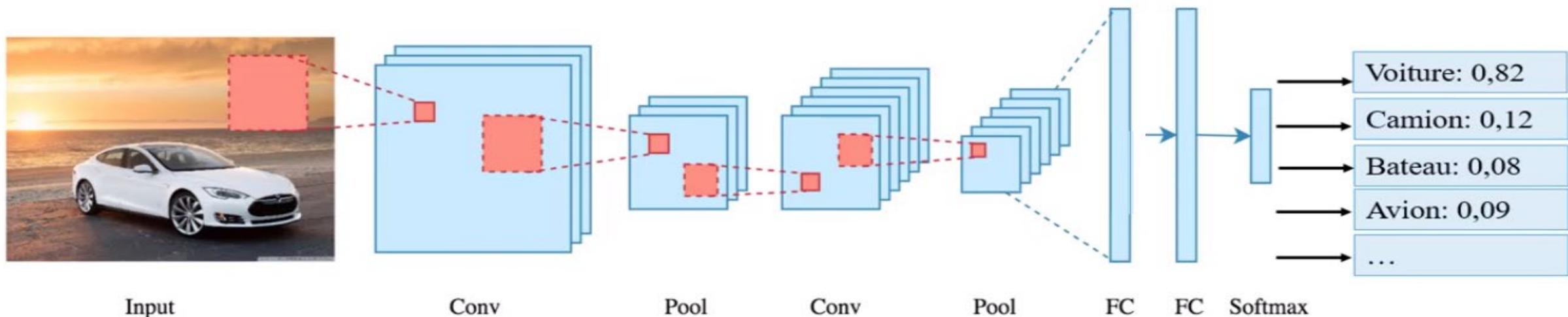
Algorithmes de deep learning

Réseaux de neurones convolutifs

Couche entièrement connectée (fully-connected)

La couche **fully-connected** constitue toujours la dernière couche d'un réseau de neurones, ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

La dernière couche **fully-connected** permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.



Algorithmes de deep learning

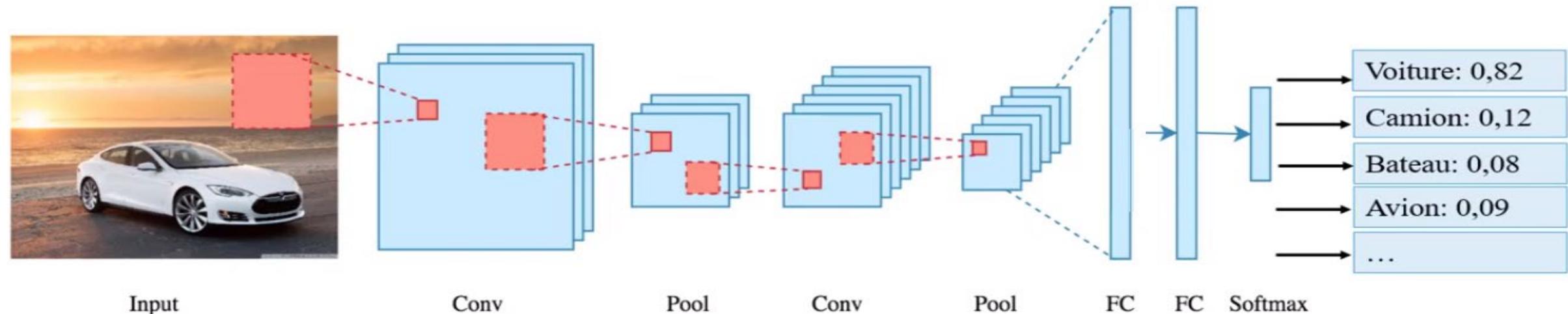
Réseaux de neurones convolutifs

Couche entièrement connectée (fully-connected)

Par exemple, si le problème consiste à distinguer les voitures des autres véhicules, le vecteur final sera de taille N (N nombre de classe) : le premier élément (respectivement, le deuxième...) donne la probabilité d'appartenir à la classe « Voiture» (respectivement « Camion»). Ainsi, le vecteur [0.82 0.12 ...] signifie que l'image a 82% de chances de représenter une Voiture.

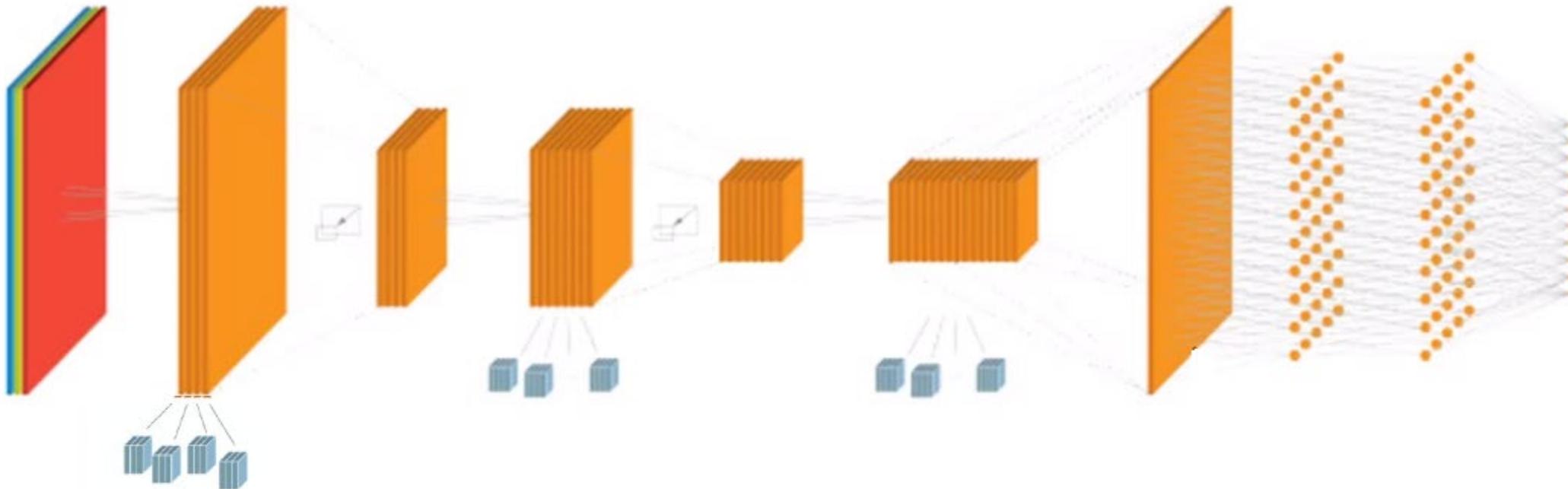
Pour calculer les probabilités, la couche **fully-connected** multiplie donc chaque élément en entrée par un poids, fait la somme, puis applique une fonction d'activation (**logistique** si $N=2$, **softmax** si $N>2$) :

Ce traitement revient à multiplier le vecteur en entrée par la matrice contenant les poids. Le fait que chaque valeur en entrée soit connectée avec toutes les valeurs en sortie explique le terme **fully-connected**.



Réseaux de neurones convolutifs

Synthèse



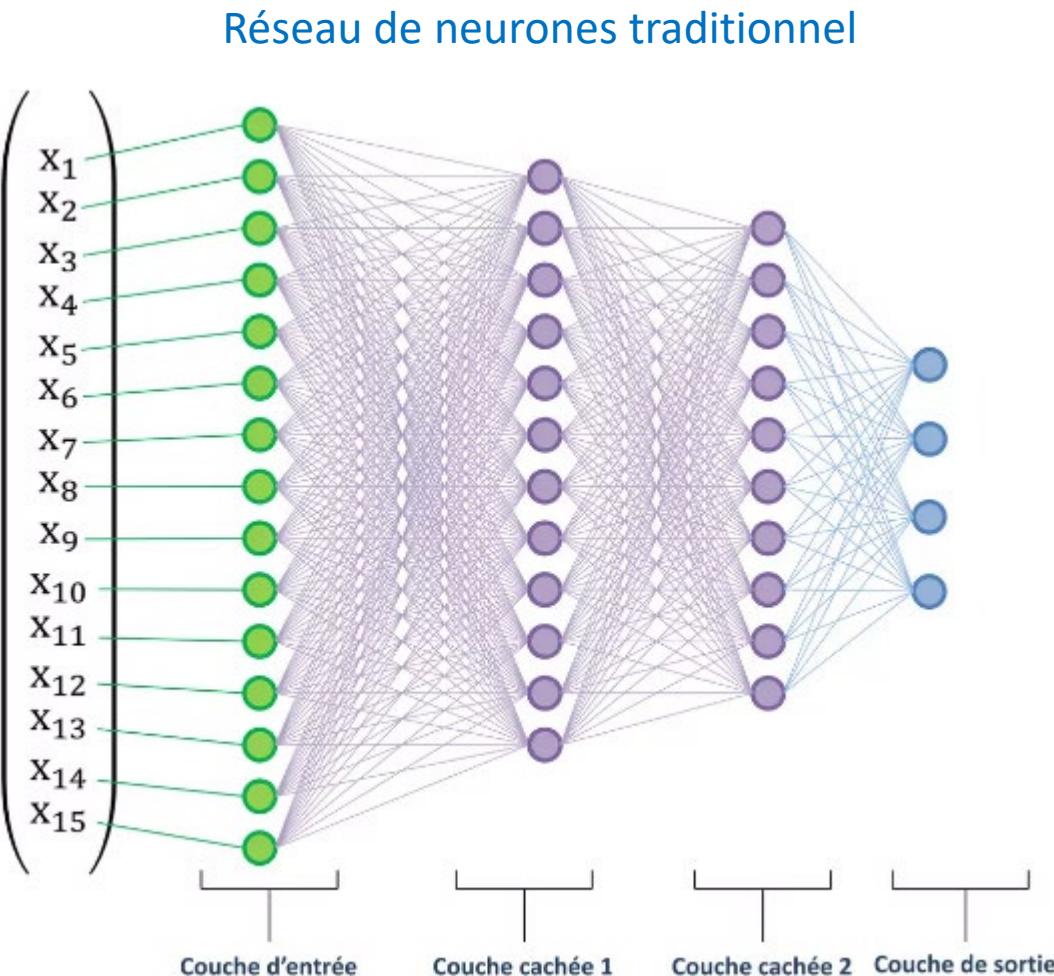
Input Layer	Convolutional Layers	Pooling	Convolutional Layers	Pooling	Convolutional Layers	Flat Layer	Dense Layers	Output Layer
-------------	----------------------	---------	----------------------	---------	----------------------	------------	--------------	--------------

Implémentation pratique avec Python et sklearn, tensorflow, keras (TP – CNN)

Algorithmes de deep learning

Les réseaux de neurones récurrents

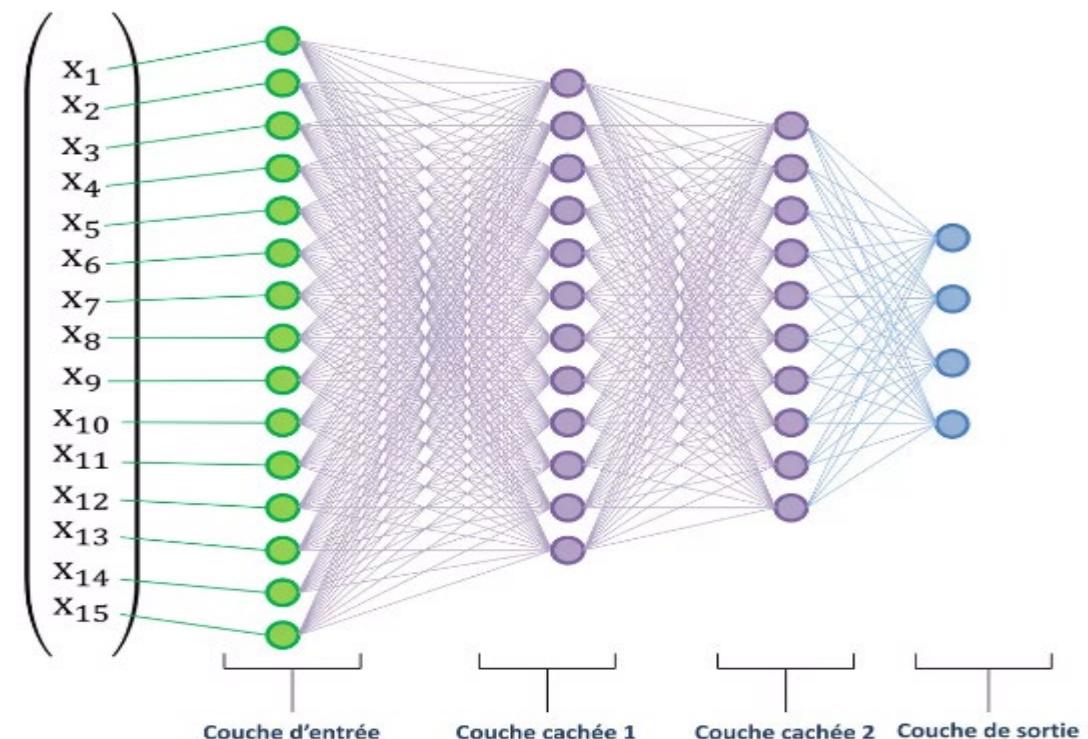
Exemple d'architecture



Les réseaux de neurones récurrents

Limites des architectures traditionnelles :

- Taille fixe de la couche d'entrée
- Taille fixe de la couche de sortie
- Architecture fixe du réseau
- Ne tiennent pas compte de l'ordre des données
- Non adaptées aux séries temporelles



Les réseaux de neurones récurrents

Définition

Un réseau de neurones récurrent (RNN, recurrent neural network) est un type de réseau de neurones artificiels principalement utilisé dans la reconnaissance automatique de la parole, dans l'écriture manuscrite et dans le traitement automatique du langage naturel, en particulier dans la traduction automatique.

Les RNN sont conçus de manière à reconnaître les caractéristiques séquentielles et pour prédire le scénario suivant le plus probable.

Ils se distinguent des réseaux de neurones traditionnels en ce sens qu'ils utilisent des boucles de rétroaction pour traiter une séquence de données qui façonne le résultat final, lui-même pouvant être une séquence de données. Ces boucles de rétroaction permettent aux informations de persister, effet souvent assimilé à la mémoire.

Les RNN sont ainsi en mesure de traiter des données séquentielles et temporelles.

→ Concept de mémoire

Algorithmes de deep learning

Les réseaux de neurones récurrents

Application de RNN

Traduction automatique

Traduire un texte d'une langue à une autre sans nécessiter d'intervention humaine. Vous pouvez, par exemple, traduire un texte de votre langue maternelle vers l'anglais.

Génération du texte

Sur la base de la séquence précédente de mots/caractères utilisés dans le texte, un modèle entraîné apprend la probabilité d'occurrence d'un mot/caractère. Un modèle peut être entraîné au niveau du caractère, du n-gramme, de la phrase ou du paragraphe.

Reconnaissance de la parole

Ceci est également connu sous le nom de reconnaissance automatique de la parole (ASR), et il est capable de convertir la parole humaine en format écrit ou texte.

Prévision de séries chronologiques

Vous pouvez utiliser les données boursières pour créer un modèle d'apprentissage automatique capable de prévoir les cours boursiers futurs en fonction de ce que le modèle apprend à partir des données historiques. Cela peut aider les investisseurs à prendre des décisions d'investissement basées sur les données.

Algorithmes de deep learning

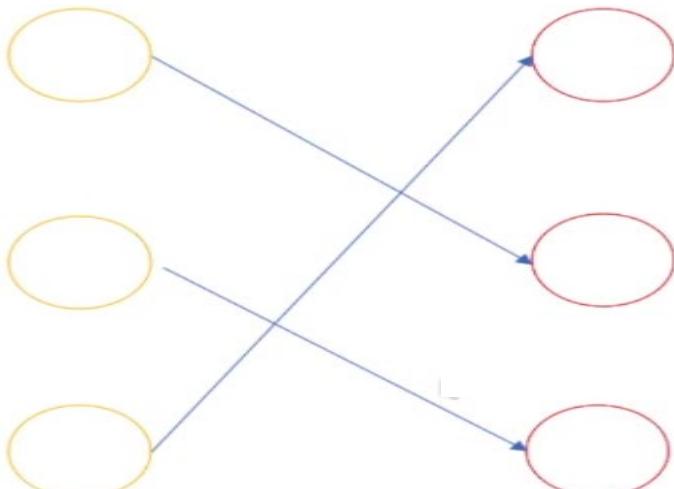
Les réseaux de neurones récurrents

Exemple

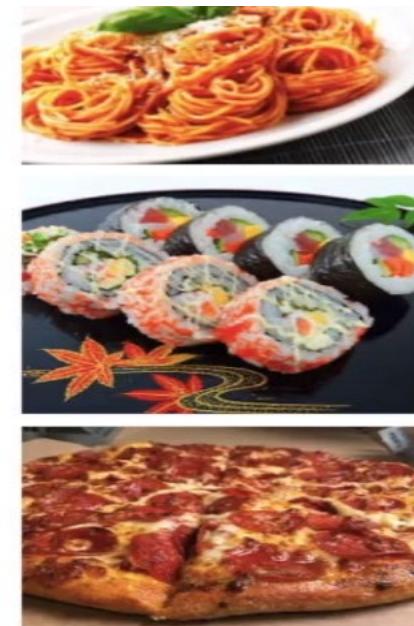
Je mange quoi pour le dîner avec cette séquence



Hier j'ai pris une pizza



Hier j'ai pris des pâtes



Hier j'ai pris du sushi

Une relation séquentielle

Algorithmes de deep learning

Les réseaux de neurones récurrents

Exemple

Décalage à cause des imprévus



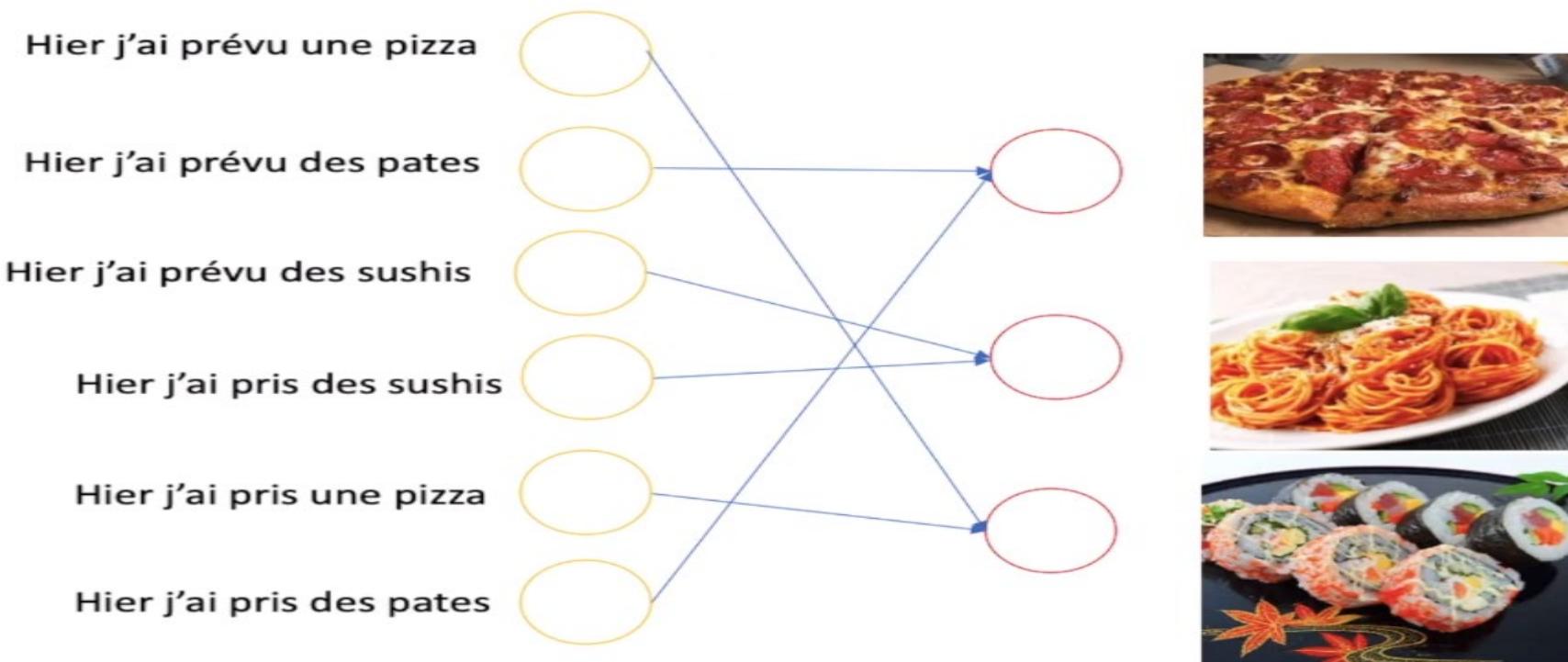
	Lundi	Mardi	Mercredi	Jeudi	Vendredi
Diner prévu					
Ce que j'ai mangé					

Algorithmes de deep learning

Les réseaux de neurones récurrents

Exemple

Représenter l'exemple par un réseaux de neurones



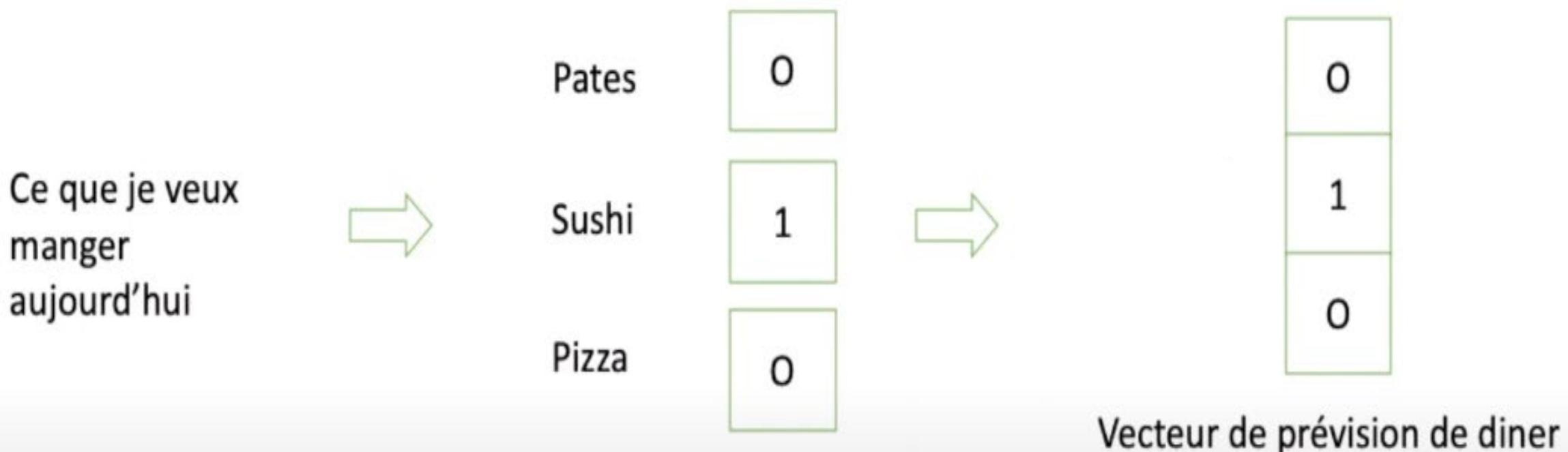
La prévision de dîner de la journée dépend de ce qui j'ai prévu hier et de ce qui j'ai pris hier

Algorithmes de deep learning

Les réseaux de neurones récurrents

Exemple

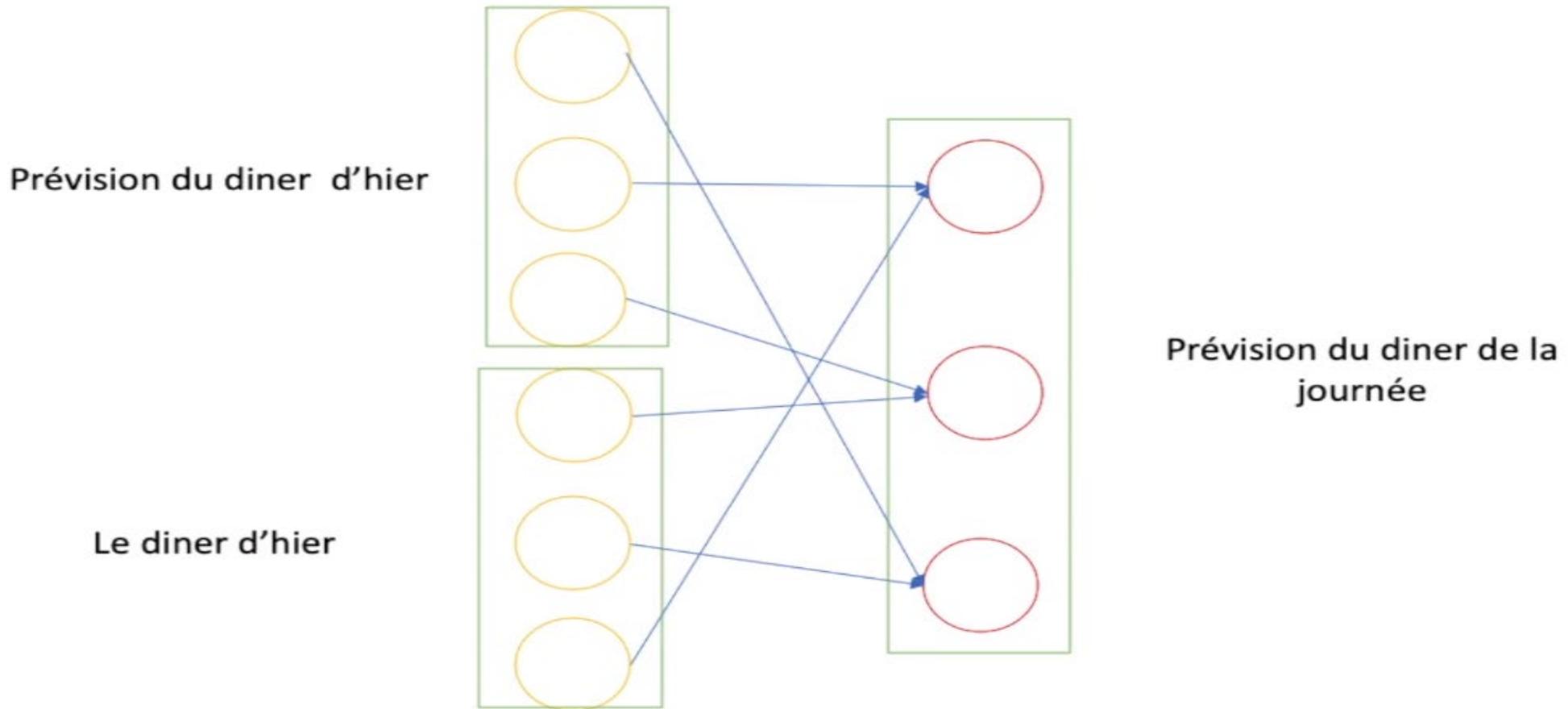
Transformer les données vers des vecteurs



Algorithmes de deep learning

Les réseaux de neurones récurrents

Exemple



Algorithmes de deep learning

Les réseaux de neurones récurrents

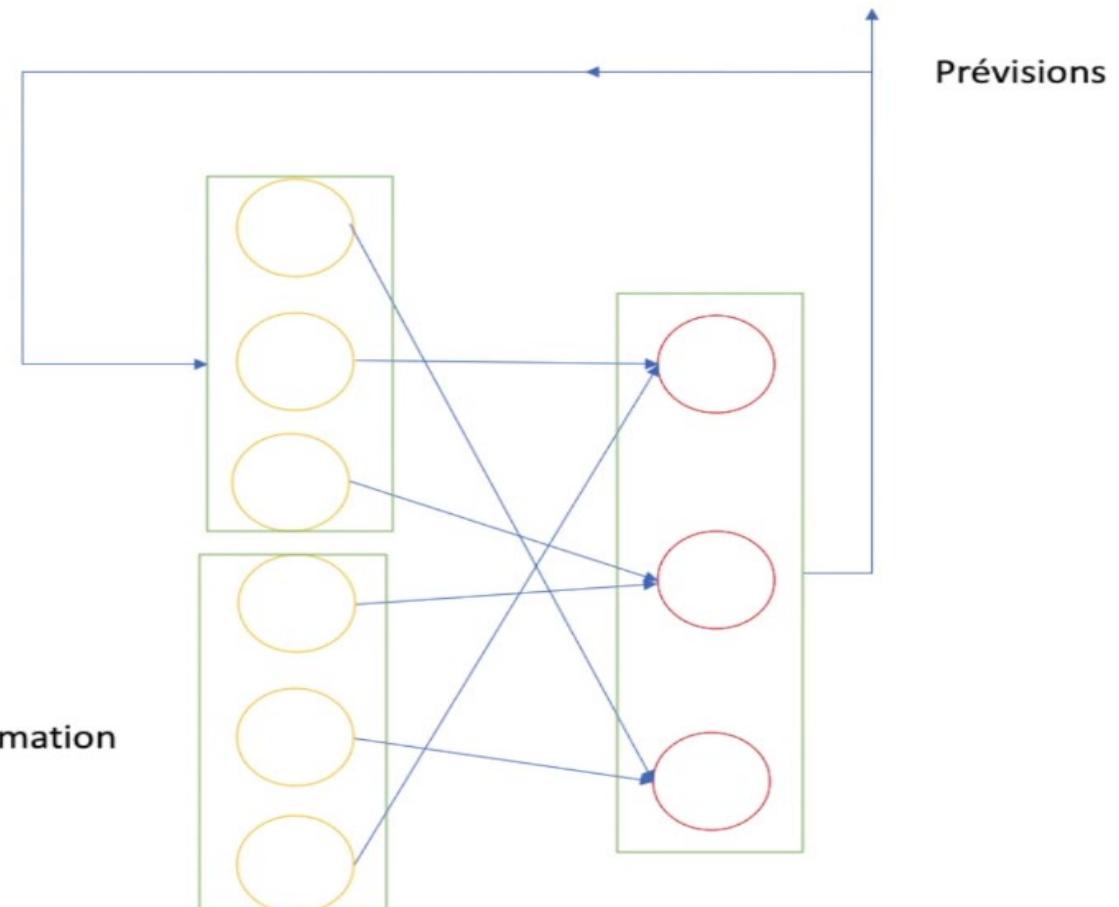
Exemple

Ce ci nous ramène au schéma suivant:
Les prévisions seront injectés comme entrée pour le lendemain.

Les prévision d'aujourd'hui devient les prévisions d'hier pour une nouvelle information.

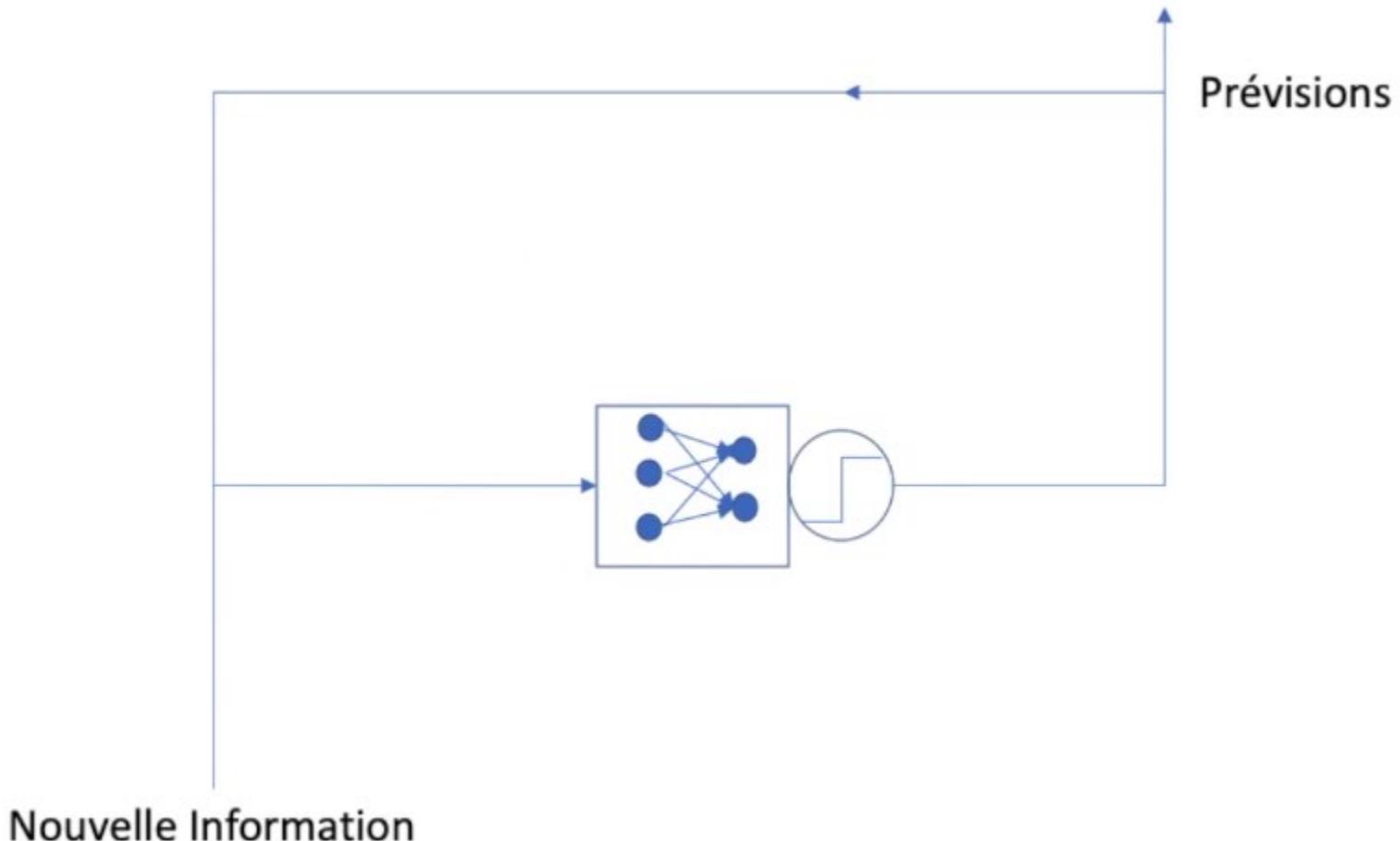
La clé de RNN, la sortie de réseau sera réutilisée dans l'entrée, d'où le terme **récurrent**.

Nouvelle Information



Les réseaux de neurones récurrents

Principe

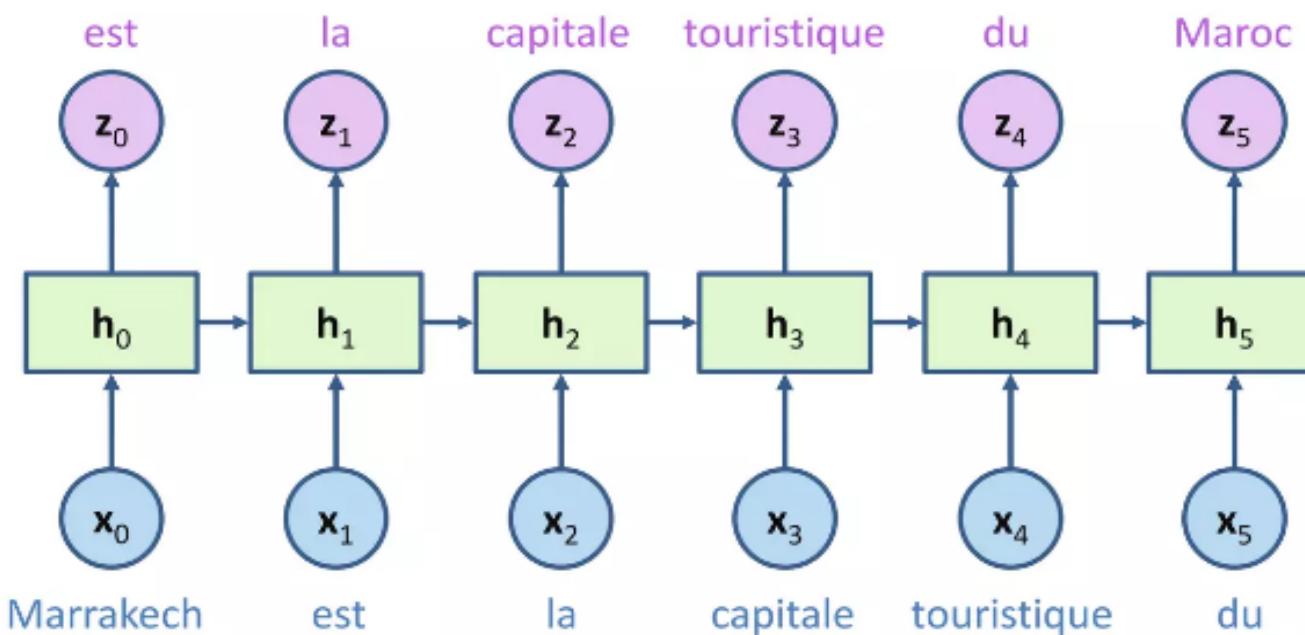


Algorithmes de deep learning

Les réseaux de neurones récurrents

Exemple

Un réseau récurrent peut par exemple générer une séquence de mots à partir d'un premier mot entré, « Marrakech » dans l'exemple suivant :

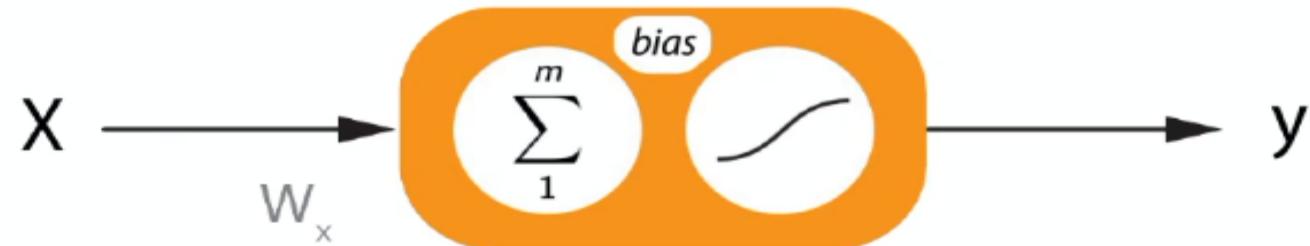


Algorithmes de deep learning

Les réseaux de neurones récurrents

ANN vs RNN

Réseau de neurones classique



X : les entrées (vecteur)

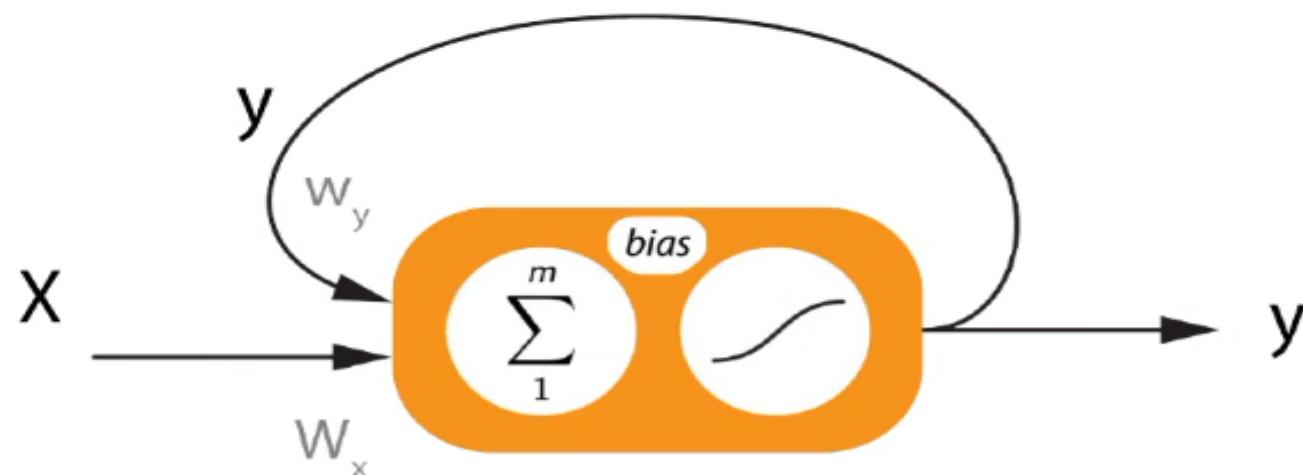
W_x : les poids (vecteur)

Y : la sortie (scalaire)

W_y : un poids (scalaire)

σ : Fonction d'activation

Réseau de neurones récurrent

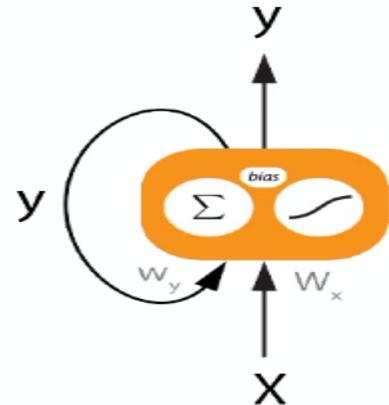


$$y_{(t)} = \sigma(W_x^T \cdot X_{(t)} + w_y \cdot y_{(t-1)} + b)$$

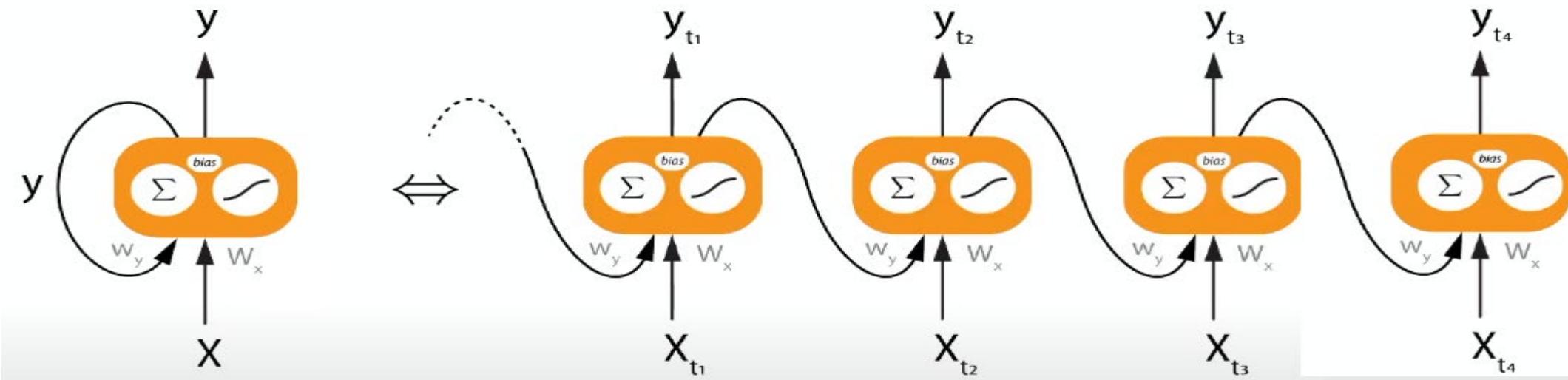
Algorithmes de deep learning

Les réseaux de neurones récurrents

ANN vs RNN



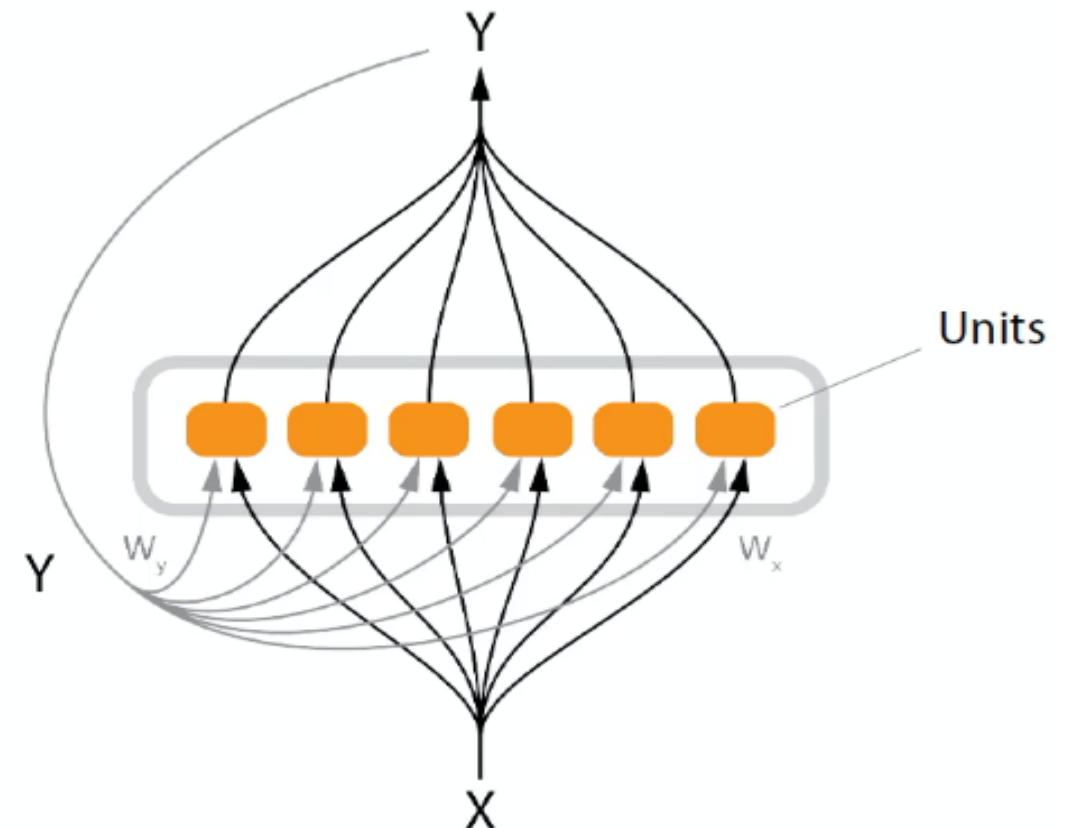
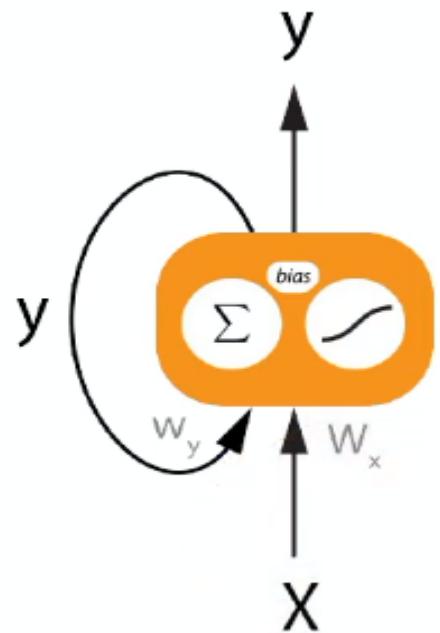
$$y_{(t)} = \sigma(W_x^T \cdot X_{(t)} + w_y \cdot y_{(t-1)} + b)$$



Les réseaux de neurones récurrents

ANN vs RNN

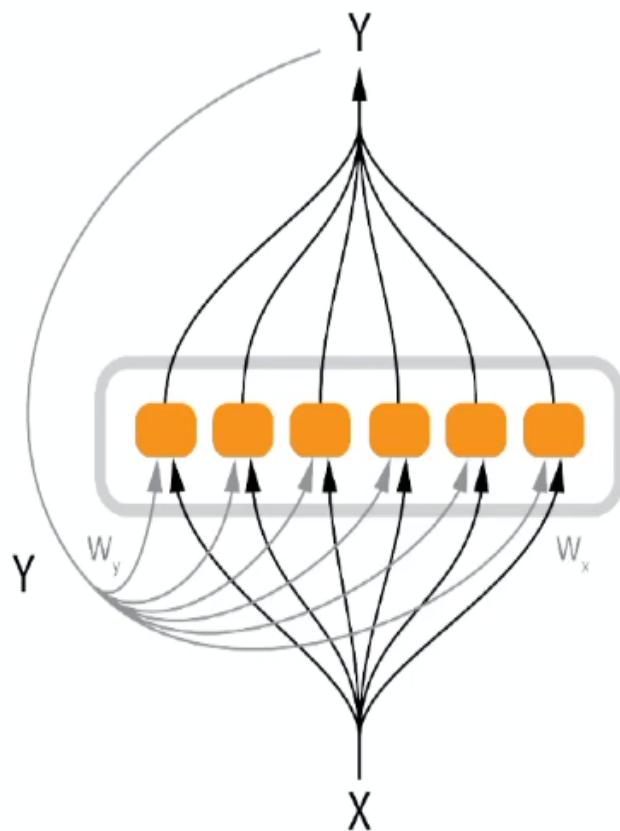
Les réseaux de neurones récurrents seront utilisées ensemble pour construire une cellule ou couche récurrente qui se compose des **Units** (neurones récurrents)



Algorithmes de deep learning

Les réseaux de neurones récurrents

ANN vs RNN



X: les entrées (vecteur)

W_x : les poids (Matrice)

Y: la sortie (Vecteur)

W_y : les poids (Matrice)

\emptyset : Fonction d'activation

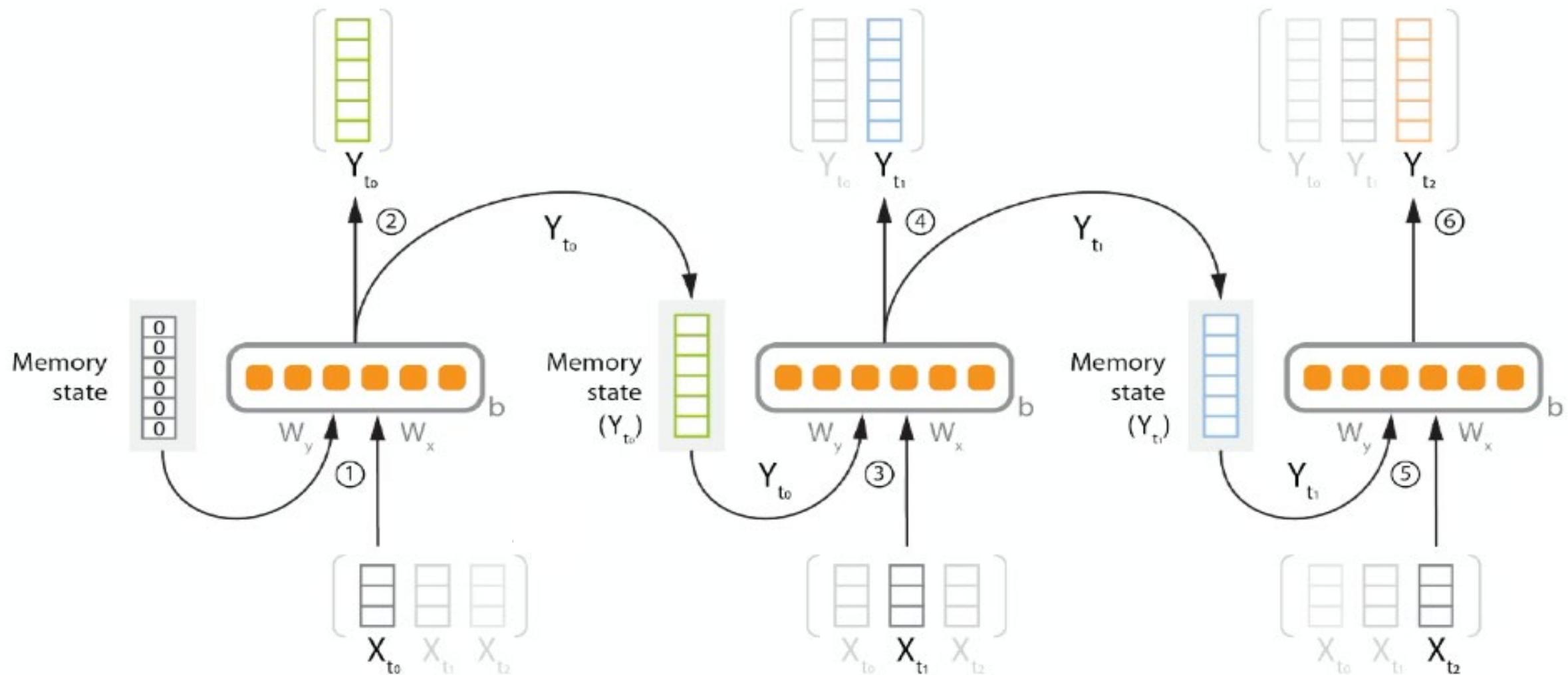
b: Bias (Vecteur)

$$Y_{(t)} = \emptyset (W_x^T \cdot X_{(t)} + W_y^T \cdot Y_{(t-1)} + b)$$

Algorithmes de deep learning

Les réseaux de neurones récurrents

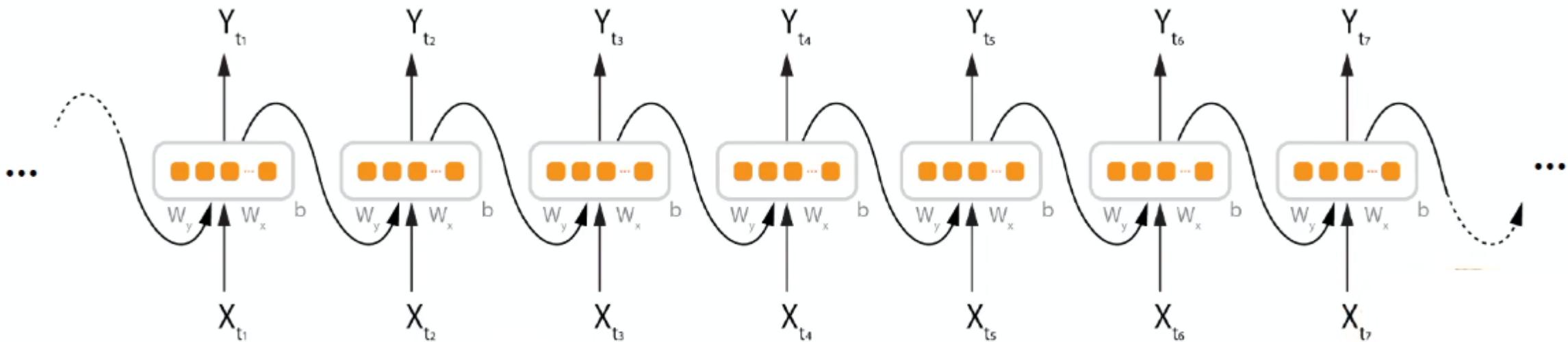
Fonctionnement



Algorithmes de deep learning

Les réseaux de neurones récurrents

Limites de RNN



Si on part sur des grandes séquences qui nécessitent beaucoup d'étapes, notre mémoire se dissout complètement, on arrive à la fin sans se rappeler du tout de ce qu'il y a au début.

Mémoire courte

Bien que les RNN soient très pratiques comparé à une architecture ANN classique pour le traitement des données séquentielles, il s'avère qu'ils sont extrêmement difficiles à entraîner pour gérer la dépendance à long terme en raison du problème de la disparition du gradient (Gradient Vanishing). Une explosion de gradient peut également se produire mais très rarement. Pour surmonter ces lacunes, de nouvelles variantes RNN ont été introduites dans la littérature (LSTM).

Les réseaux de neurones récurrents

Limites de RNN

Les RNN ne permettent pas de mémoriser les dépendances à long ou très long terme. Par exemple, dans la phrase suivante, le réseau récurrent risque de ne pas considérer le lien entre les mots « Marrakech » et « rouge » :



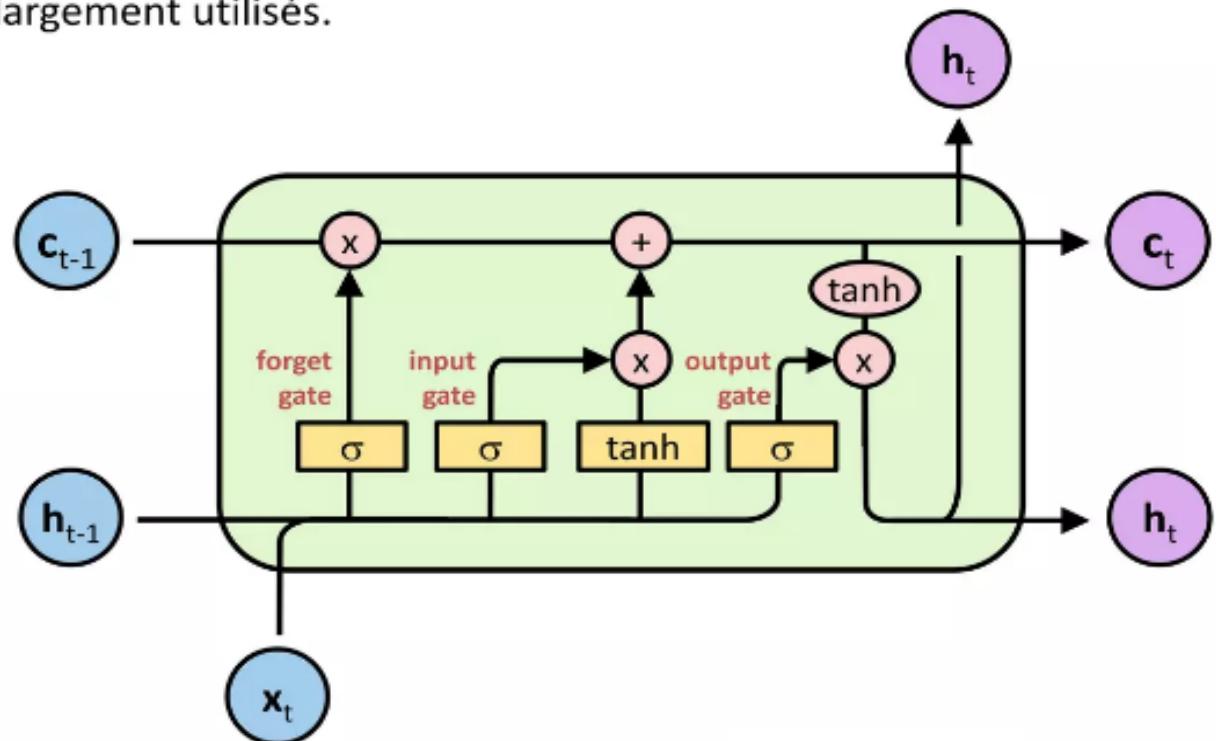
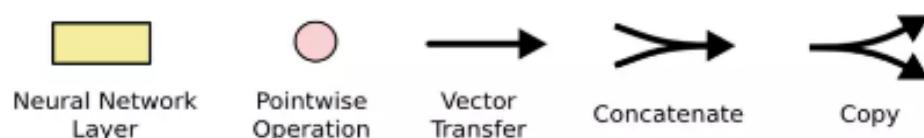
Long Short Term Memory (LSTM)

Définition

Les réseaux LSTM (Long Short Term Memory ou mémoire à long terme et à court terme) sont un type spécial de RNN, capable d'apprendre les dépendances à long terme. Ils ont été introduits par Hochreiter et Schmidhuber en 1997, et ont été par la suite affinés et popularisés à travers plusieurs travaux. Ils fonctionnent extrêmement bien sur une grande variété de problèmes et sont maintenant largement utilisés.

Nous allons présenter les LSTM dans une suite de transformations des réseaux RNN standard.

Nous utiliserons alors les symboles suivants^(*) :

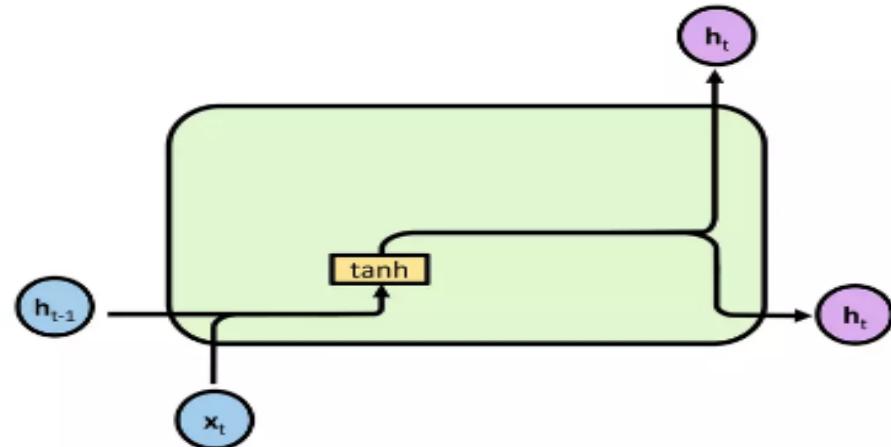


Long Short Term Memory (LSTM)

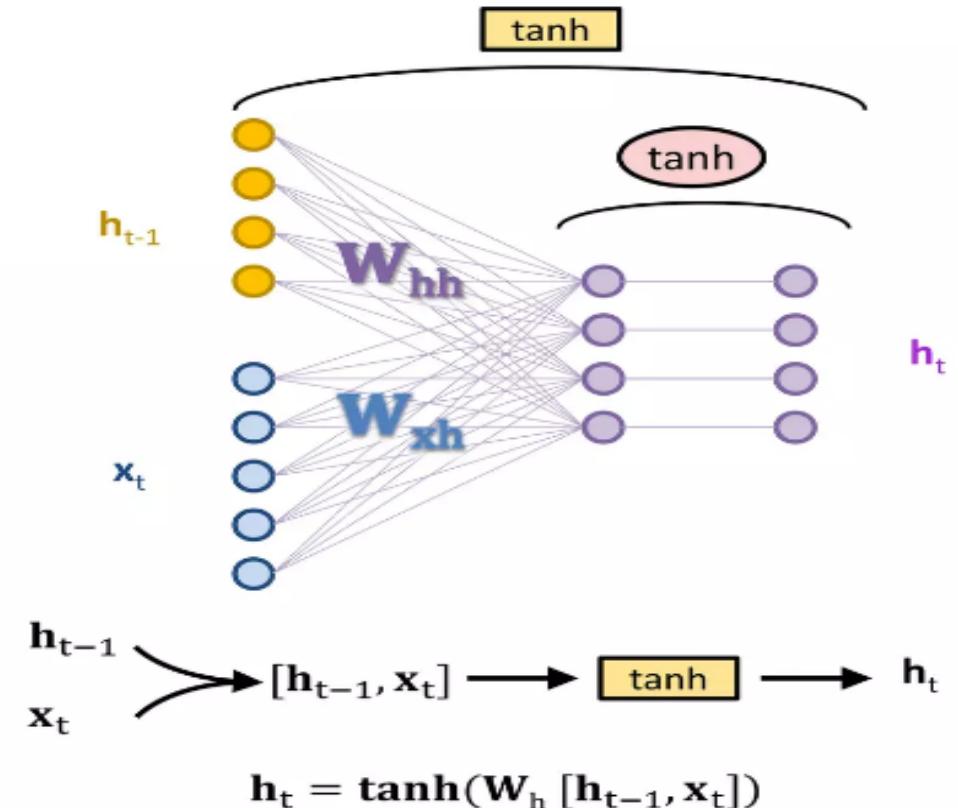
RNN standard

Pour un RNN standard, l'état caché est actualisé par la formule :

$$\mathbf{h}_t = \tanh(\mathbf{W}_{hh} \mathbf{h}_{t-1} + \mathbf{W}_{xh} \mathbf{x}_t)$$



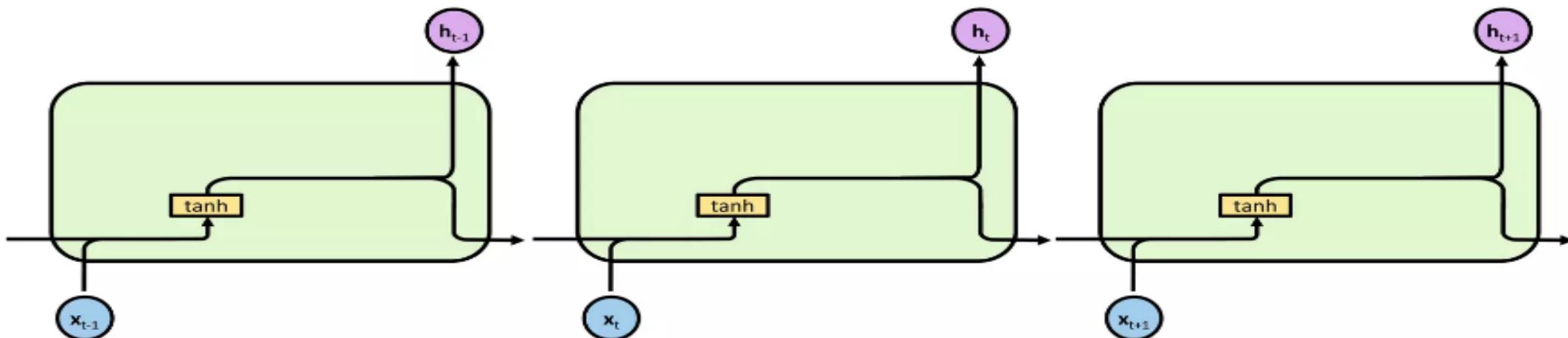
\tanh est utilisée pour aider à réguler les valeurs circulant à travers le réseau. La fonction \tanh écrase les valeurs pour qu'elles soient toujours comprises entre -1 et 1.



Long Short Term Memory (LSTM)

Réseaux RNN

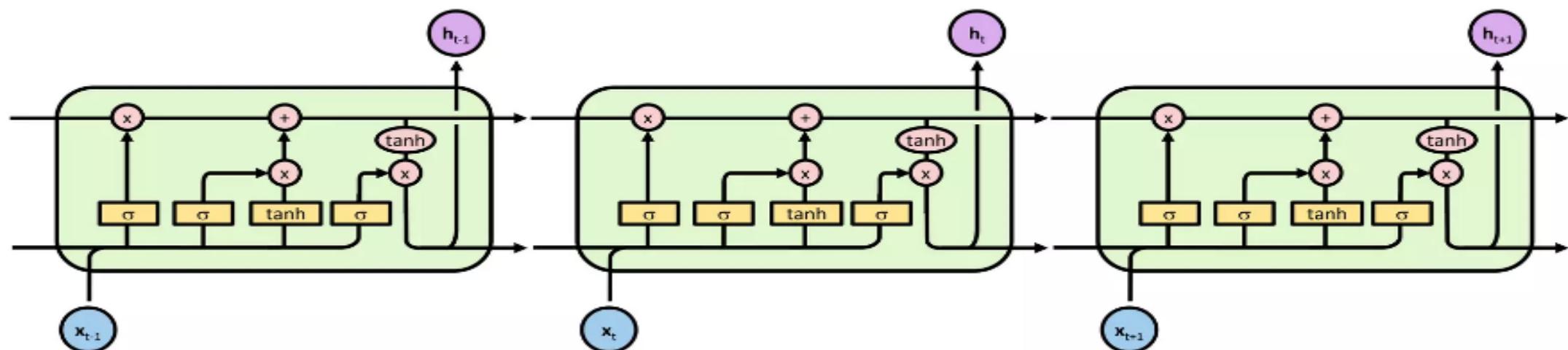
Tous les réseaux de neurones récurrents ont la forme d'une chaîne de modules (ou cellules) répétitifs de réseau de neurones. Dans les RNN standard, ce module répétitif aura une structure très simple, telle qu'une seule couche de tanh.



Long Short Term Memory (LSTM)

Réseaux LSTM

Les LSTM ont également cette structure en chaîne, sauf que le module répétitif au lieu d'avoir une seule couche de réseau de neurones dispose de quatre, qui ont des rôles spéciaux que nous allons détailler par la suite.



Long Short Term Memory (LSTM)

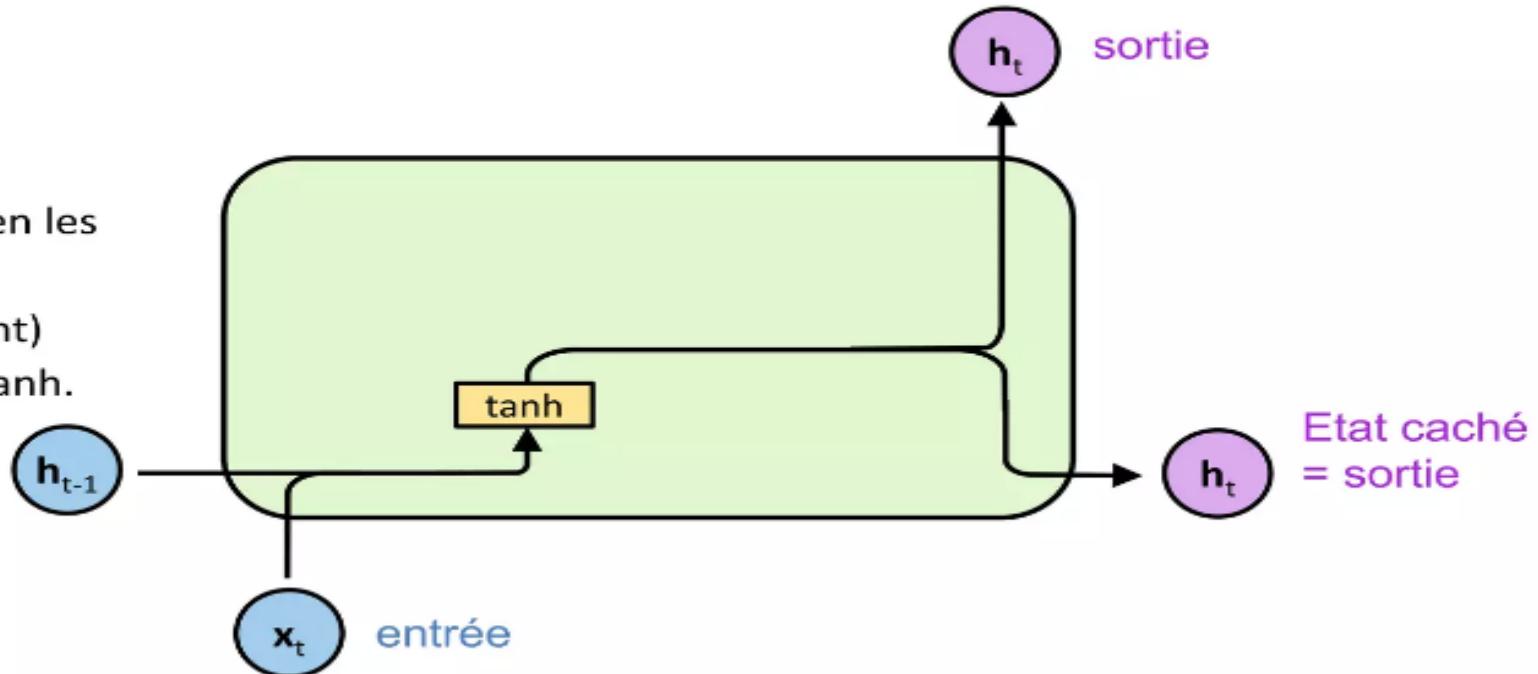
Cellule RNN

Nous allons présenter les LSTM dans une suite de transformations du réseau RNN standard. L'état caché est actualisé par la formule :

Equations

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

Le plus souvent ce réseau ne gère pas bien les dépendances à long terme à cause de la disparition du gradient (vanishing gradient) comme conséquence de l'utilisation de tanh.



Long Short Term Memory (LSTM)

Construire la cellule LSTM

Les LSTM sont explicitement conçus pour éviter le problème de dépendance à long terme. On introduit une nouvelle variable c_t qui permettra de se souvenir des informations pendant de longues périodes.

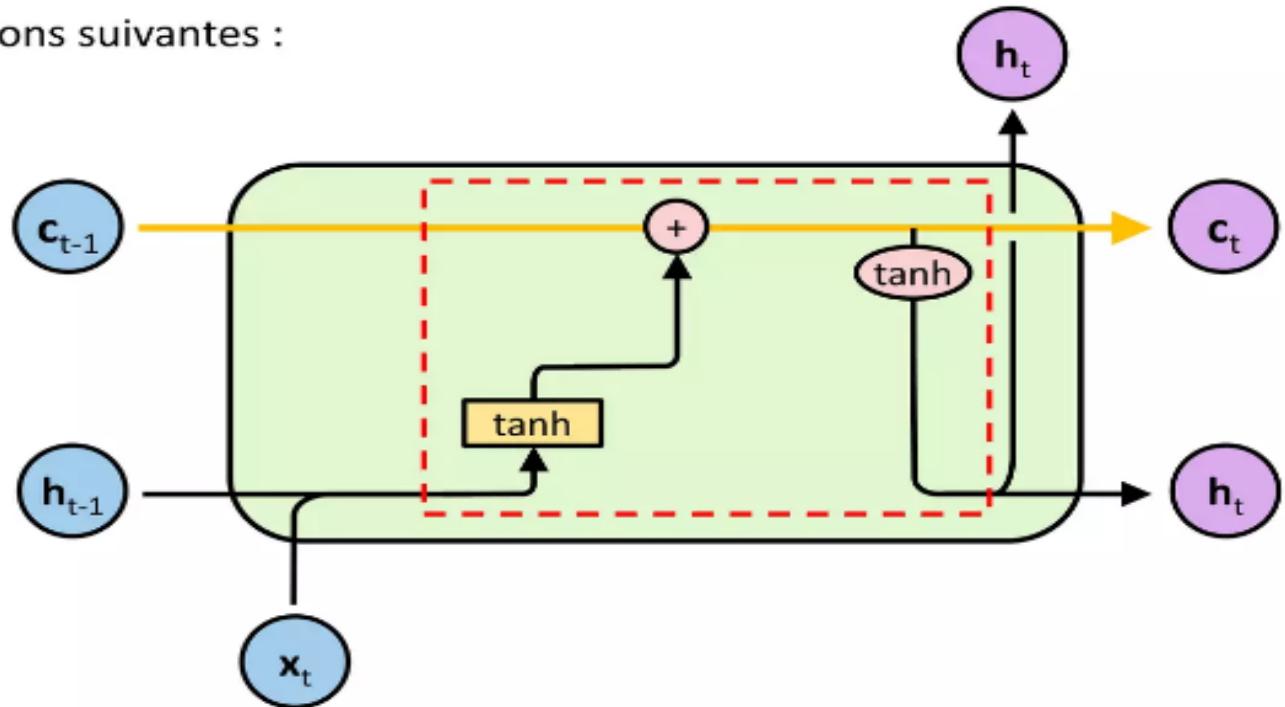
La mise à jour des variables se fera alors selon les équations suivantes :

Equations :

$$c_t = c_{t-1} + \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$h_t = \tanh(c_t)$$

Cette fois, on est assuré que l'état caché h_t calculé à l'instant t tient compte de tous les états cachés précédents.



Long Short Term Memory (LSTM)

Construire la cellule LSTM

Les LSTM sont explicitement conçus pour éviter le problème de dépendance à long terme. On introduit une nouvelle variable c_t qui permettra de se souvenir des informations pendant de longues périodes.

c_t est appelée état de la cellule.

La mise à jour des variables se fera alors selon les équations suivantes :

Equations :

$$c_t = c_{t-1} + \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$h_t = \tanh(c_t)$$

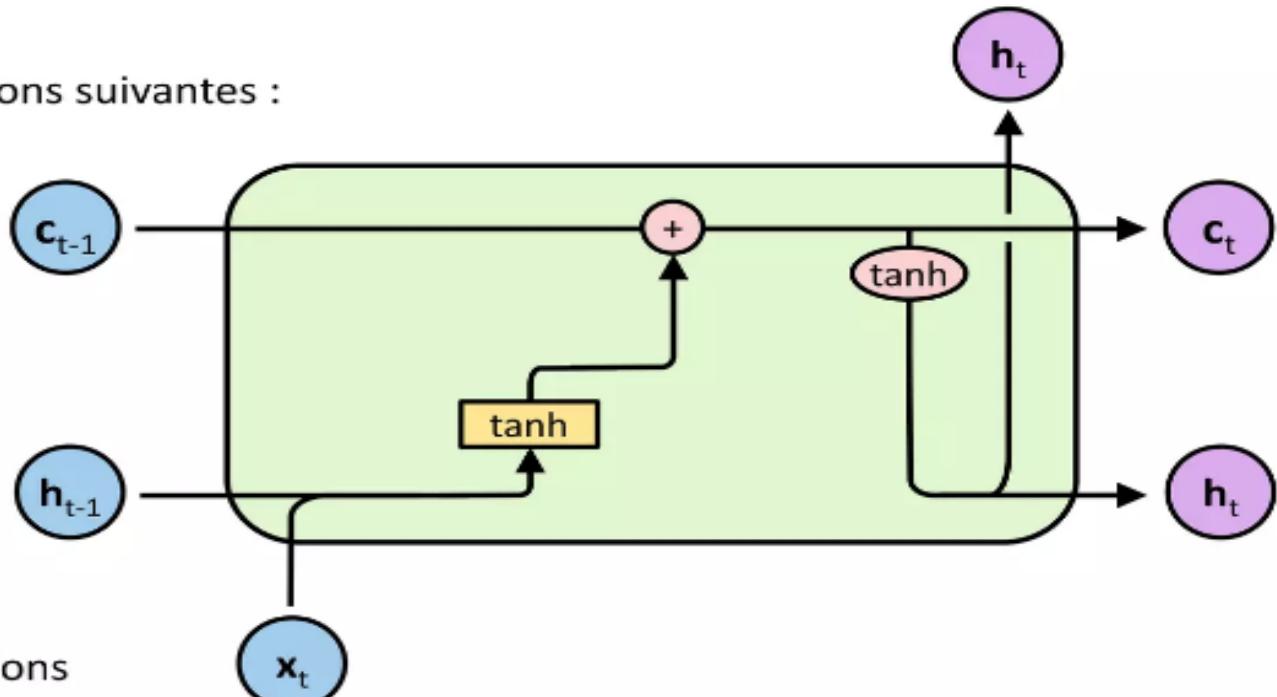
Inconvénient :

Le passé est toujours aussi important que le présent.

Ce qui n'est pas pertinent.

Solution idéale :

Permettre au réseau de garder ou d'oublier les informations du passé selon qu'elles soient déterminantes ou pas.



Long Short Term Memory (LSTM)

Construire la cellule LSTM

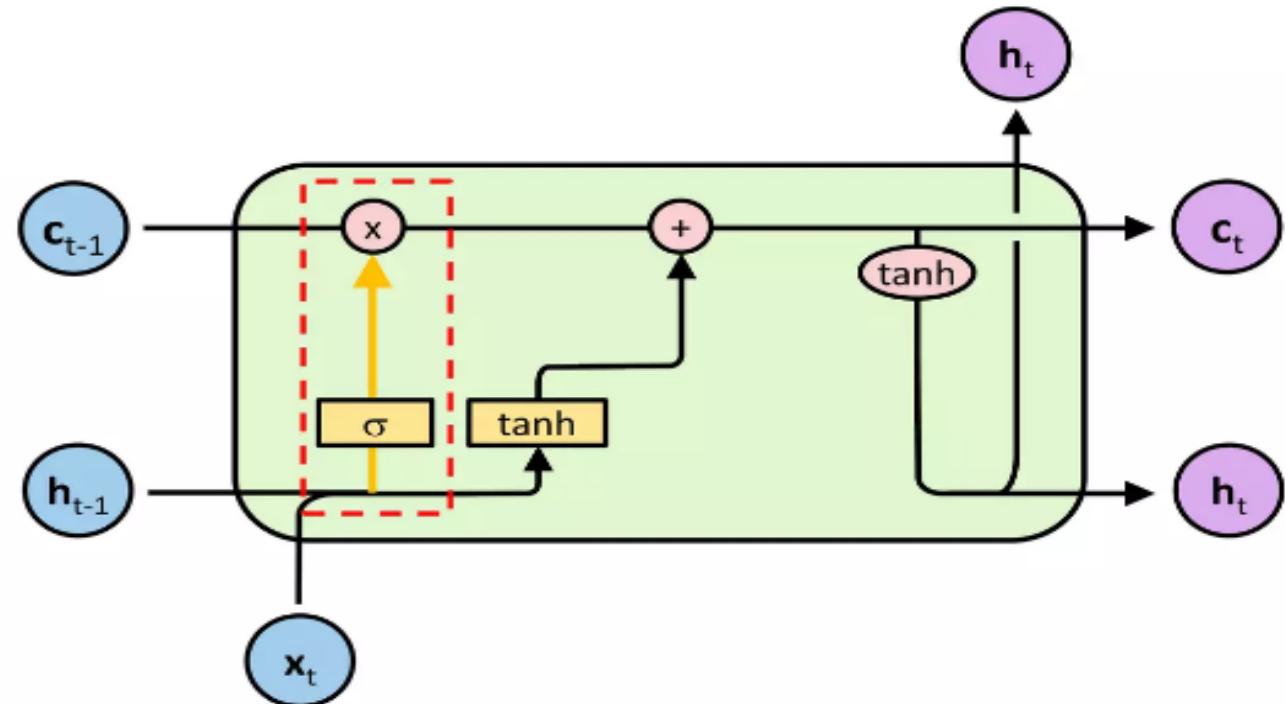
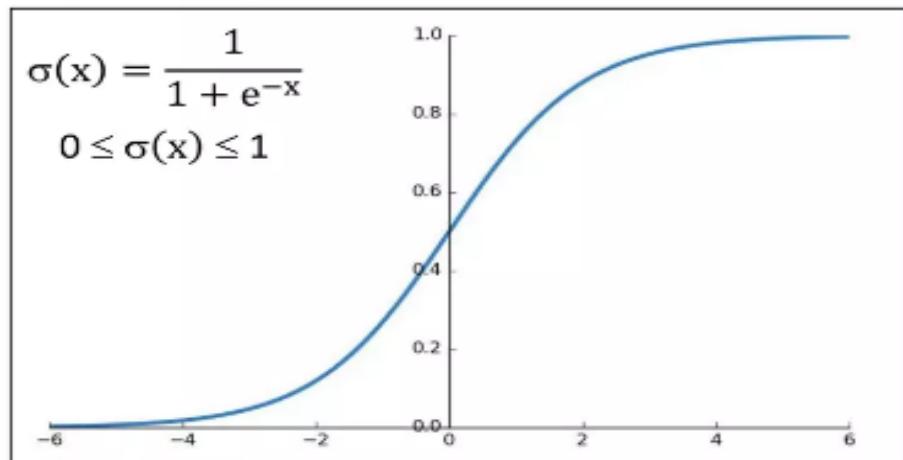
La solution adoptée est de pondérer la mémoire du passé représentée par c_t par un réseau de neurones avec une seule couche sigmoïde. Ce réseau va apprendre à oublier les informations non pertinentes. Il est appelé porte d'oubli ou forget gate.

Equations :

$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t)$$

$$c_t = f_t * c_{t-1} + \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$h_t = \tanh(c_t)$$



Long Short Term Memory (LSTM)

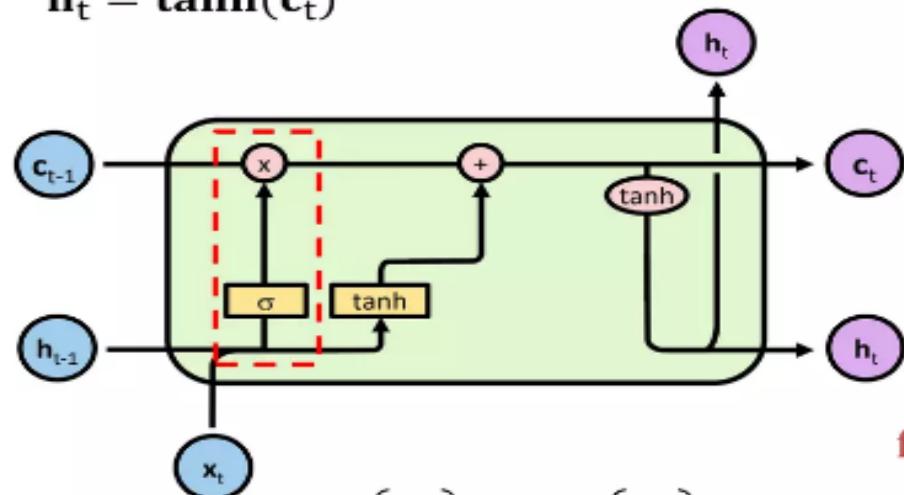
Construire la cellule LSTM

Equations :

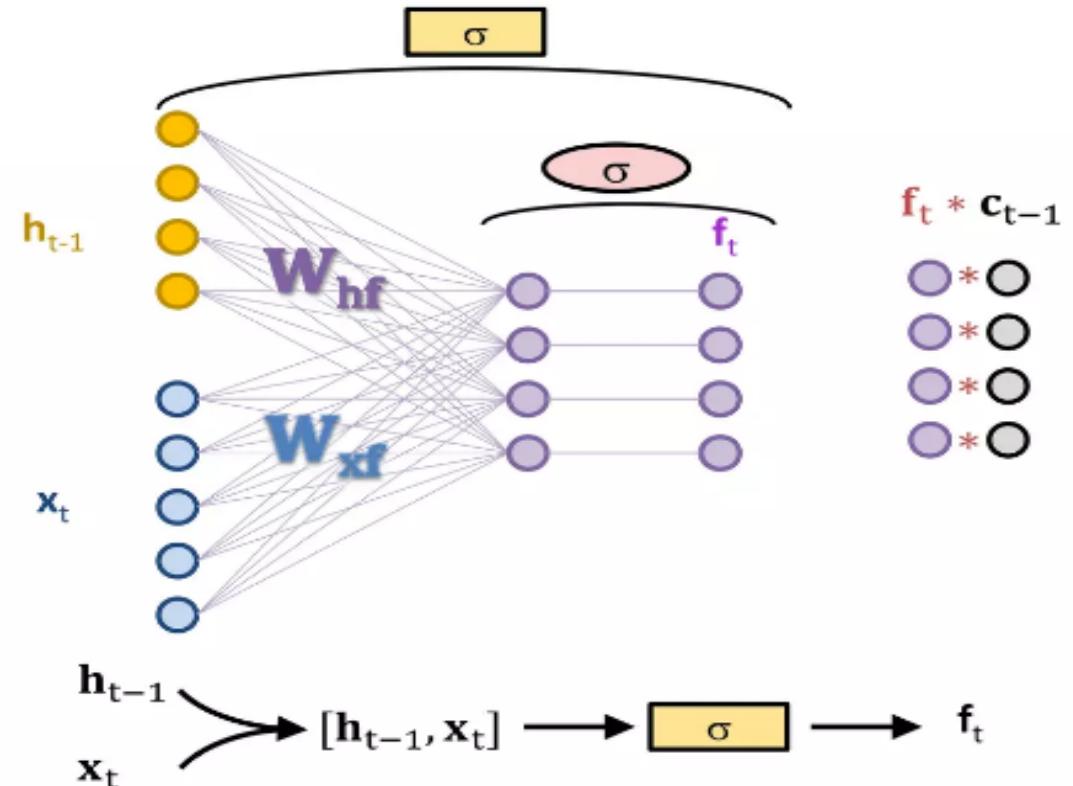
$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t)$$

$$c_t = f_t * c_{t-1} + \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$h_t = \tanh(c_t)$$



$$c_{t-1} = \begin{pmatrix} 0.2 \\ -0.4 \\ 0.75 \\ 0.83 \end{pmatrix} \quad f_t = \begin{pmatrix} 0.9 \\ 0.01 \\ 0.3 \\ 0.95 \end{pmatrix} \rightarrow c_t = \begin{pmatrix} 0.18 \\ 0.00 \\ 0.23 \\ 0.79 \end{pmatrix} \leftarrow \text{oubli}$$



Long Short Term Memory (LSTM)

Construire la cellule LSTM

Une autre porte est ajoutée pour pondérer la mise à jour (additive) de l'état de la cellule c_t par la sortie tanh prenant comme entrée h_{t-1} et x_t . Cette porte qui s'appelle input gate va apprendre à utiliser, à ignorer ou à moduler les informations entrées selon leur importance.

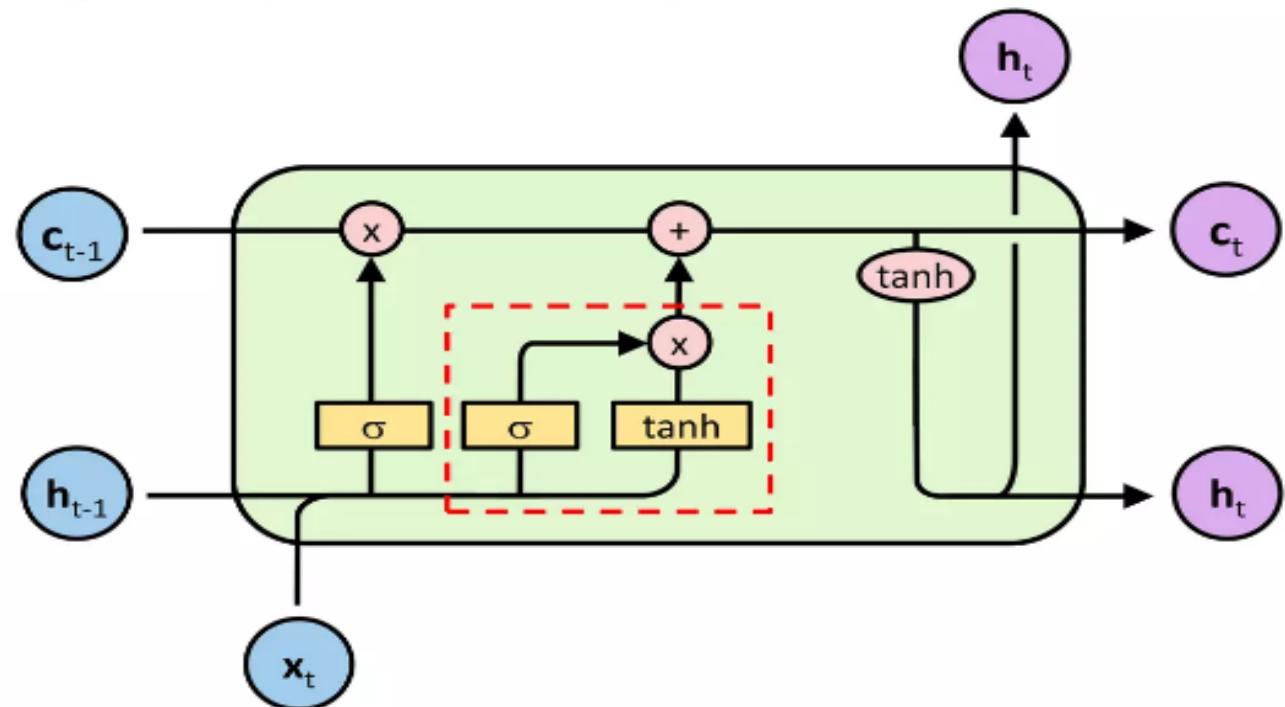
Equations :

$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t)$$

$$i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t)$$

$$c_t = f_t * c_{t-1} + i_t * \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$h_t = \tanh(c_t)$$



Long Short Term Memory (LSTM)

Construire la cellule LSTM

De la même manière, une porte de sortie est ajoutée pour pondérer la mise à jour de l'état caché \mathbf{h}_t . Cela permet de décider quelles informations l'état caché \mathbf{h}_t doit porter. Cette porte qui s'appelle output gate.

Equations :

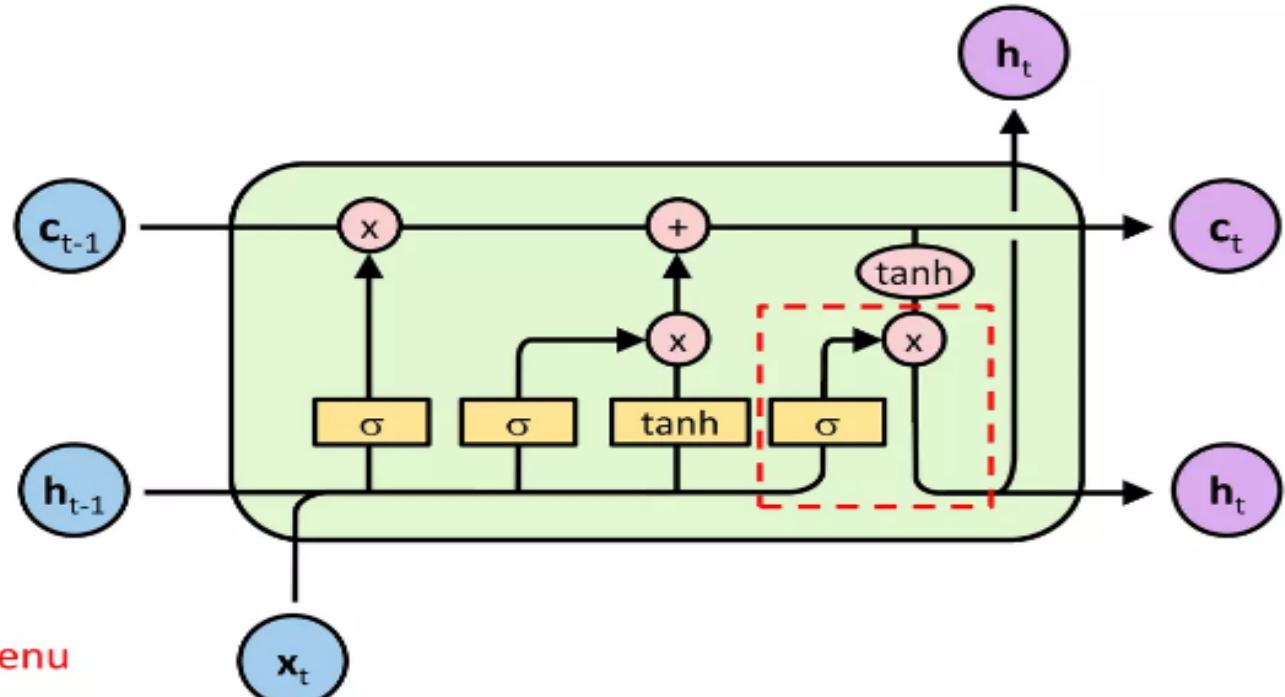
$$\mathbf{f}_t = \sigma(\mathbf{W}_{hf} \mathbf{h}_{t-1} + \mathbf{W}_{xf} \mathbf{x}_t)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{hi} \mathbf{h}_{t-1} + \mathbf{W}_{xi} \mathbf{x}_t)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ho} \mathbf{h}_{t-1} + \mathbf{W}_{xo} \mathbf{x}_t)$$

$$\begin{aligned}\mathbf{c}_t &= \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \tanh(\mathbf{W}_{hh} \mathbf{h}_{t-1} + \mathbf{W}_{xh} \mathbf{x}_t) \\ \mathbf{h}_t &= \mathbf{o}_t * \tanh(\mathbf{c}_t)\end{aligned}$$

$$\tanh(\mathbf{c}_t) = \begin{pmatrix} -0.8 \\ -0.3 \\ 0.64 \\ -0.01 \end{pmatrix} \quad \mathbf{o}_t = \begin{pmatrix} 0.96 \\ 0.1 \\ 0.01 \\ 0.38 \end{pmatrix} \rightarrow \mathbf{h}_t = \begin{pmatrix} -0.77 \leftarrow \text{retenu} \\ -0.03 \\ 0.01 \leftarrow \text{non retenu} \\ 0.00 \end{pmatrix}$$



Long Short Term Memory (LSTM)

LSTM, une synthèse

Forget gate : $f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t)$

Dans le cas extrême, indique à l'état de la cellule (mémoire à long-terme) les informations à oublier (multiplication par 0) ou à conserver (multiplication par 1).

Input gate : $i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t)$

déterminer quelles informations produites par la couche tanh doivent entrer dans l'état de la cellule (donc à sauvegarder dans la mémoire à long terme).

Output gate : $o_t = \sigma(W_{ho} h_{t-1} + W_{xo} x_t)$

Indique les informations qui doivent passer au prochain état caché h_t .

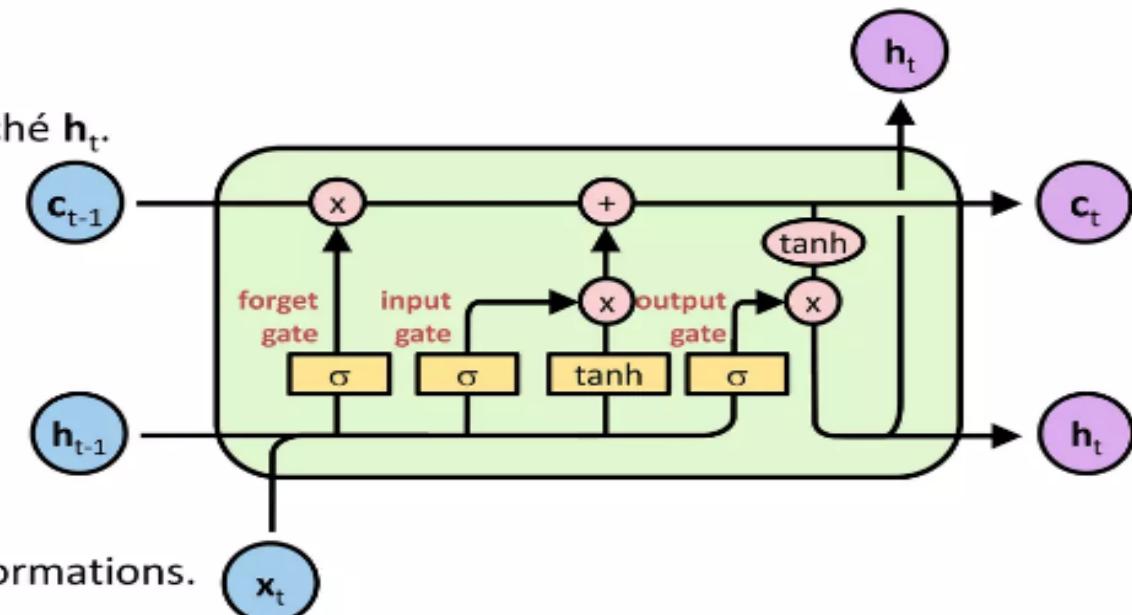
Calcul de c_t et de h_t :

$$c_t = f_t * c_{t-1} + i_t * \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$h_t = o_t * \tanh(c_t)$$

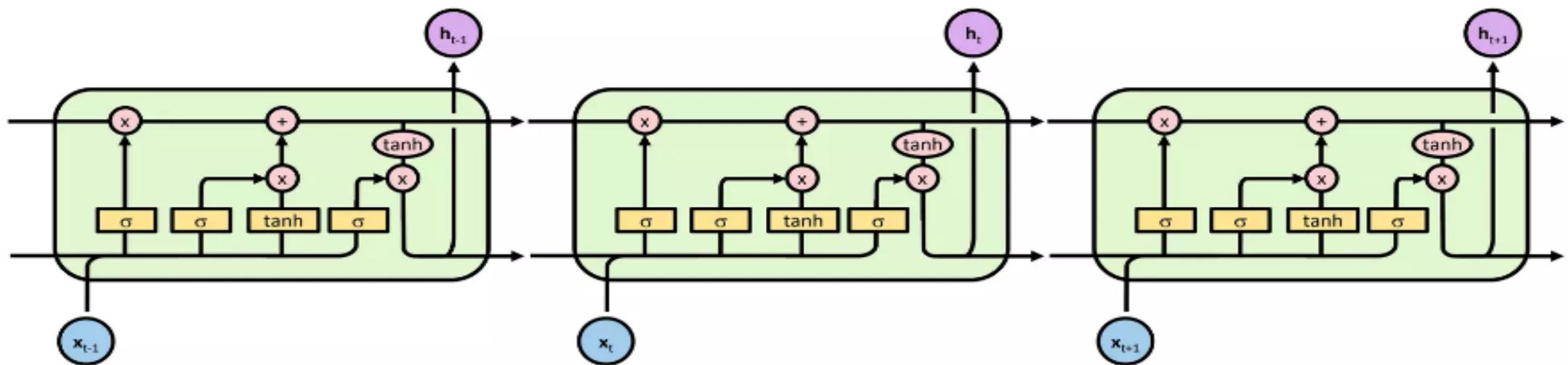
L'état de la cellule c_t parcourt toute la chaîne du réseau avec des interactions linéaires mineures, ce qui leur permet de transporter efficacement les informations.

Les portes (gates) permettent de réguler intelligemment ces informations.



Long Short Term Memory (LSTM)

Réseau LSTM

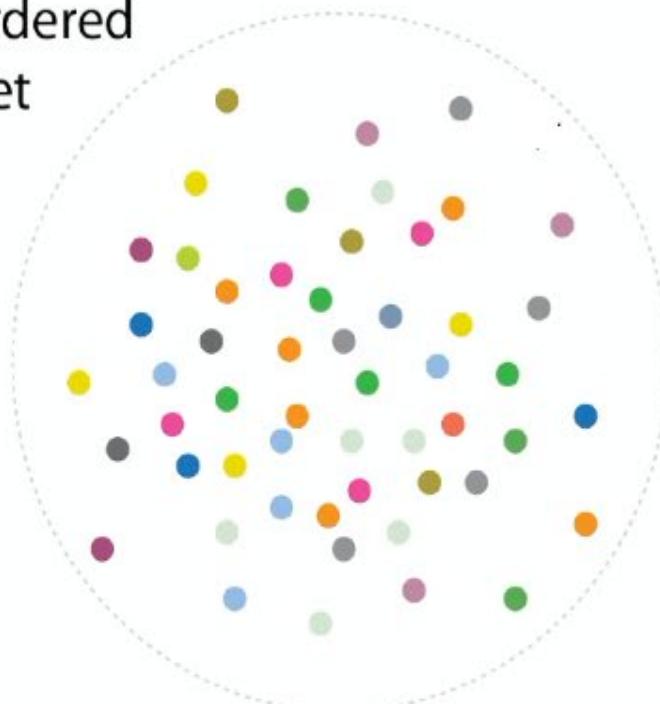


Algorithmes de deep learning

Les réseaux de neurones récurrents

Préparation des données

Not ordered
Dataset



Train set



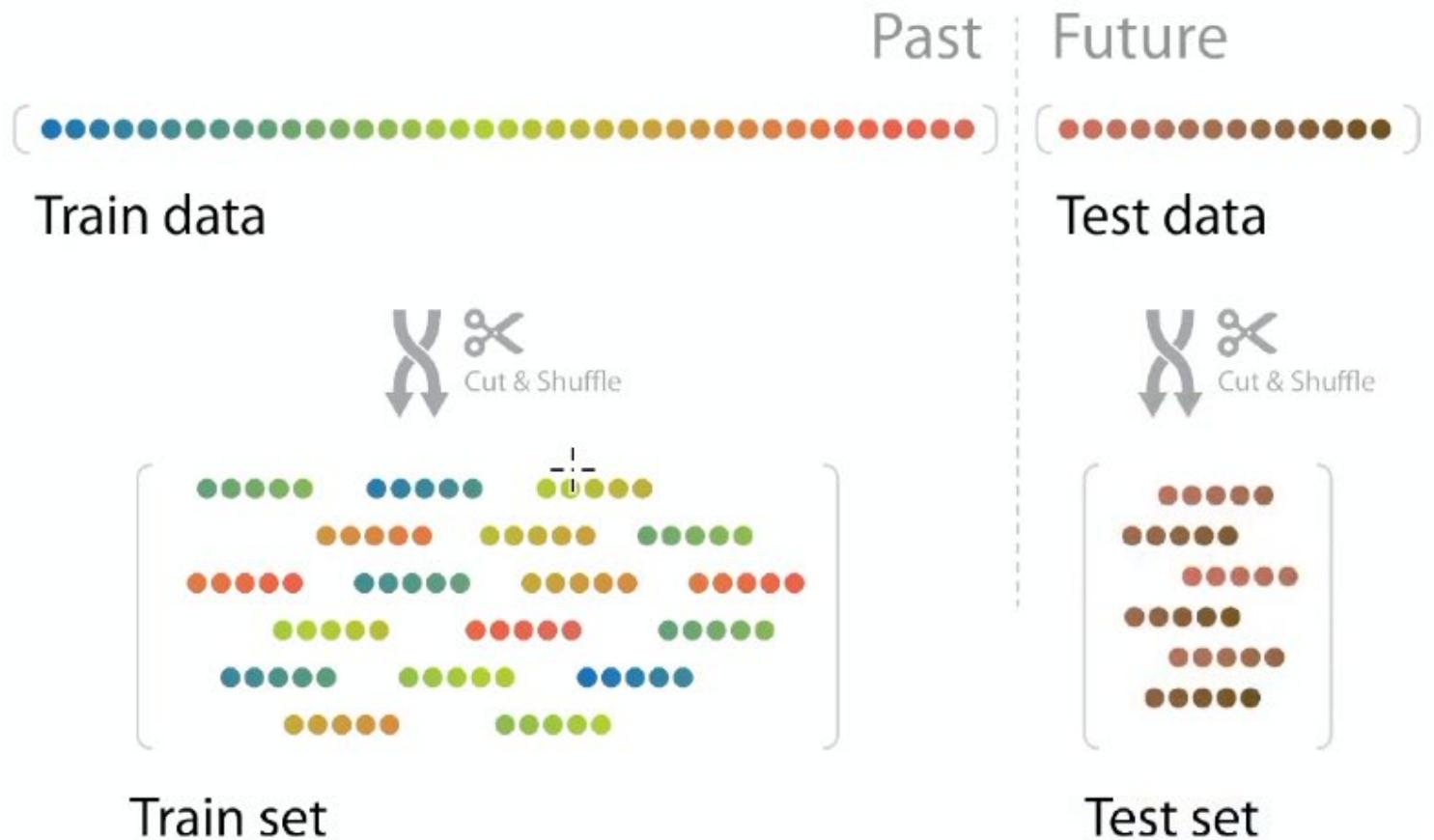
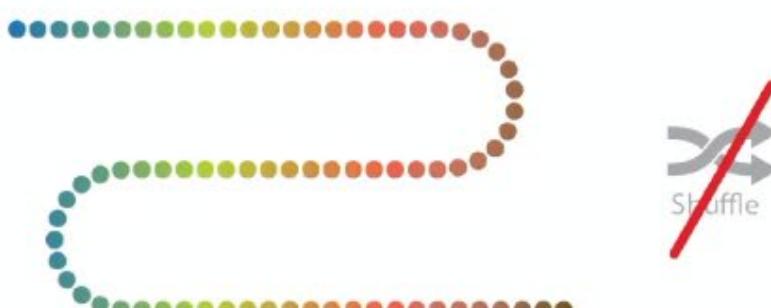
Test set

Algorithmes de deep learning

Les réseaux de neurones récurrents

Préparation des données

Ordered dataset



Note that a data generator should be very useful !

Implémentation pratique avec Python et sklearn, tensorflow, keras (TP – RNN-LSTM)

Auto-Encodeurs

Définition

Un Auto-encodeur est un réseau de neurones artificiels qui est souvent utilisé dans l'apprentissage des caractéristiques discriminantes d'un ensemble de données. Il peut être vu comme l'ensemble d'un encodeur et décodeur.

En effet, l'encodeur est constitué par un ensemble de couches de neurones, qui traitent les données afin d'obtenir une nouvelle représentation des données tandis que les couches de neurones du décodeur analysent les données encodées pour essayer de reconstruire les données d'origines. Généralement, la nouvelle représentation des données a moins de caractéristiques. Ce qui permet de réduire la dimensionnalité des données d'origines. La différence entre les données d'origines et les données reconstruites par le décodeur permet d'évaluer l'erreur de reconstruction. Le but de l'entraînement de l'auto-encodeur est de modifier les paramètres afin de minimiser l'erreur de reconstruction.

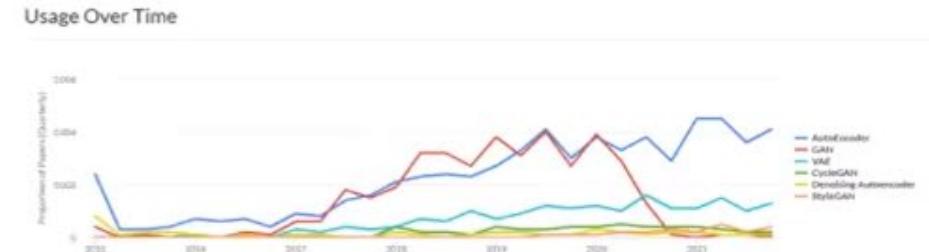
Il ne nécessite pas de données labelisées, et peut ainsi être utilisé pour réduire l'effort de labélisation des données.

Algorithmes de deep learning

Auto-Encodeurs

Domaine d'application

Travaux scientifiques sur les Autoencoder sont de plus en plus utilisé récemment dans: Détection d'anomalie; Denoising; Séries temporelles; Réduction des dimensions; Génération des images; ...



Denoising



Transformation de la voix



Génération d'image



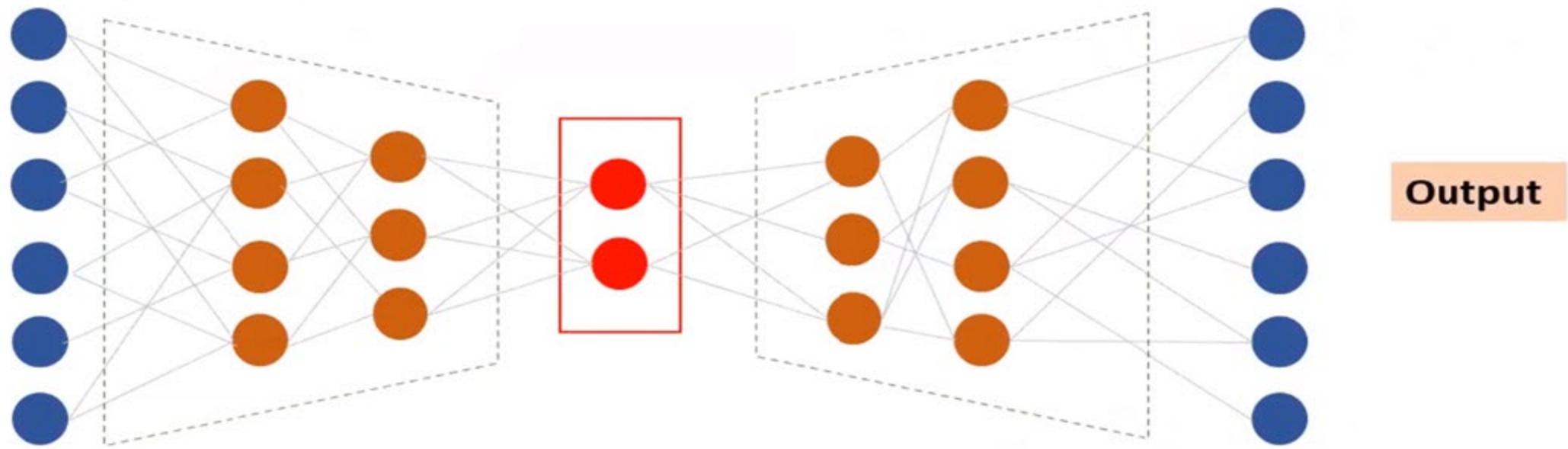
Détection d'anomalie



Algorithmes de deep learning

Auto-Encodeurs

Architecture

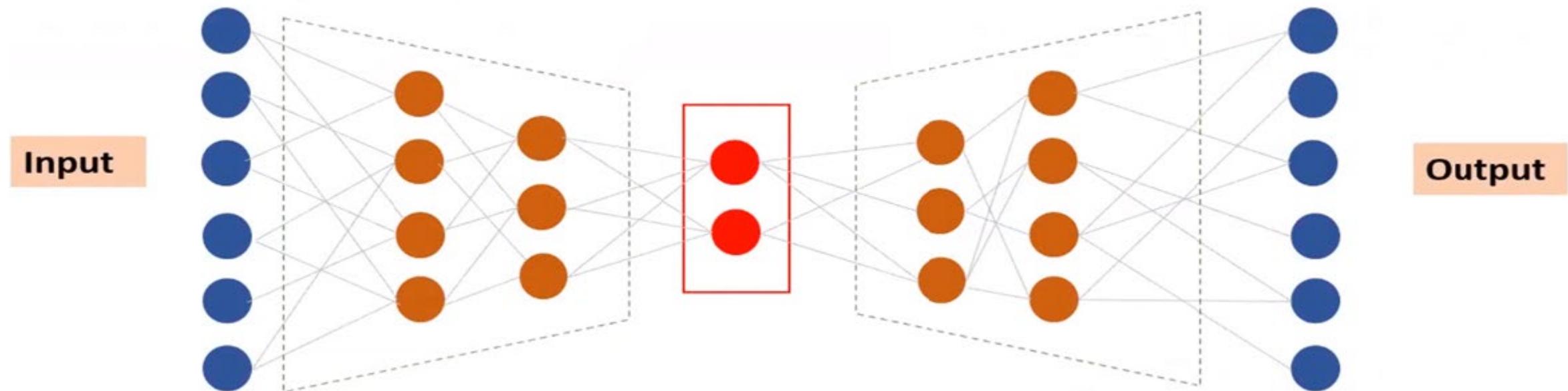


AE est un réseau de neurones profond, mais la particularité qui le différencier des autres modèles, ce qu'il permet d'obtenir une reconstruction de l'entrée au lieu d'une sortie boolean ou encore des valeurs définis par Softmax... etc comme le cas des réseaux de neurones classiques.

Auto-Encodeurs

Architecture

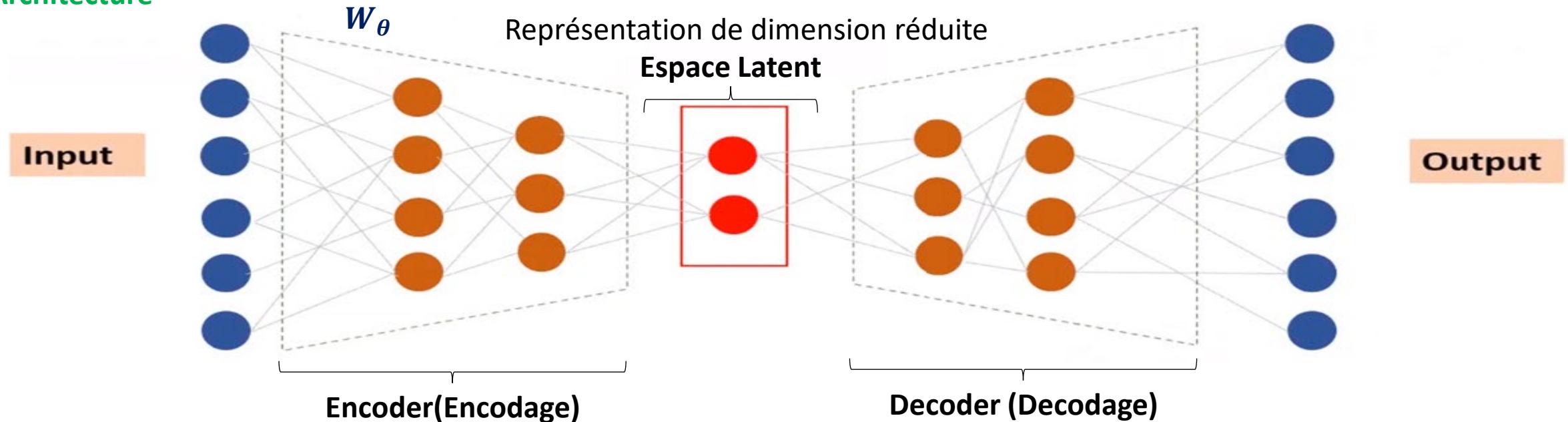
L'architecture AE se compose de trois blocs principales, la figure ci-dessous montre l'architecture d'un auto-encodeur, elle est constituée des couches de neurones de codage et de décodage. Les couches de codage compriment les données d'origines d'entrée pour permettre d'avoir une représentation compressée des données. Tandis que les couches de décodage essayent de reconstruire les données d'origines à partie des données compressées



Algorithmes de deep learning

Auto-Encodeurs

Architecture



Encoder plusieurs couches de neurones qui permet l'agrégation des valeurs d'entrées par le moyen de la matrice des poids W_θ

Espace Latent il s'agit d'une réduction ou d'une compression des informations d'entées

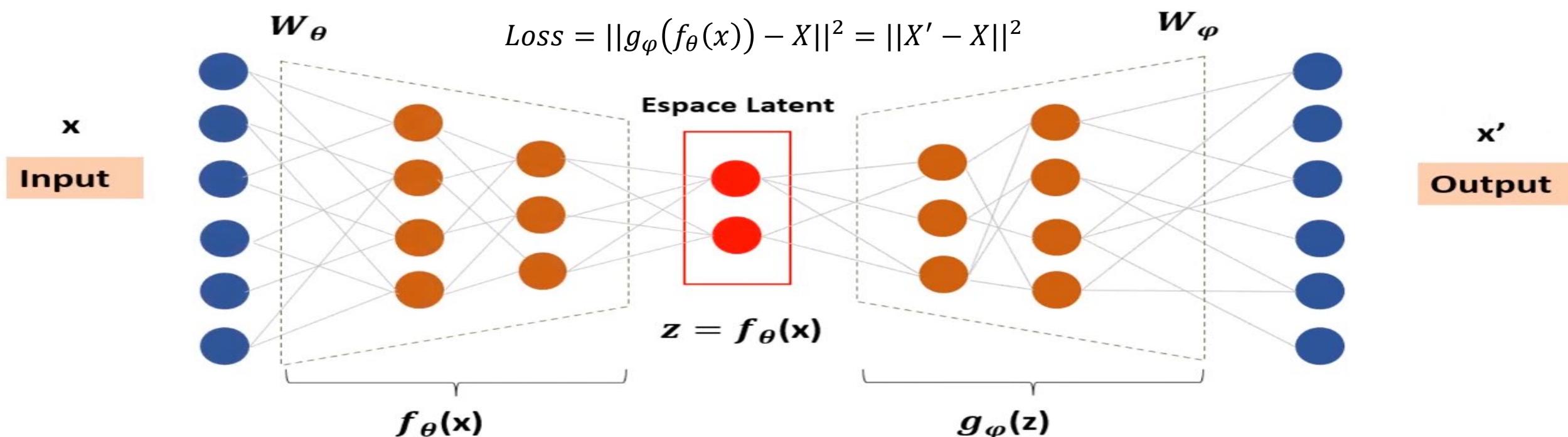
Decoder il s'agit d'une partie symétrique inverse de la partie **Encoder** qui permet de reconstruire l'entrée à partir l'entrée obtenue dans l'espace latent par le moyen de la matrice des poids W_θ qui sont différent de celui de l'**Encoder**

Algorithmes de deep learning

Auto-Encodeurs

Architecture

- Idée est de poser : $x=x'$



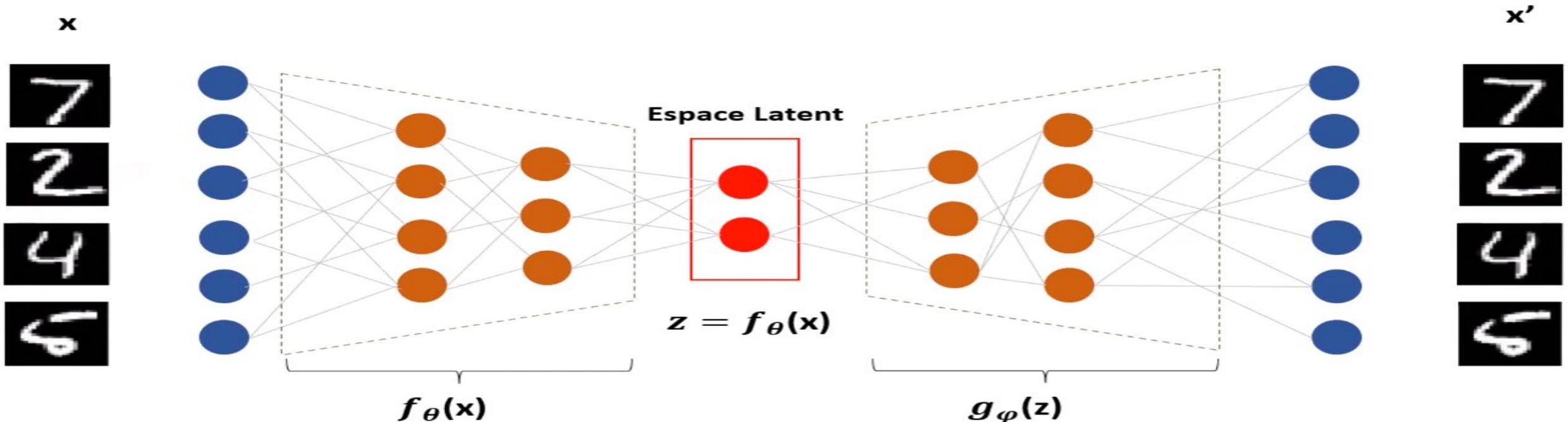
Si on veut que le EA nous fournit une reconstruction de l'image d'entrée on peut mettre la même image aussi dans la sortie . Dans le processus d'apprentissage, EA va converger en minimisant la fonction de perte (**Loss**)

L'implémentation de Auto-Encodeurs peut se faire soit avec les réseaux de neurones **simples** soit avec **CNN**

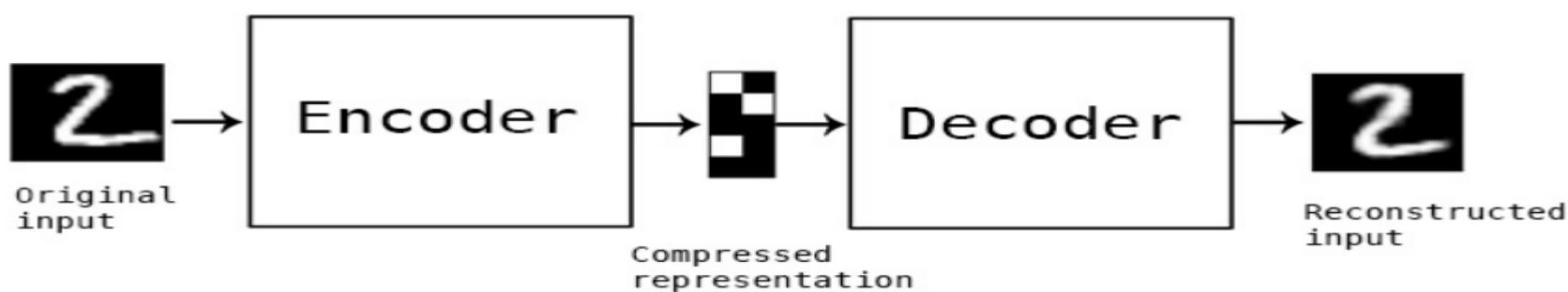
Algorithmes de deep learning

Auto-Encodeurs

Application sur MNIST



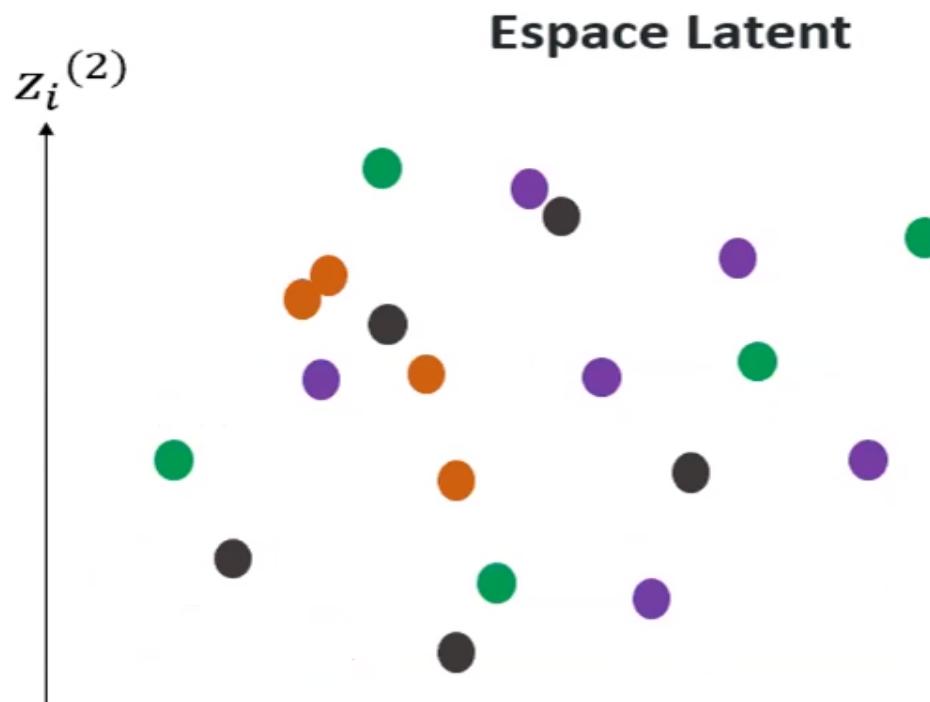
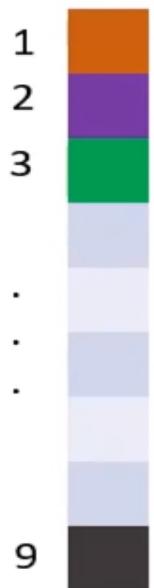
Le **Decoder** sera une symétrie de le **Encoder** c-à-dire, si l'architecture de **Encoder** est 4:3:2 alors le **Decoder** sera de 2:3:4



Auto-Encodeurs

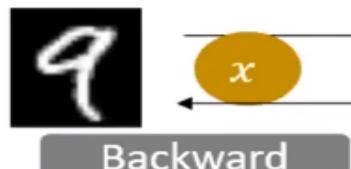
Application sur MNIST

1^{ère} itération:



- Le rôle de l'espace Latent est de réduire ou de compresser les images et de retenir les informations pertinentes, et puis générer une reproduction de l'image.
- Les images d'entrées sont de dimension $28 \times 28 = 784$, et sont réduites à un espace 2 dimensions

Input



Forward

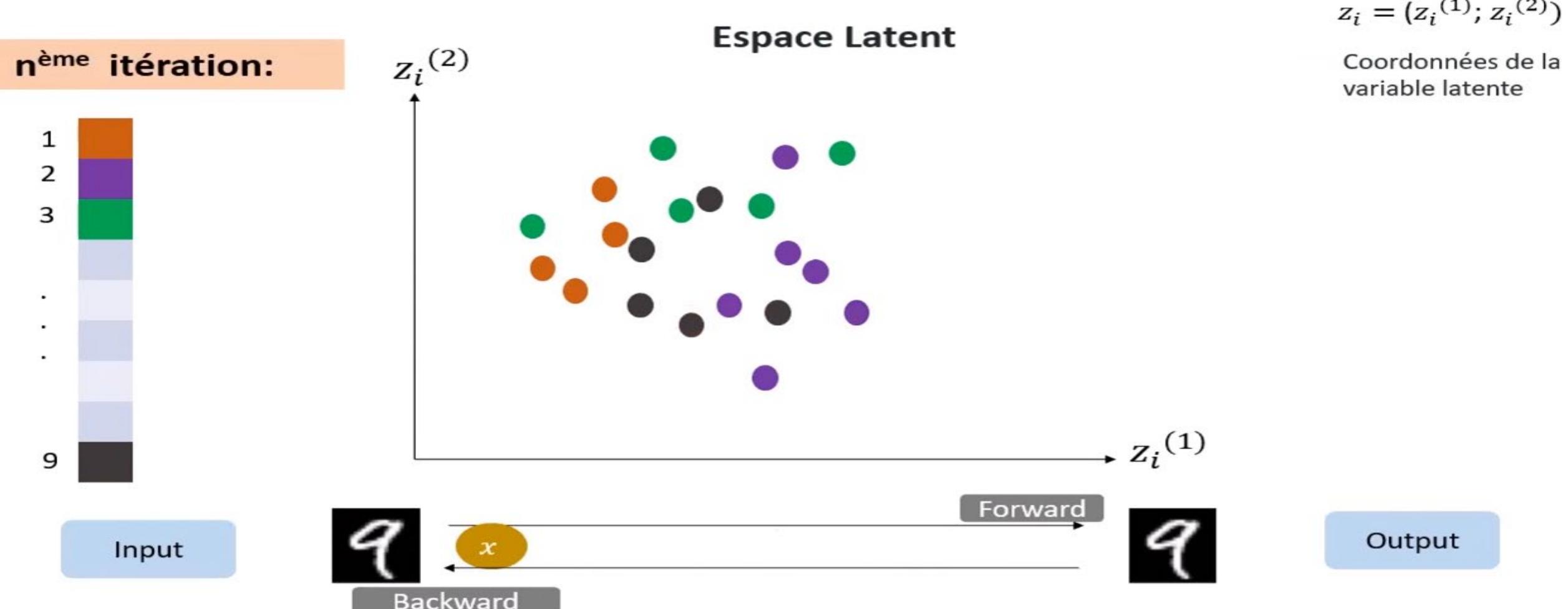


Output

Algorithmes de deep learning

Auto-Encodeurs

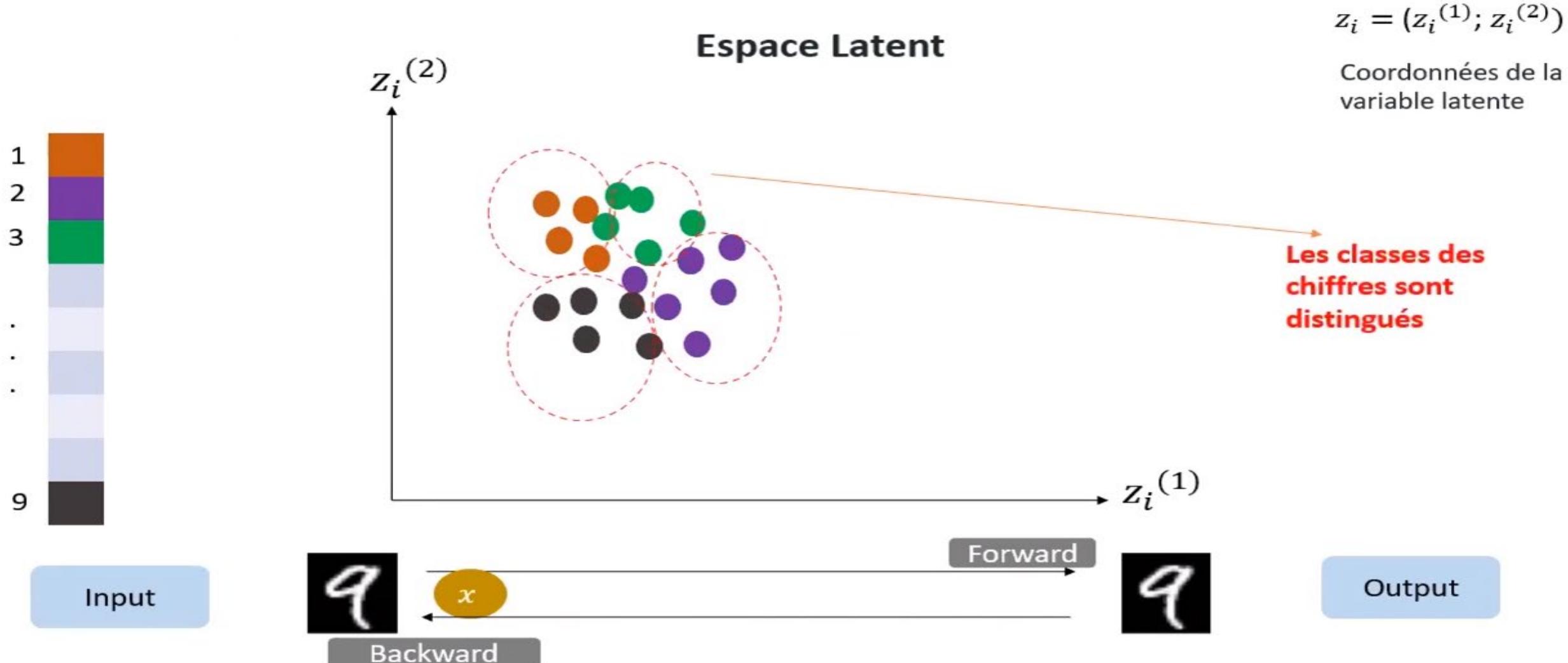
Application sur MNIST



Algorithmes de deep learning

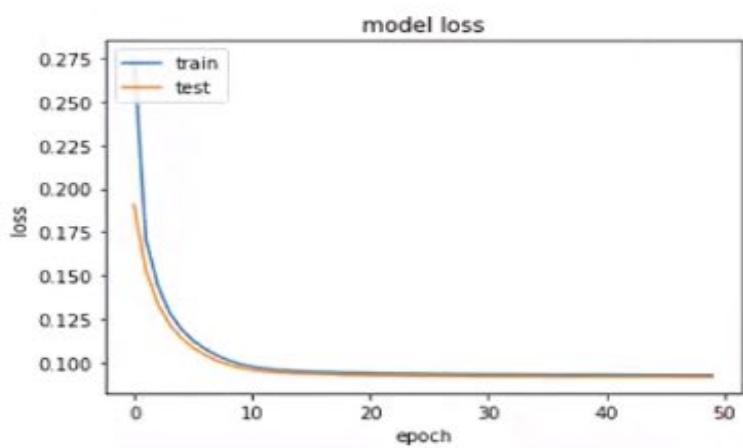
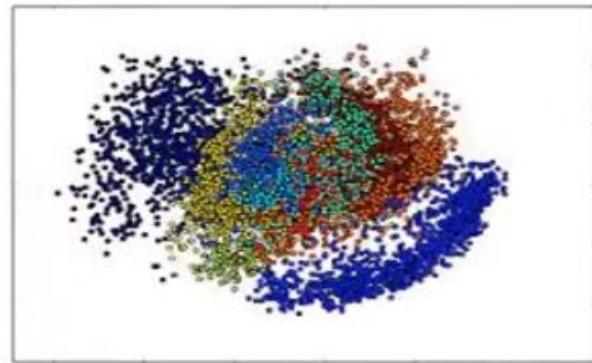
Auto-Encodeurs

Application sur MNIST



Auto-Encodeurs

Application sur MNIST



Loss Function:

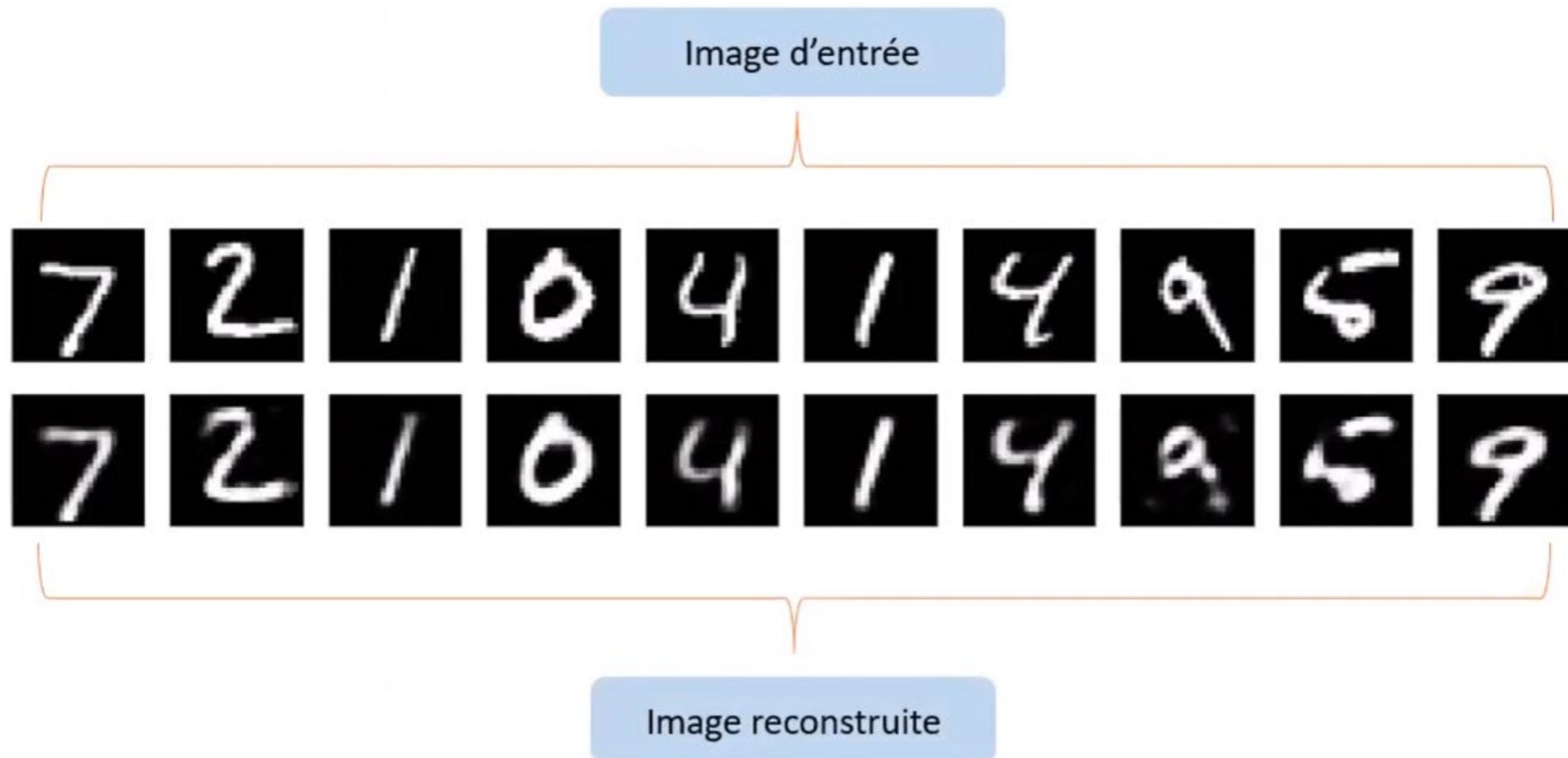
$$\text{Loss} = \|g_\varphi(f_\theta(x)) - x\|^2$$

- On met pour l'entrée les données de MNIST pour l'apprentissage du modèle Autoencoder, et puis on pose les même images des chiffres dans la sortie.
- Durant le processus d'apprentissage, le modèle réduit les informations des images d'entrée en un espace latent, puis il sépare les différents classes des chiffres (de 1 à 9) selon les coordonnées.
- La fonction de **Loss** utilisée est minimisée de telle façon à $x \approx x'$
- En visualisant l'espace latent, on peut distinguer les classes des chiffres par les coordonnées de l'espace latent.

Algorithmes de deep learning

Auto-Encodeurs

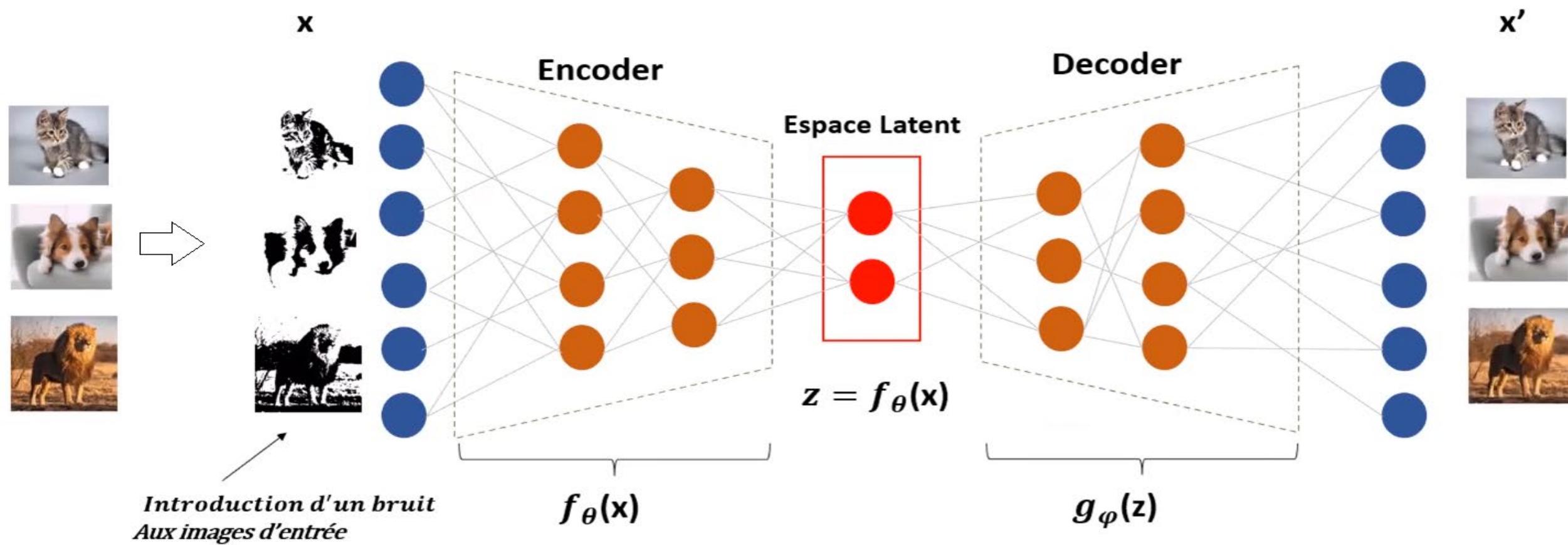
Application sur MNIST



Algorithmes de deep learning

Auto-Encodeurs

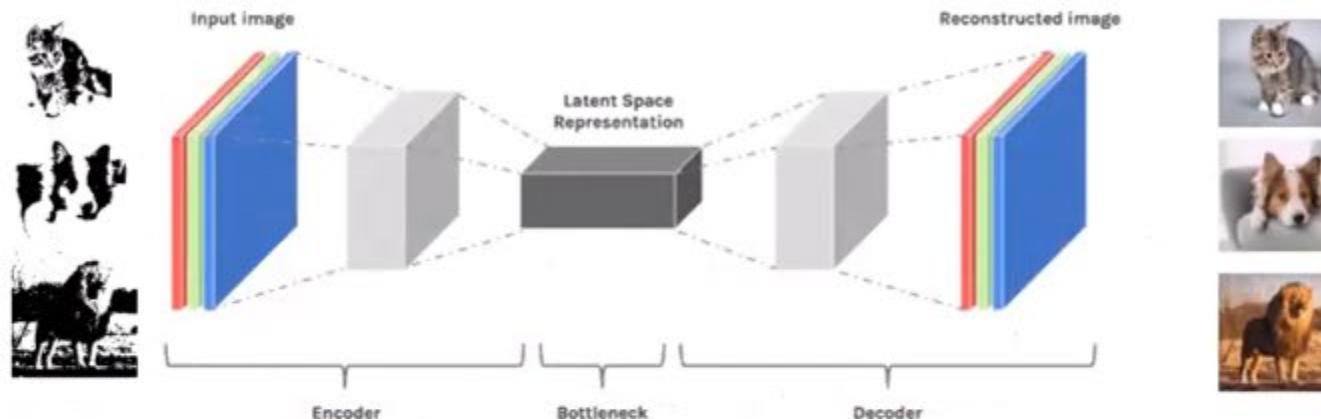
Application sur image Denoising



Algorithmes de deep learning

Auto-Encodeurs

Application sur image Denoising



- On applique un bruit aux images originales avant la phase de l'apprentissage.
- On met dans la sortie les mêmes images originales (sans bruit)
- L'Autoencoder apprend à faire le lien entre les images modifiées et les bonnes images
- L'espace Latent permet de réduire ou de condenser les images et retenir les informations pertinentes

Implémentation pratique avec Python et sklearn, tensorflow, keras (TP – Auto-Encoders)