
بسم الله الرحمن الرحيم

Université Constantine 2
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département de l'Informatique Fondamentale et ses Applications

Module

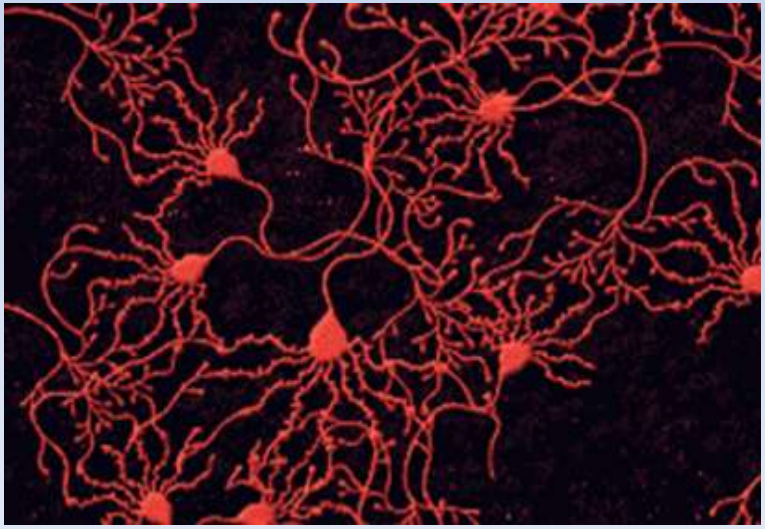
Machine Learning and Computational Intelligence

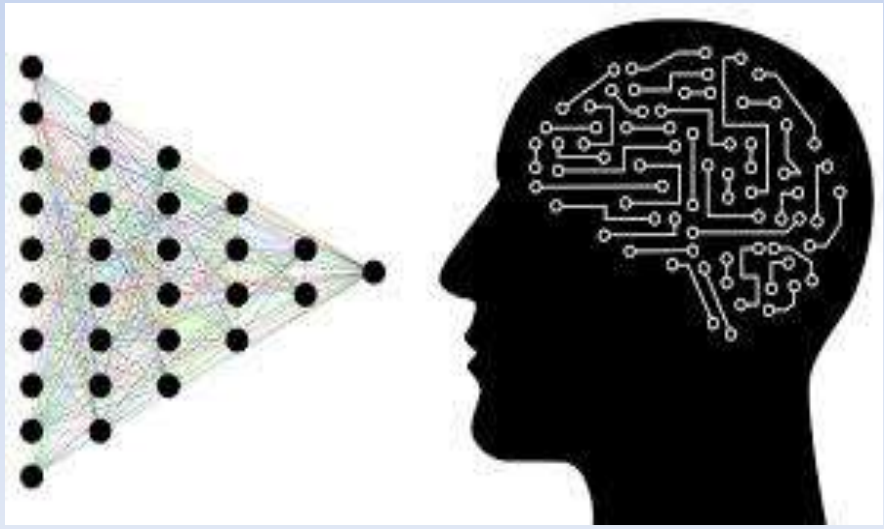
MLCI

Unité d'enseignement: UEF3
Crédit: 3
Coefficient: 3
Cours:1H30/semaine
TP: 1H30/semaine

Dr. Fergani
Baha.fergani@univ-constantine2.dz

Apprentissage profond
(Deep Learning)

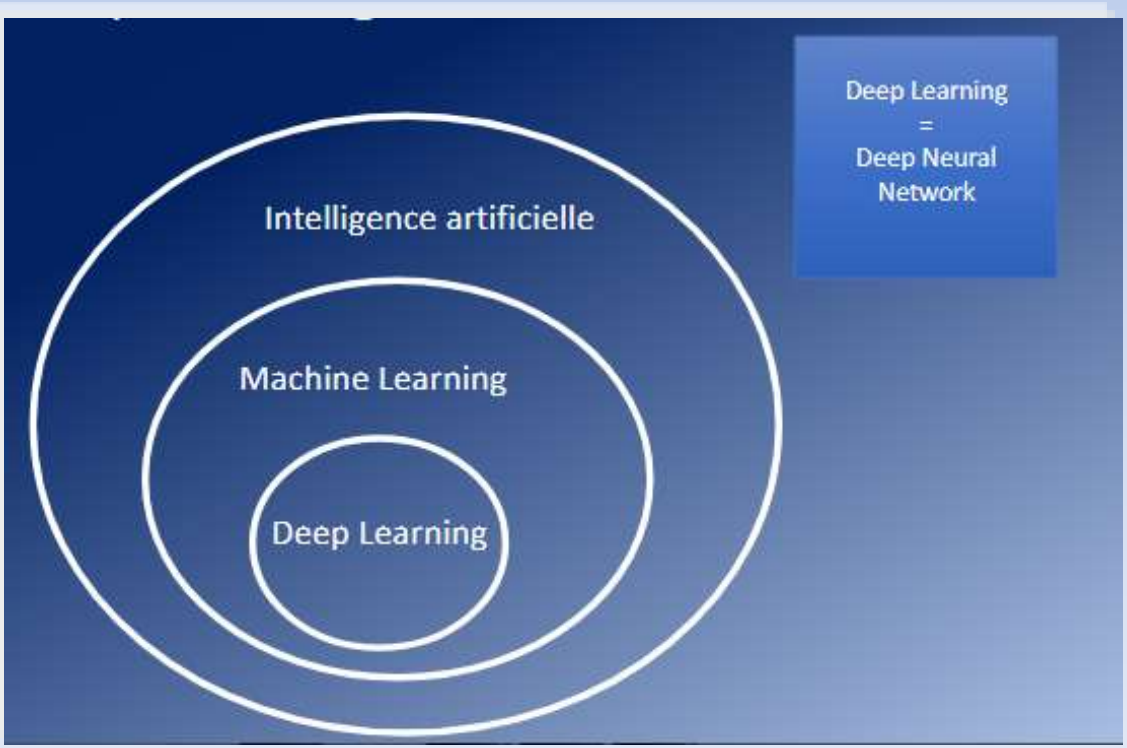




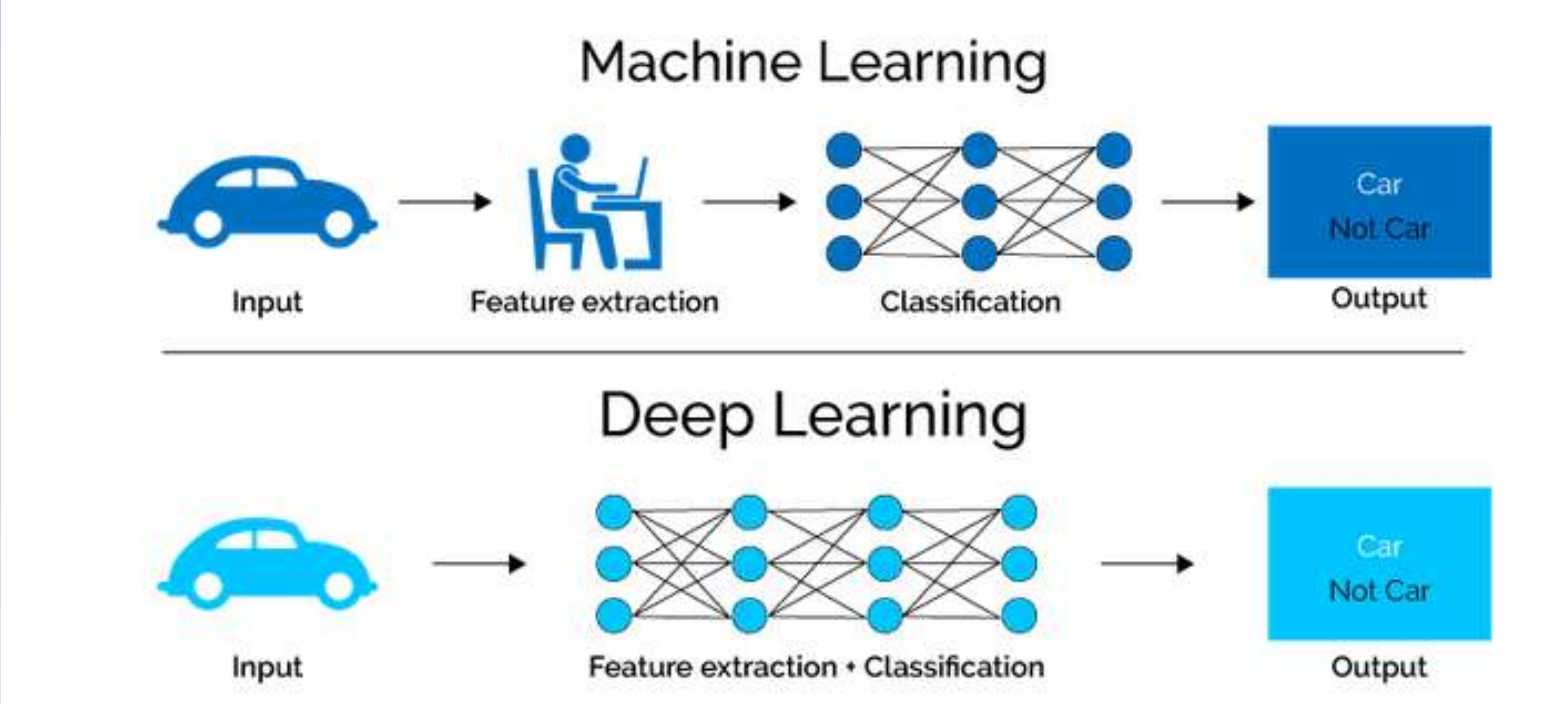
Apprentissage profond

L'apprentissage profond est:

- Une forme d'apprentissage automatique.



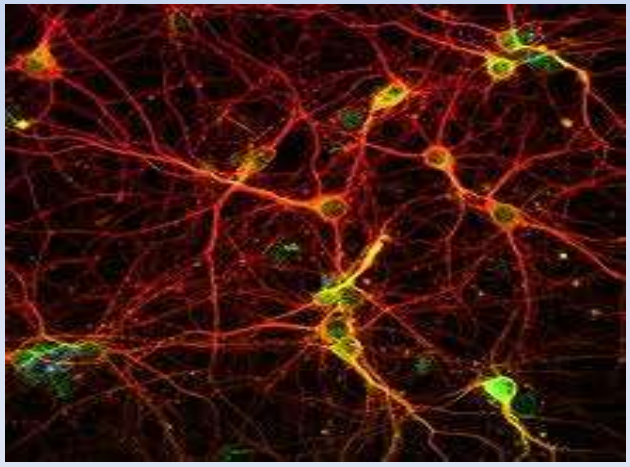
Apprentissage automatique
&
apprentissage profond

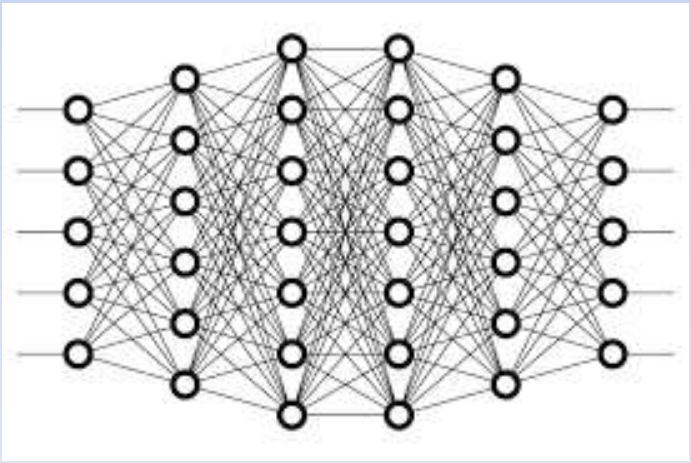


Apprentissage profond

L'apprentissage profond est:

- Une forme d'apprentissage automatique.
- Il utilise les réseaux de neurones artificiels.

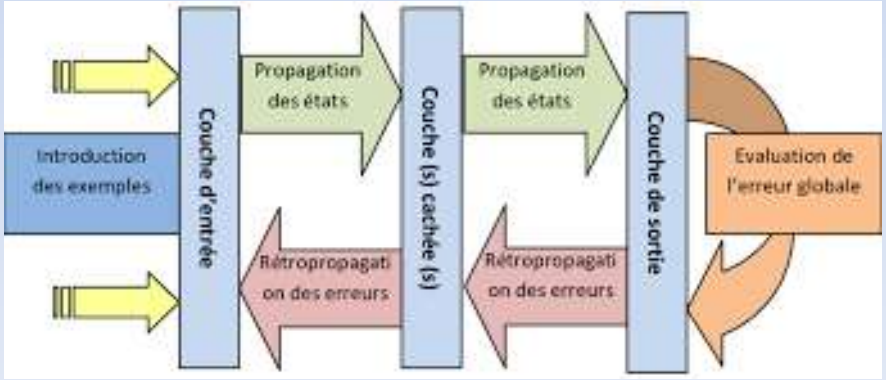




Apprentissage profond

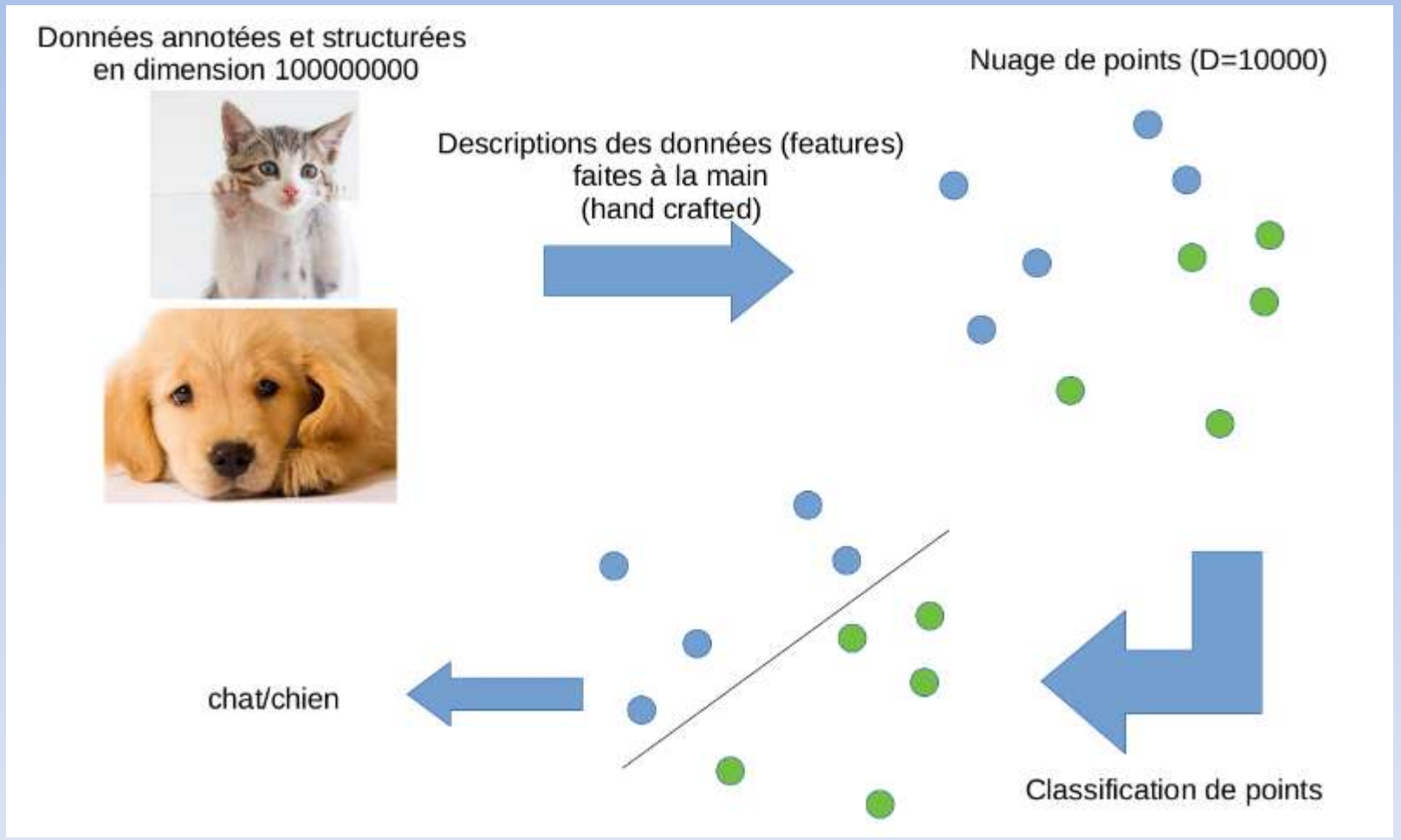
Sa particularité est de:

Découvrir des **structures complexes** dans de **grands** ensembles de données par l'utilisation d'algorithmes de **rétro-propagation**.



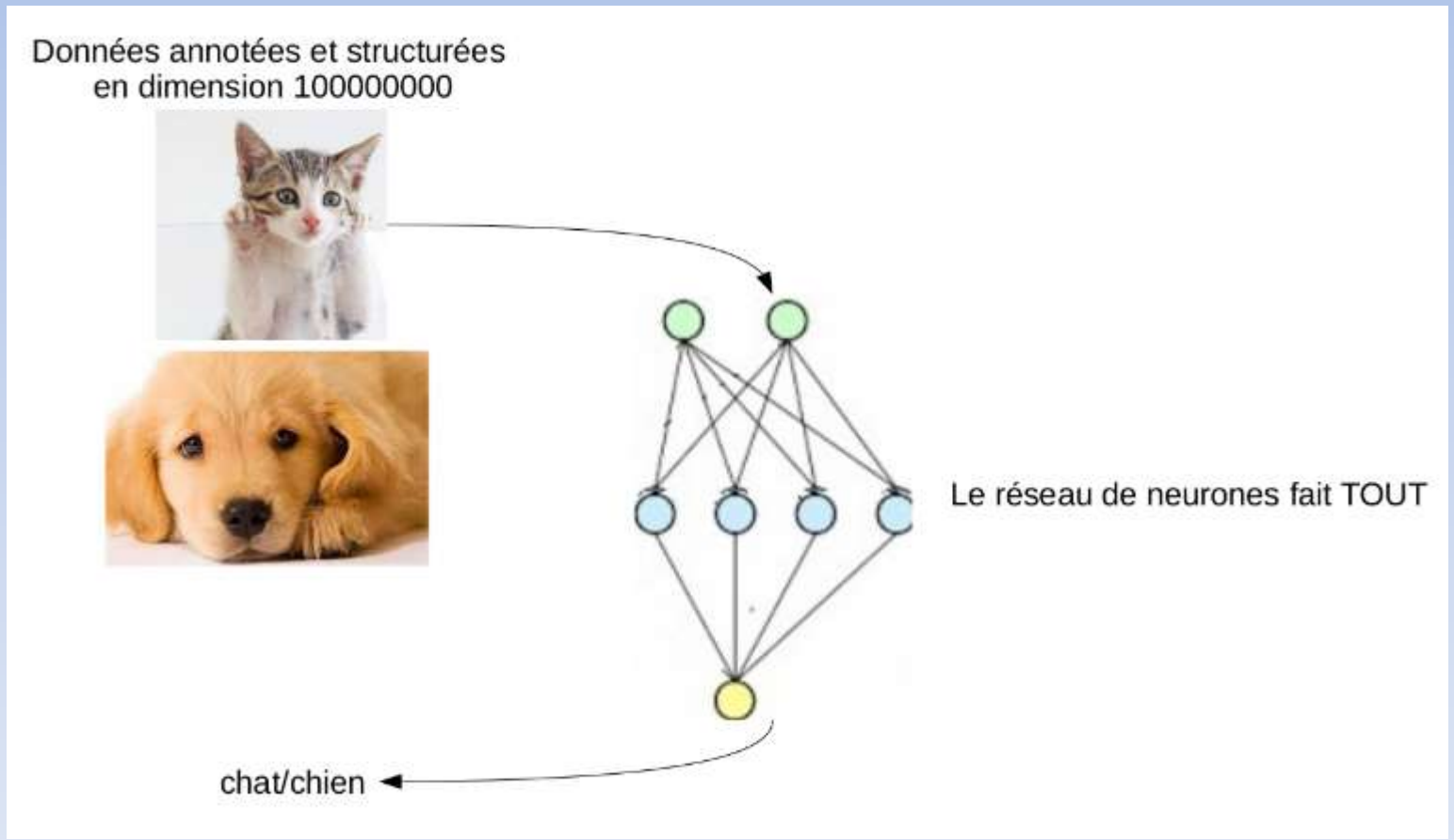
Apprentissage profond

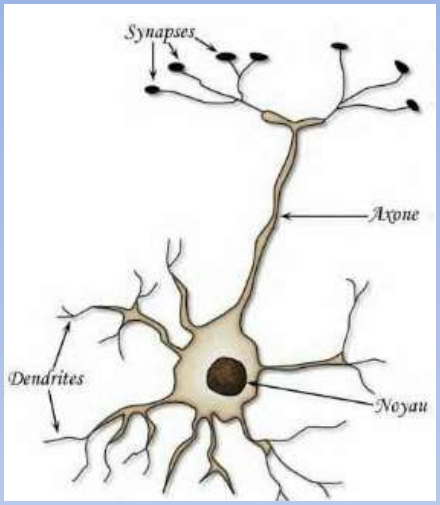
Avant l'apprentissage profond

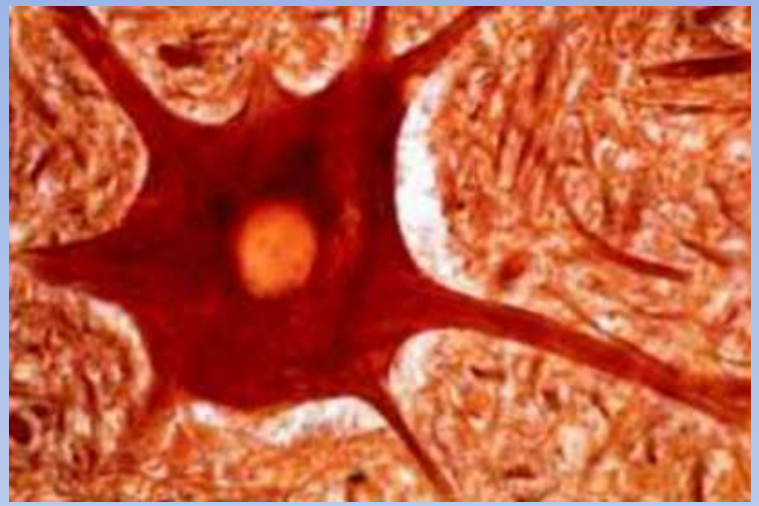


Apprentissage profond

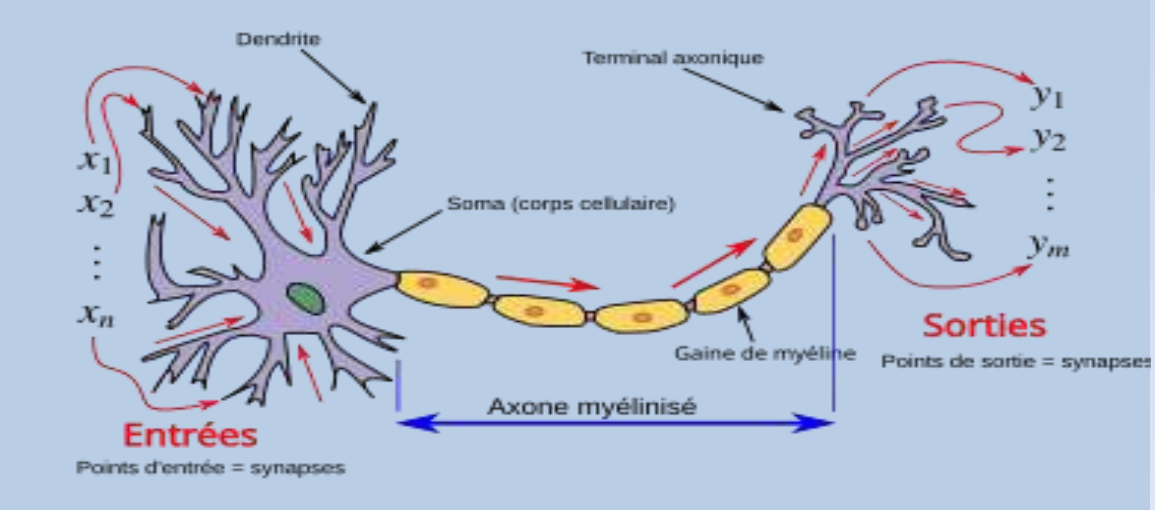
Après l'apprentissage profond







Réseaux de neurones

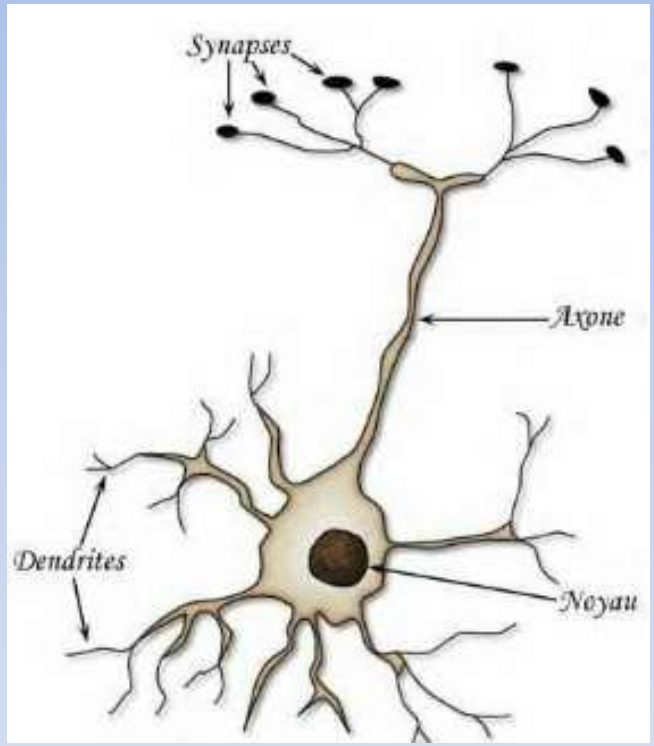


Neurone biologique

Un **neurone** est:

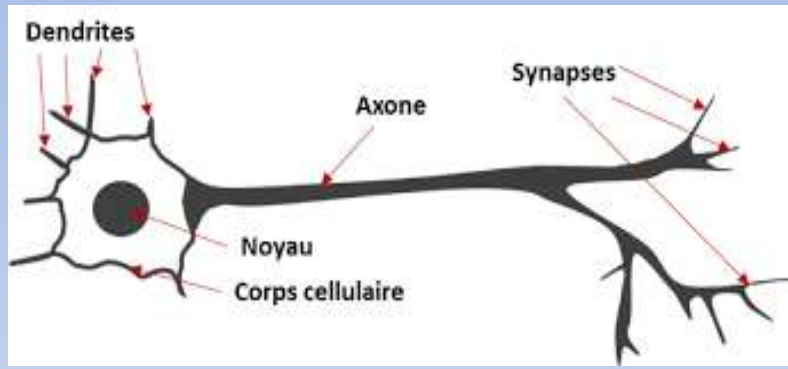
- Une **cellule nerveuse.**
- Une cellule excitable.
- Elle constitue l'unité fonctionnelle

du système nerveux.



Les neurones assurent La transmission d'un signal bioélectrique appelé **influx nerveux**.

Neurone biologique



Le neurone est composé:

- Un **corps** cellulaire appelé **soma**.
- Deux types de prolongements :

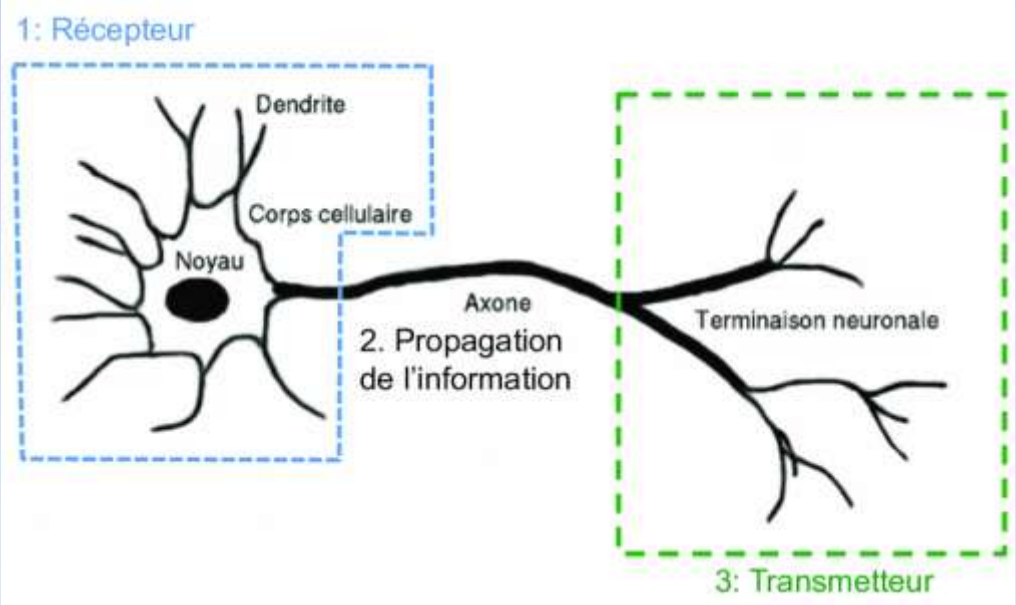
1. L'**axone** (unique): c'est l'émetteur.
2. Les **dendrites**: les récepteurs du potentiel d'action.

•Les connexions entre deux neurones se font via les **synapses.**

Neurone biologique

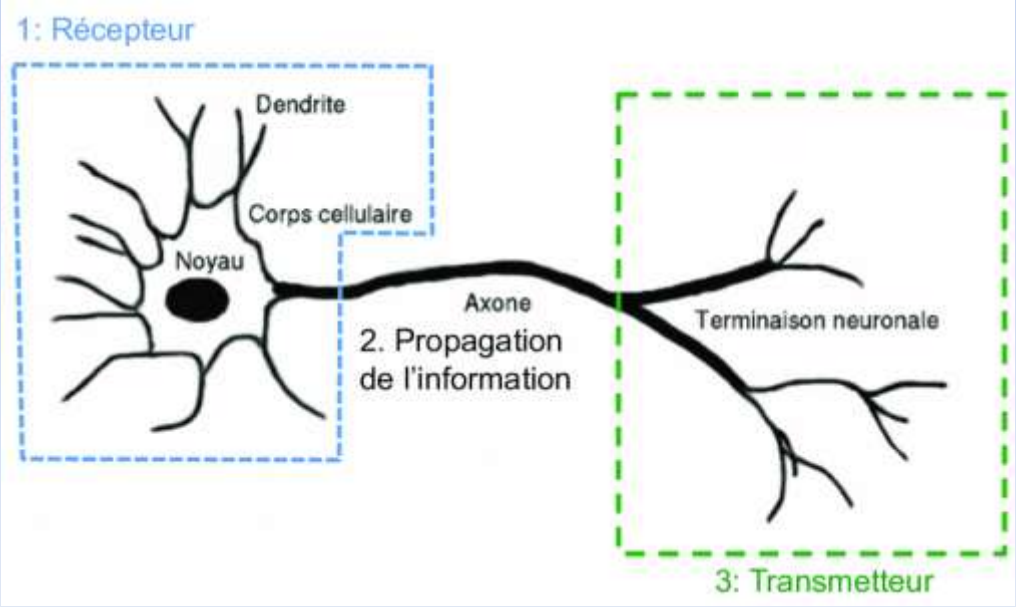
L'**influx nerveux** est assimilable à un signal électrique se propageant comme ceci:

- Le neurone évalue l'ensemble de la simulation reçue.
- Si elle est [illegible] ansmet un signal (



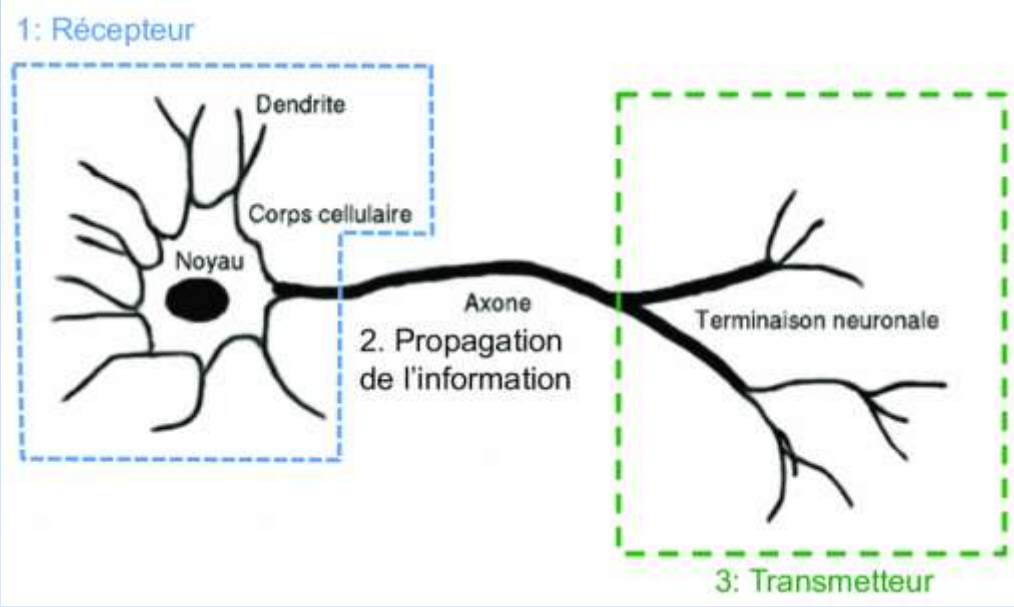
Neurone biologique

- Si elle est suffisante, il est excité: il transmet un signal (0/1) le long de l'axe.



Neurone biologique

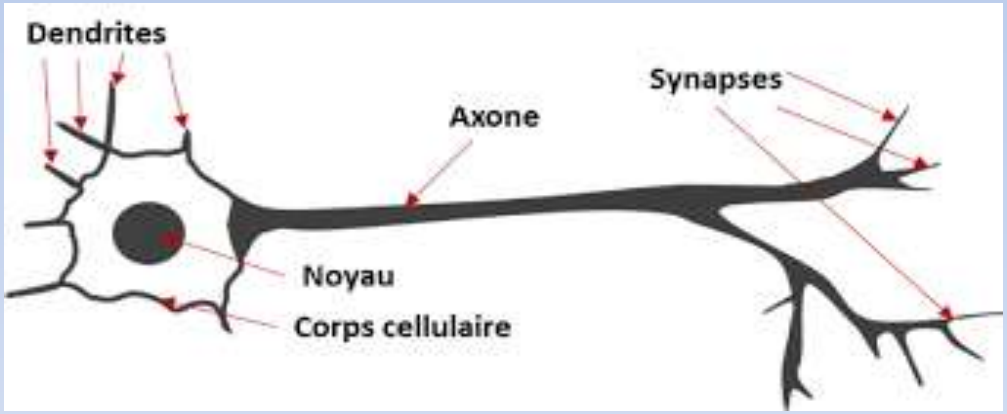
- L'excitation est propagée jusqu'aux autres neurones qui y sont connectés via les **synapse**s.

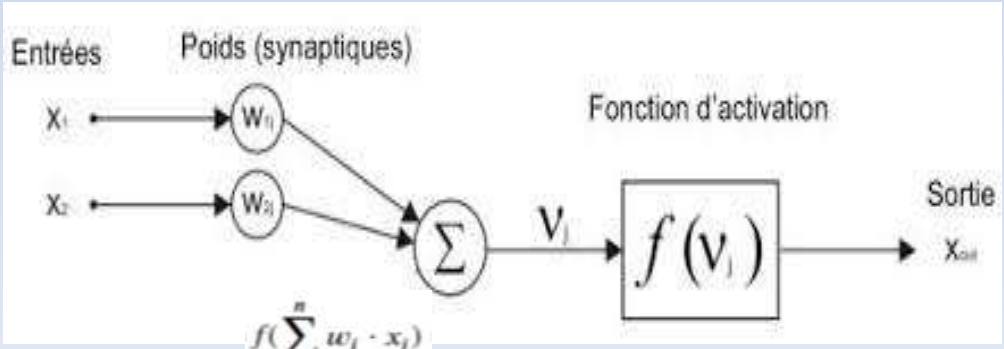


Neurone formel

Neurone formel

- Le neurone formel est une cellule composée d'un corps cellulaire et d'un noyau.





Neurone biologique	Neurone artificiel
Dentrites	Entrées (input)
Synapses	Poids
Axone	Sortie (output)
Activation	Fonction d'activation

La fonction d’activation

La fonction d'activation

Un neurone biologique s'active:

- Si un seuil de tension électrique a été dépassé.

La fonction d'activation du neurone artificiel se déclenche aussi en fonction d'un seuil.

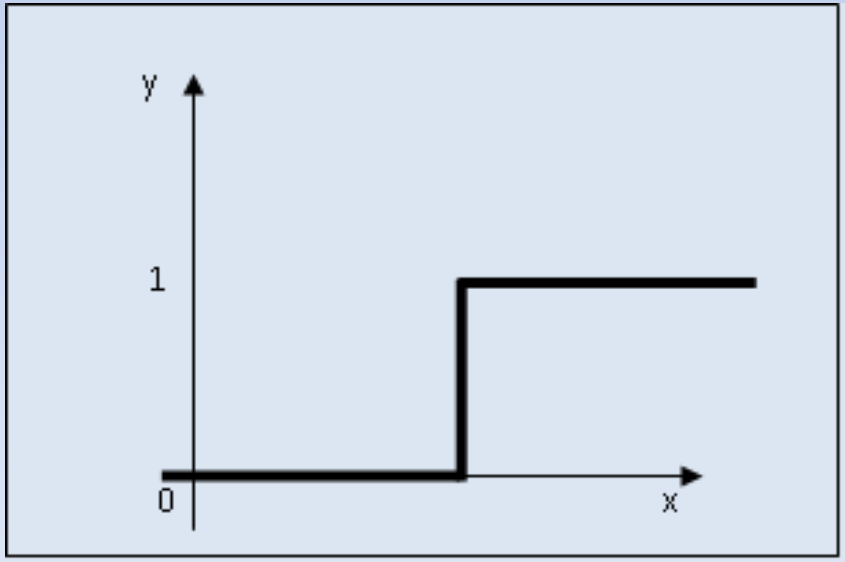
Exemples de fonctions d’activation

Il existe **plusieurs types** de fonctions d’activation

La fonction d'activation

La fonction de seuil binaire

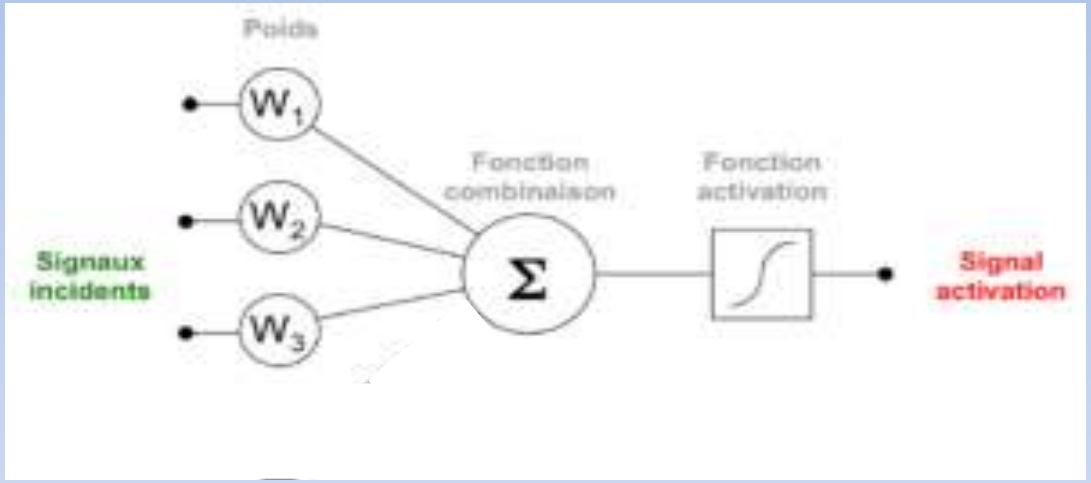
La fonction de seuil binaire retourne une valeur égale à 0 ou 1.



La fonction d'activation

La fonction de seuil binaire

Exemple :



Valeur en entrée	Poids lié à l'entrée	Valeur de l'entrée * valeur des poids
2	0.2	0.4
1	0.1	0.1
3	0.1	22

La fonction d'activation

La fonction de seuil binaire

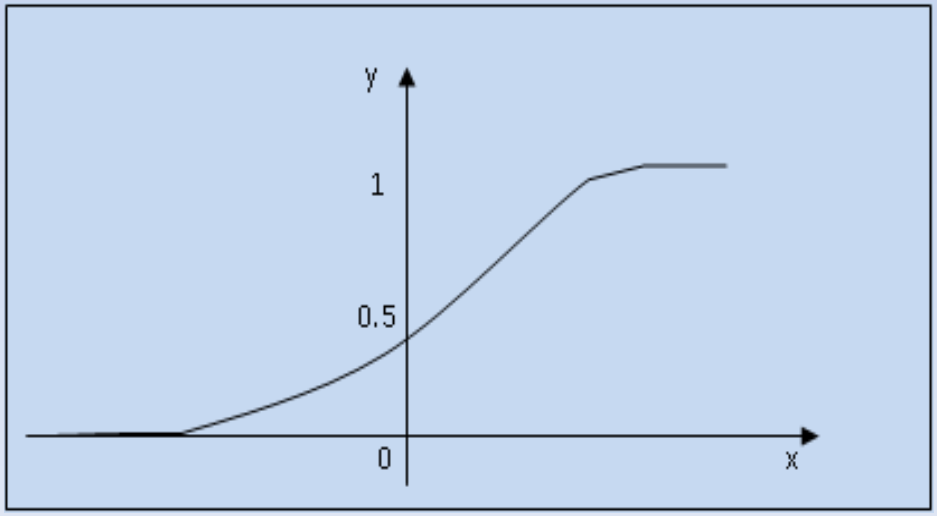
Exemple :

- La valeur du seuil d'activation est égale à 0.5.
- La somme pondérée est donc de **0.8** (0.4+0.1+0.3) provoquant une activation du neurone.

La fonction d'activation

La fonction sigmoïde

- La prédiction est égale à des valeurs numériques comprises entre 0 et 1 (0.50, 0.99, …) exprimant un pourcentage de probabilité.



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

La fonction d'activation

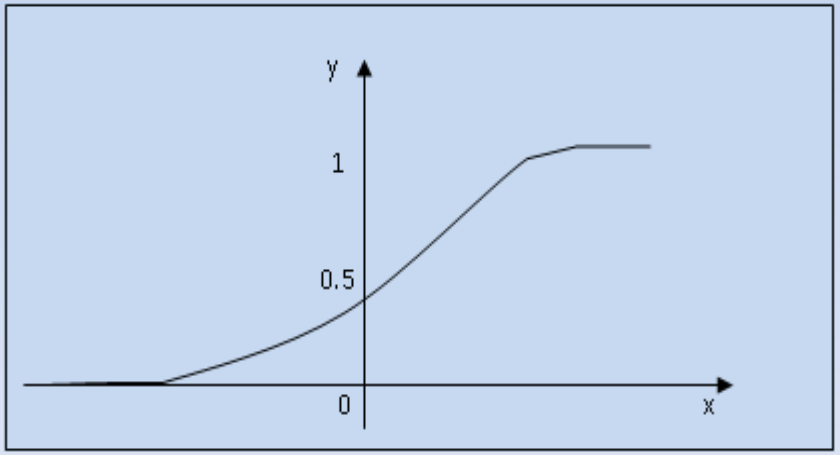
La fonction sigmoïde

Inconvénients:

- Les valeurs négatives peuvent donner des valeurs positives.

Par exemple x=-2, alors le calcul de la fonction sigmoïde de x aura pour valeur 0.45.

- Problème de saturation.



La fonction d'activation

La fonction tangente hyperbolique (tanH)

- La fonction ressemble quelque peu à la fonction sigmoïde.
- **Avantage:** pour toute valeur négative celle-ci reste ou devient fortement négative.

Par exemple: **tan**(-0.2)=-0.19.

$$\tan(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}}$$

- **Inconvénient:**

Problème de saturation.

La fonction d'activation

La fonction de rectification linéaire (ReLU)

Rectified Linear Unit

- **Problème de saturation**:

Les grandes valeurs sont cantonnées à l'intervalle [0,1] et les plus petites à l'intervalle [-1, 0] et une fois la saturation est atteinte, la mise à jour des poids devient difficile.

La fonction d'activation

La fonction de rectification linéaire (ReLU)

- La fonction ReLU pallie le problème de saturation.
- Si la valeur issue de la somme pondérée est inférieur à 0 => celle-ci prend la valeur 0.
- Sinon elle prend la valeur de la somme calculée.

La fonction d'activation

La fonction de rectification linéaire (ReLU)

Avantages :

- Un cout faible en terme de calcul, car il s'agit de prendre le maximum entre la valeur 0 et la valeur de la somme pondérée.
- Elle ne souffre pas de la saturation, car elle n'a pas une valeur limite dans la zone positive.

La fonction d'activation

La fonction de rectification linéaire (ReLU)

Inconvénient :

- Lorsque la fonction fait face à une valeur négative, la valeur 0 est automatiquement attribuée, ce qui entraîne un non-apprentissage du réseau.

La fonction d'activation

La fonction de rectification linéaire (ReLU)

Inconvénient :

- La solution pour pallier à ce problème est apportée par la fonction leakly ReLU qui multiplie la valeur négative par la valeur 0.01, permettant ainsi de déclencher un apprentissage du réseau.

La fonction d'activation

La fonction de rectification linéaire (ReLU)

Remarque :

- Malgré son inconvénient, la fonction ReLU est la plus utilisée dans un réseau de neurone.

La fonction d'activation

La fonction softmax

- Si on veut classifier l'observation parmi plusieurs classes, c'est alors qu'intervient la fonction **Softmax** qui attribue une probabilité à chacune de ces différentes classes tout en veillant à ce que la somme de ces probabilités soit égale à 1.

La fonction d'activation

La fonction softmax

- La fonction Softmax est généralement utilisé dans un réseau de neurone **multicouche** et dans le cas de la classification **multi-classes**.

La fonction d'activation

La fonction softmax

Exemple:

Classe	Probabilité
Animal	0.01
Fruit	0.95
Véhicule	0.04

La rétro propagation de l’erreur

La rétro propagation de l’erreur

- La particularité des réseaux de neurones est: qu’ils apprennent de leurs erreurs.

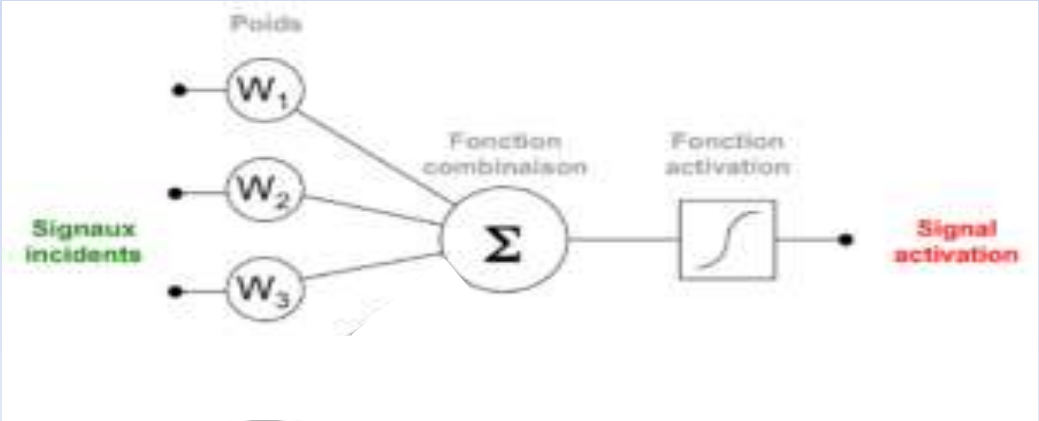
La rétro propagation de l'erreur

- La **propagation** consiste à:

  - Réaliser la somme pondérée des entrées.

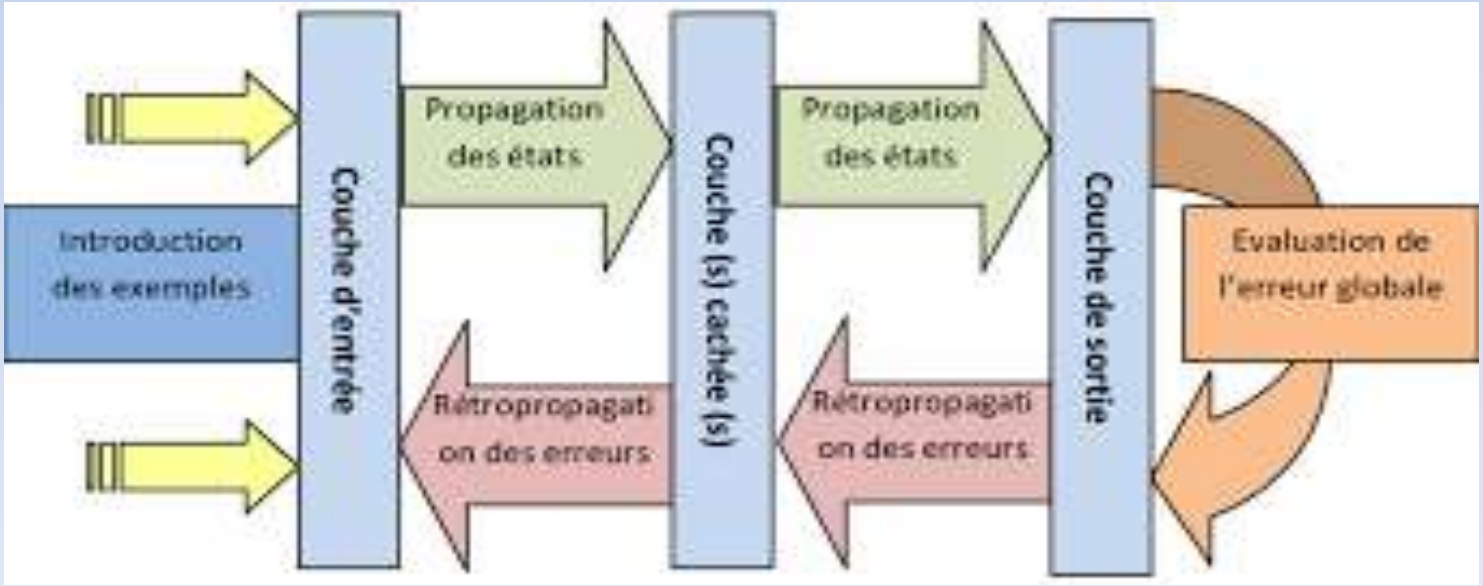
  - A utiliser une fonction d'activation.

Pour obtenir une valeur de prédiction.



La rétro propagation de l'erreur

Nous partons des points d'entrée du neurone artificiel **vers** son point de sortie pour réaliser ces calculs.



La rétro propagation de l'erreur

- Une fois la prédiction réalisée.
- **Erreur de prédiction**=

La prédiction attendue

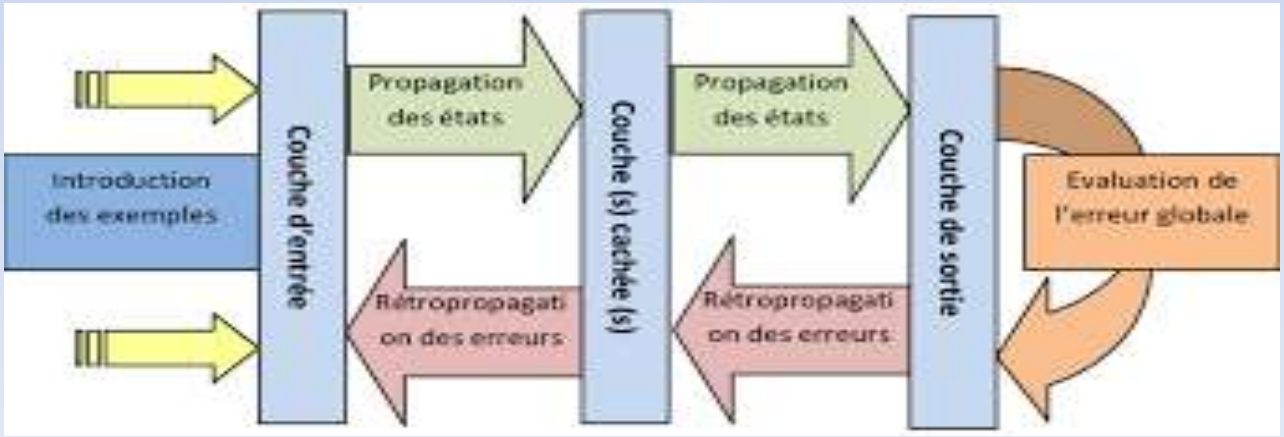
-

la prédiction réalisée

par le neurone artificiel avec

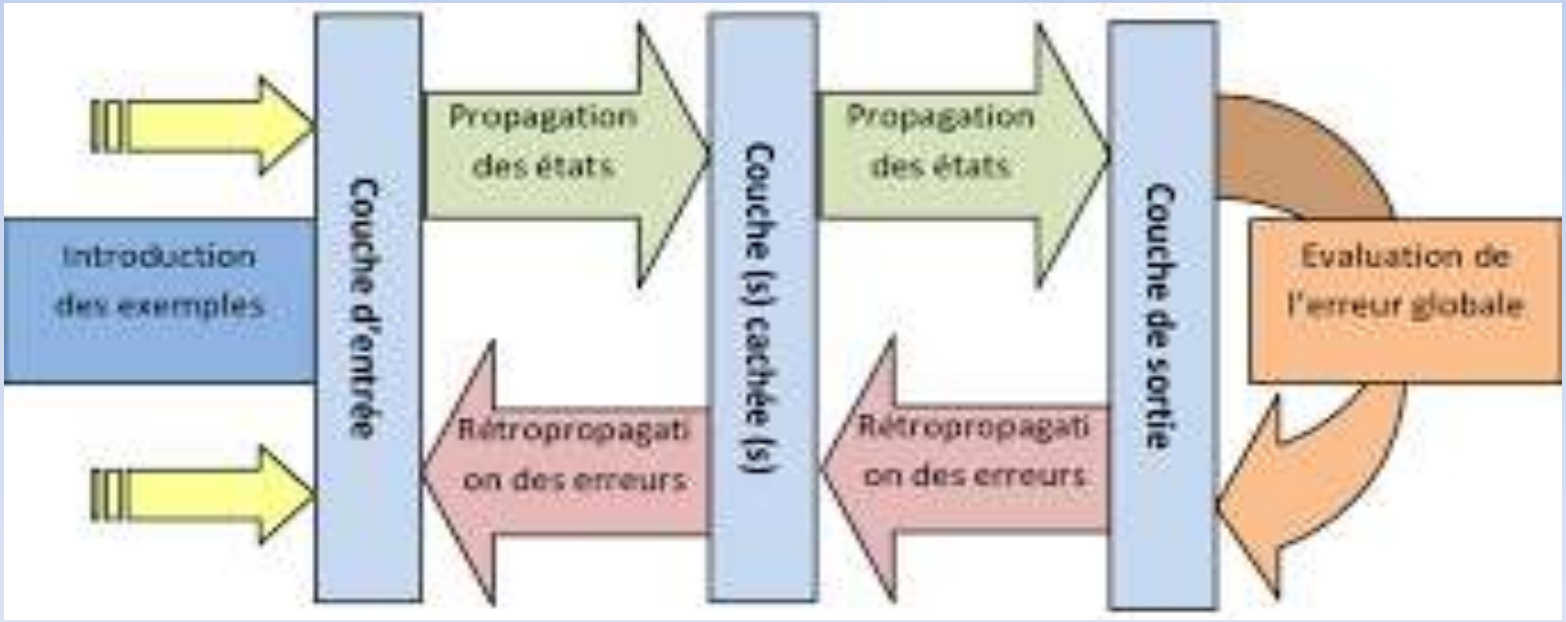
La rétro propagation de l'erreur

- Une fois cette erreur obtenue.
- Nous allons à présent parcourir le neurone en sens inverse (de la sortie vers les entrées) afin de prendre en compte l'erreur commise lors de la prédiction dans l'apprentissage en ajoutant les valeurs des différents poids.



La rétro-propagation de l'erreur

- Cette phase est appelée la rétro-propagation.



Les fonctions de pertes (Loss function)

Les fonctions de pertes

Une fonction de perte est:

- Une fonction qui évalue l'écart entre les **prédictions réalisées** par le réseau de neurone et les **valeurs réelles** des observations utilisées pendant l'apprentissage.

Les fonctions de pertes

- Plus le résultat de la fonction de perte est minimisé, plus le réseau de neurones est performant.
- Sa minimisation c'est-à-dire réduire au minimum l'écart entre la valeur prédite et la valeur réelle pour une observation données, se fait en ajustant les différents poids du réseau de neurones.

Les fonctions de pertes

L’erreur linéaire ou erreur locale :

- L’erreur d’apprentissage, appelée aussi erreur locale, se calcul en réalisant la différence entre la valeur réelle à prédire et la valeur prédite par le neurone artificiel.

Erreur= Prédiction_réelle – Prédiction_realisee

Les fonctions de pertes

L’erreur linéaire ou erreur locale :

- C’est cette erreur que nous cherchons à minimiser au fur et à mesure des apprentissages.
- Cette erreur peut être qualifiée d’erreur locale, car elle se focalise sur une observation donnée en comparant la valeur réelle et sa valeur prédite.

Les fonctions de pertes

Erreur moyenne quadratique MSE ou erreur globale

$$MSE = \frac{1}{n}\sum_{i=1}^{n} E^2$$

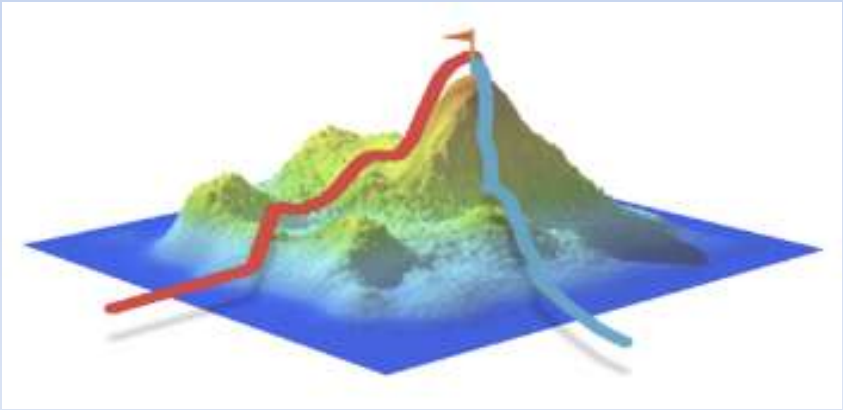
La descente du gradient

L'objectif de la descente du gradient est de:

- **Minimiser** la fonction d'erreur en ajustant **petit à petit** les paramètres d'apprentissage représentés par les différents poids.

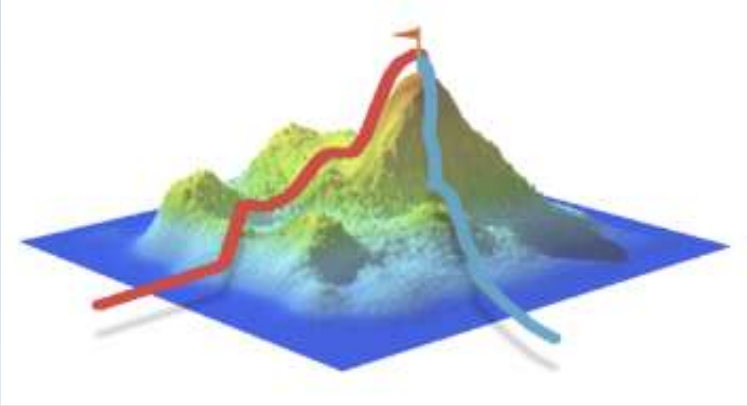
La descente du gradient

- Vous vous situez au point le plus de la montagne et vous souhaitez atteindre la plaine en contrebas.



La descente du gradient

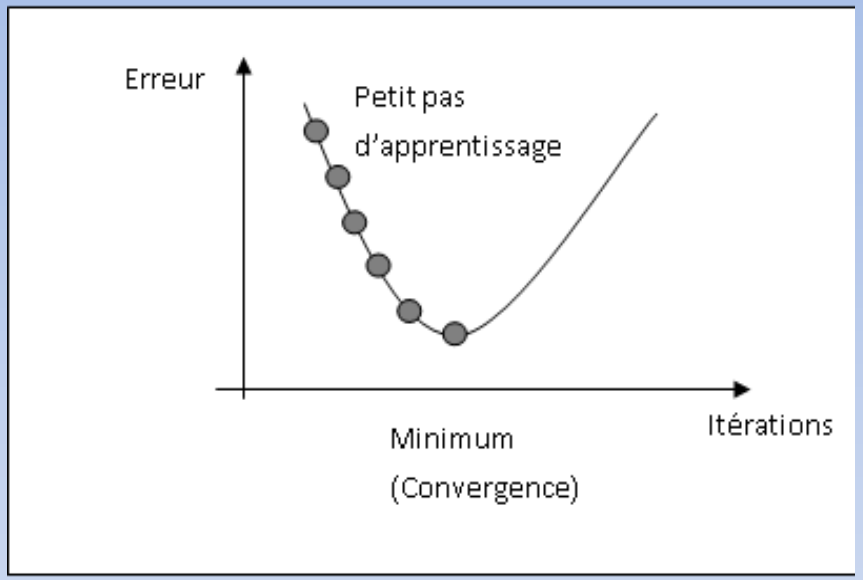
- Cependant, il fait nuit noire et vous n'êtes pas en mesure de voir où vous allez.
- Vous allez donc progresser doucement par petits pas jusqu'à atteindre le bas de la vallée.



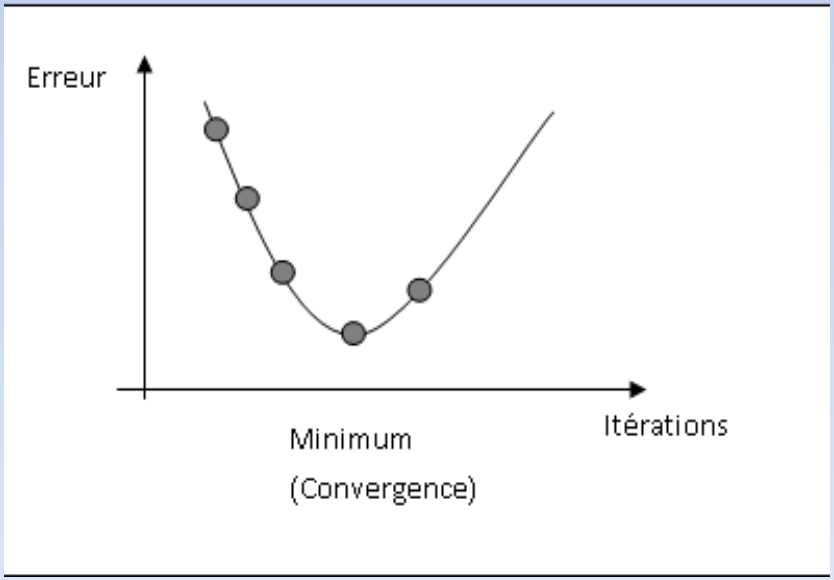
La descente du gradient

- La descente du gradient correspond à cette métaphore et se réalise par l'ajustement des différents poids du réseau de neurones jusqu'à obtenir une convergence, c'est-à-dire un minimum d'erreurs.
- Cette ajustement se fait par petit pas à l'aide d'un hyper paramètre appelé taux d'apprentissage (Learning rate).

La descente du gradient

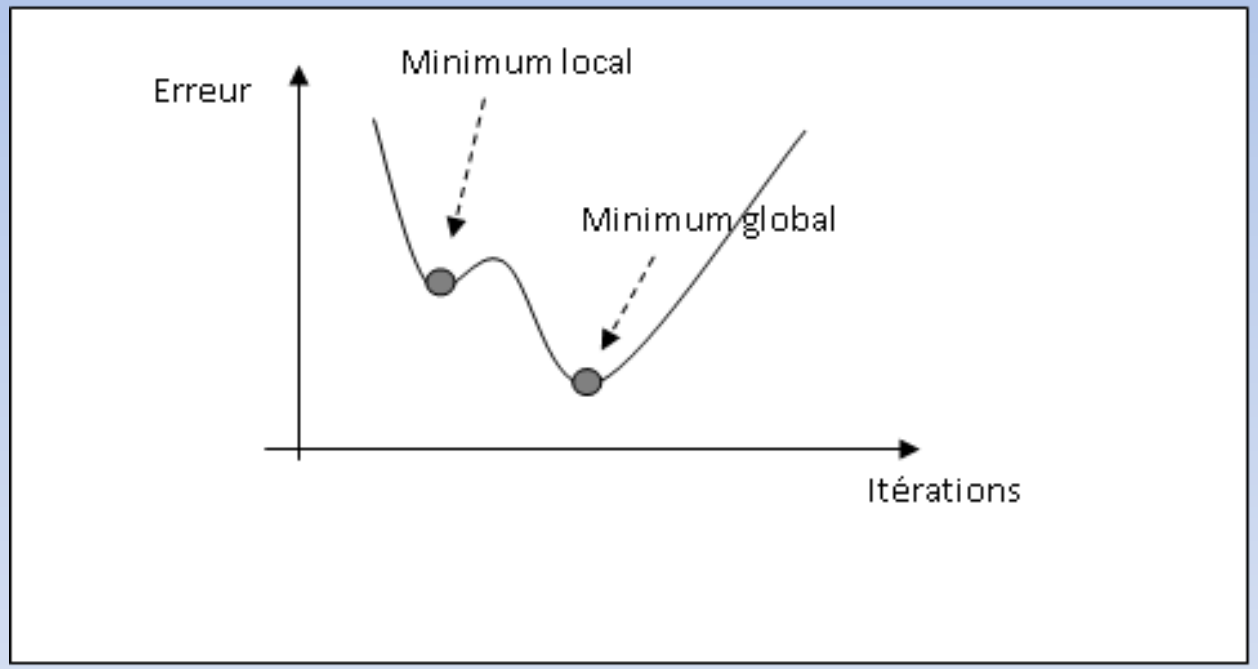


Descente du gradient



Descente du gradient Minimum dépassé

La descente du gradient



Descente du gradient : Minimum local et Minimum global

La descente du gradient

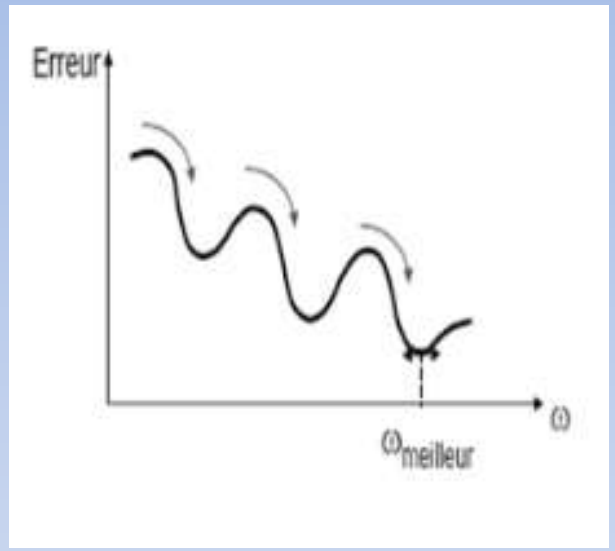
Remarque :

- Un **paramètre** est défini au niveau du modèle d'apprentissage (valeurs des poids).
- Un **hyper paramètre** est défini au niveau de l'algorithme (taux d'apprentissage, nombre de couches de neurones par exemple).

La descente du gradient

Gradient = dérivée partielle

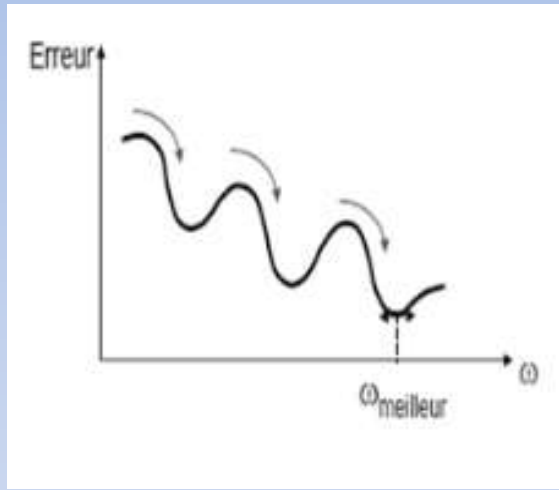
$$df(x) = \partial f / \partial x$$



Elle permet de descendre pas à pas vers la valeur minimale.

La descente du gradient

Ajuster les paramètres dans la **direction** du gradient pour descendre pas à pas vers les valeurs optimales qui minimise la fonction de perte



$$w_i = w_{i-1} + \Delta w$$

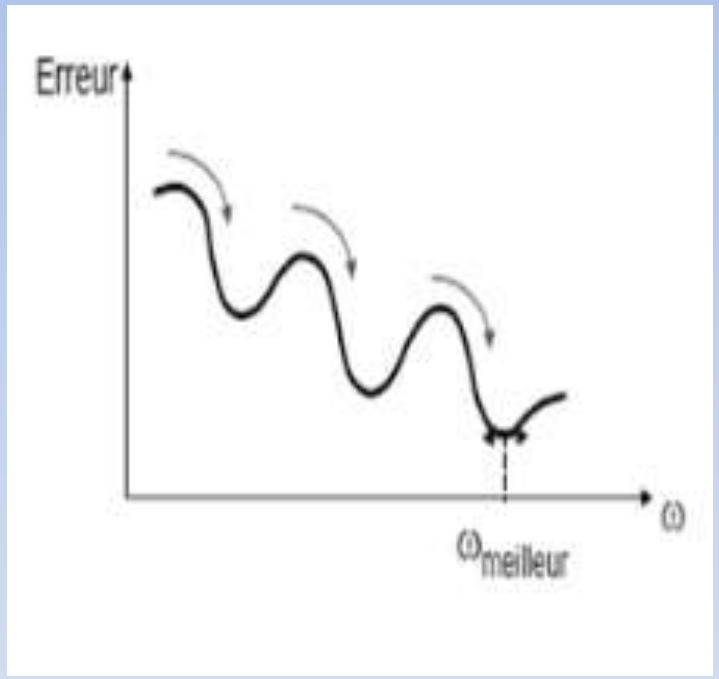
La descente du gradient

$$w_i = w_{i-1} + \Delta w$$

$$\Delta w = -t_x \nabla g(w)$$

$$\nabla g(w) = \frac{\partial g}{\partial w_i}$$

Comment la calculer?



C'est la règle d'apprentissage **Delta** (Delta Learning Rule)

La descente du gradient

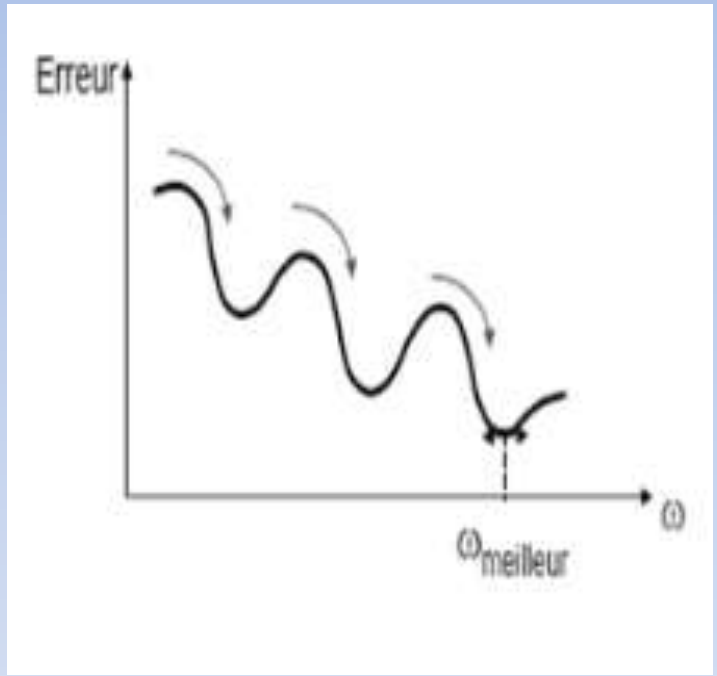
$$w_i = w_{i-1} + \Delta w$$

$$\Delta w = -t_x \nabla g(w)$$

$$\nabla g(w) = \frac{\partial g}{\partial w_i}$$

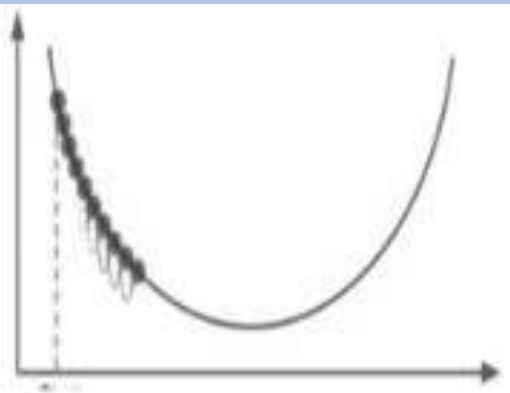
Dans le cas d'une sigmoïde $\frac{1}{1+e^{-z}}$

$$\frac{\partial g}{\partial w_i} = \frac{\partial z}{\partial w_i}\frac{dg}{dz} = x_i * g(1-g)$$

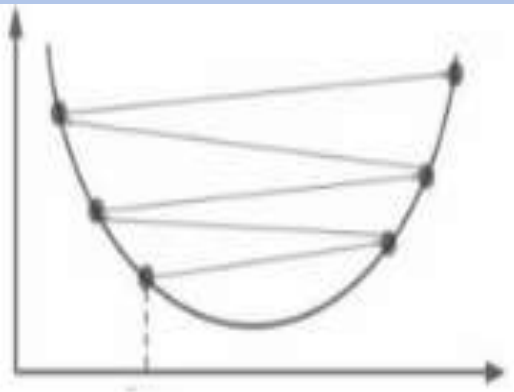


La descente du gradient

Le pas d'apprentissage permet de faire un saut vers le prochain point où on doit calculer la dérivée.



Pas d'apprentissage faible ➔ temps de convergence élevé



Pas d'apprentissage élevé ➔ Risque de non convergence

Biais

Biais

Les biais sont:

- D**es paramètres additionnels utilisés dans les réseaux de neurones.**

- Pour ajuster les valeurs d'entrées auxquelles les poids ont été appliqués, avant l'obtention de valeurs de sorties définitives.

- Elles influencent le comportement de la fonction d'activation.

Biais

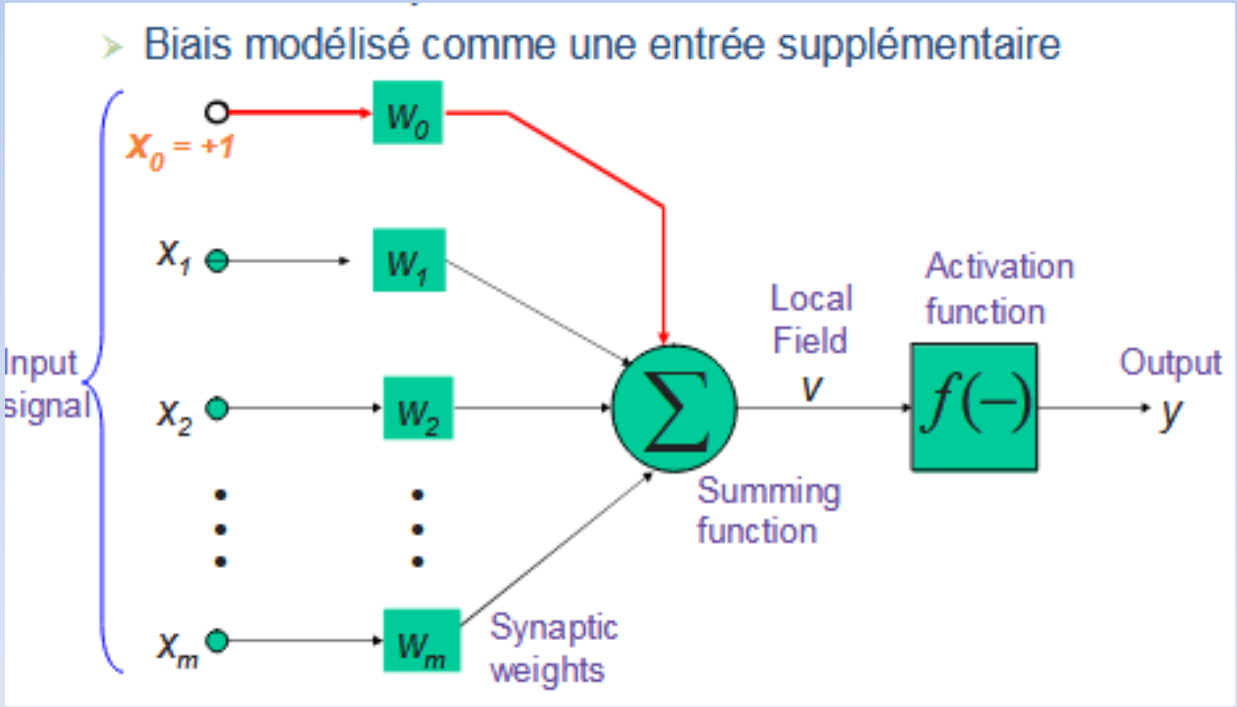
Si on veut **forcer** la valeur de la **prédiction** pour certaines valeurs d'entrée cela est possible à l'aide du « biais ».

Biais modélisé comme une entrée supplémentaire

Biais modélisé comme paramètre d'une fonction affine

Biais

Biais modélisé comme une **entrée supplémentaire**



Biais

Exemple :

Soit un neurone composé de:

1. Deux entrées ayant pour valeur 1 et 2.
2. Deux poids ayant pour valeurs -0.45 et 0.1.
3. Une sortie.

La valeur de cette sortie est: -0,25

ReLU(1*(-0.45) + 2*0.1)=0.

Biais

Exemple :

Ajoutons un neurone qui prendra pour valeur 1 avec un poids de 0.6.

Ce qui nous donne le calcul suivant :

ReLU (1*(-0.45) +2*0.1+**1*0.6**)=0.35

On constate donc que l'ajout d'un biais permet de donner une meilleure flexibilité au réseau de neurones dans leur apprentissage.

Biais

Remarque :

- En ce qui concerne la valeur du poids du biais à son initialisation, il est commun d'utiliser la valeur 0.
- La valeur du biais est égale à 1.

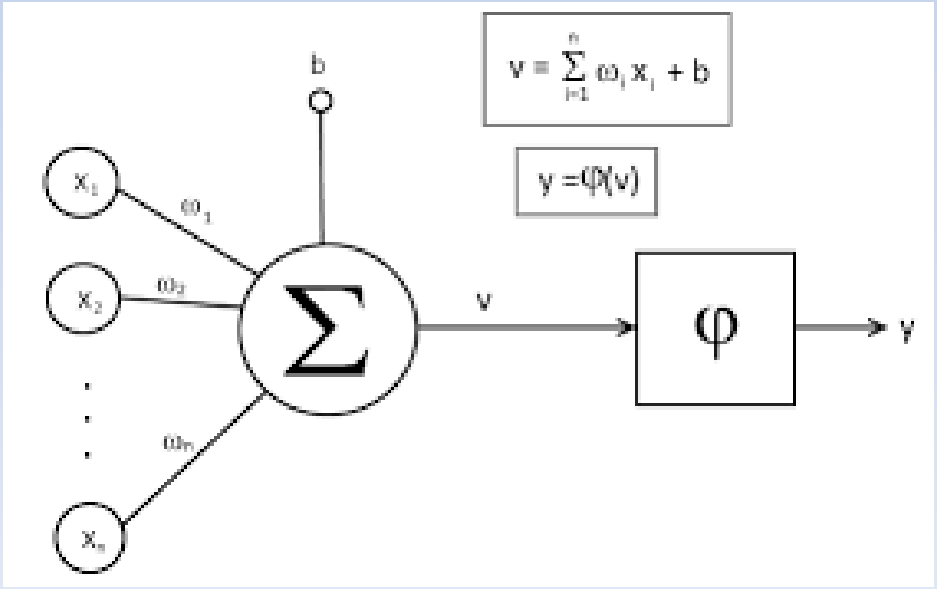
Biais

Biais b :

Pour appliquer une transformation affine à u.

$$u = \sum_{j=1}^{m} w_j x_j$$

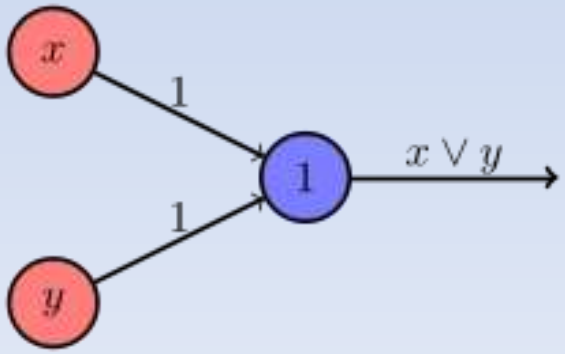
v = u + b



Perceptron

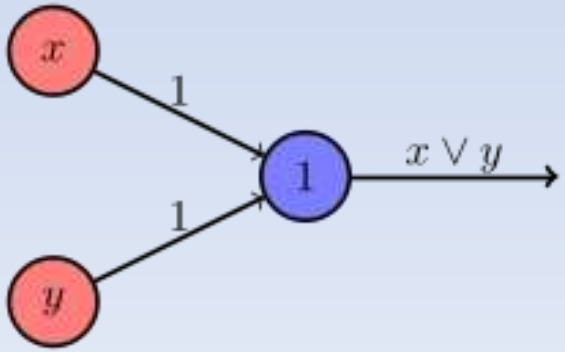
Perceptron

- Le **perceptron** est un neurone formel binaire. Cela signifie que son unique sortie est soit 0, soit 1.
- Il permet de faire une classification binaire.



Perceptron

- Les données traitées sont linéairement séparable.
- Le perceptron est monocouche et n'a qu'une seule sortie (booléenne) à laquelle toutes les entrées sont connectées.

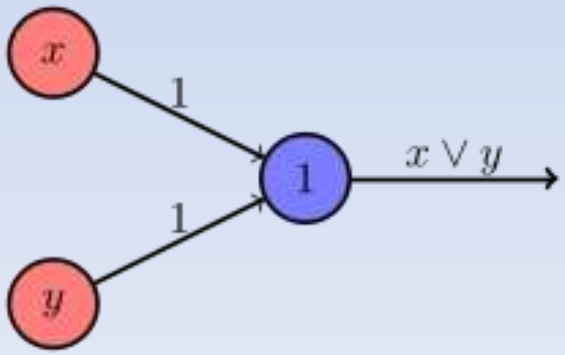


Perceptron

Exemple:

- Un perceptron avec 2 entrées x et y .
- Les poids sont marqués sur les arcs : 1 et 1.
- Ce perceptron calcule le **OU** logique de x et y.

X	Y	X ou Y
0	0	0
0	1	1
1	0	1
1	1	1



Perceptron

- Le perceptron simple permet de **classifier les données linéairement** séparables.
- Les données linéairement séparable sont celles pouvant être séparées par une droite.
- Le perceptron multicouche permet de classifier les données non linéairement séparables.

Exemple

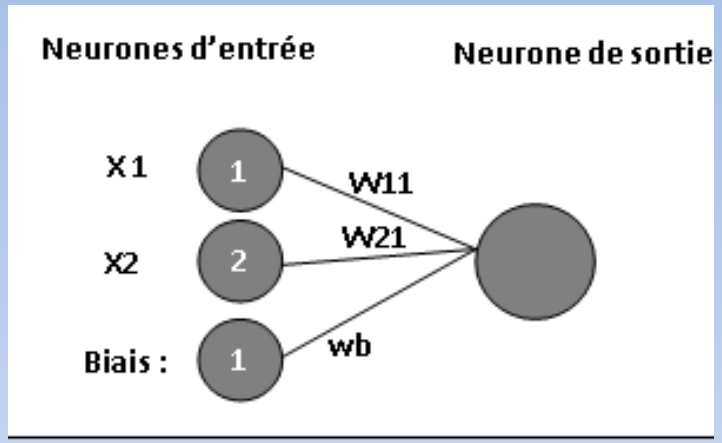
Travail demandé :

Construire un modèle capable de déterminer si un étudiant, selon des critères précis, peut être reçu ou non dans l'université X. Le tableau présenté ci-dessous regroupe différents cas d'admissions et de refus.

Réussite à l'examen de mathématiques	Réussite à l'examen d'informatique	Admis
OUI	NON	NON
OUI	OUI	OUI
NON	OUI	NON
NON	NON	NON

Exemple

Initialisation du perceptron :



Le perceptron est composé :

1. D'une **couche d'entrée** constituée de deux neurones correspondant chacun à la réussite aux examens.
2. Un **neurone de sortie** qui permettra de classifier l'étudiant en tant **qu'admis ou refusé** dans l'université.

Exemple

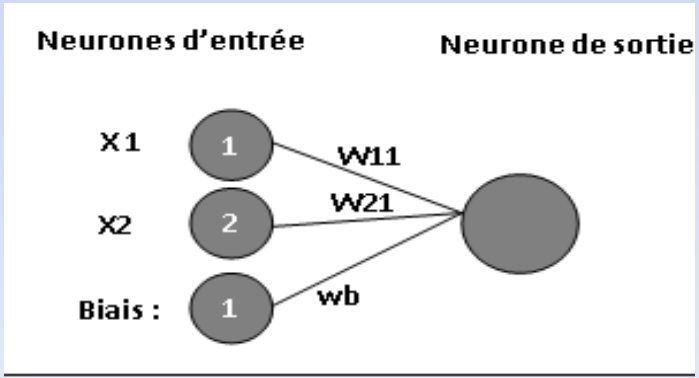
Initialisation du perceptron :

•Un neurone de biais qui a pour but de contrôler la prédisposition du neurone à s'activer ou non et qui prendra la valeur 1.

• **W11**: le poids du premier neurone de la première couche.

• **W21**: le poids du deuxième neurone de la première couche.

• **Wb** correspond au poids du biais.



Exemple

Les étapes d'apprentissage :

Etape 1 : initialisation des poids

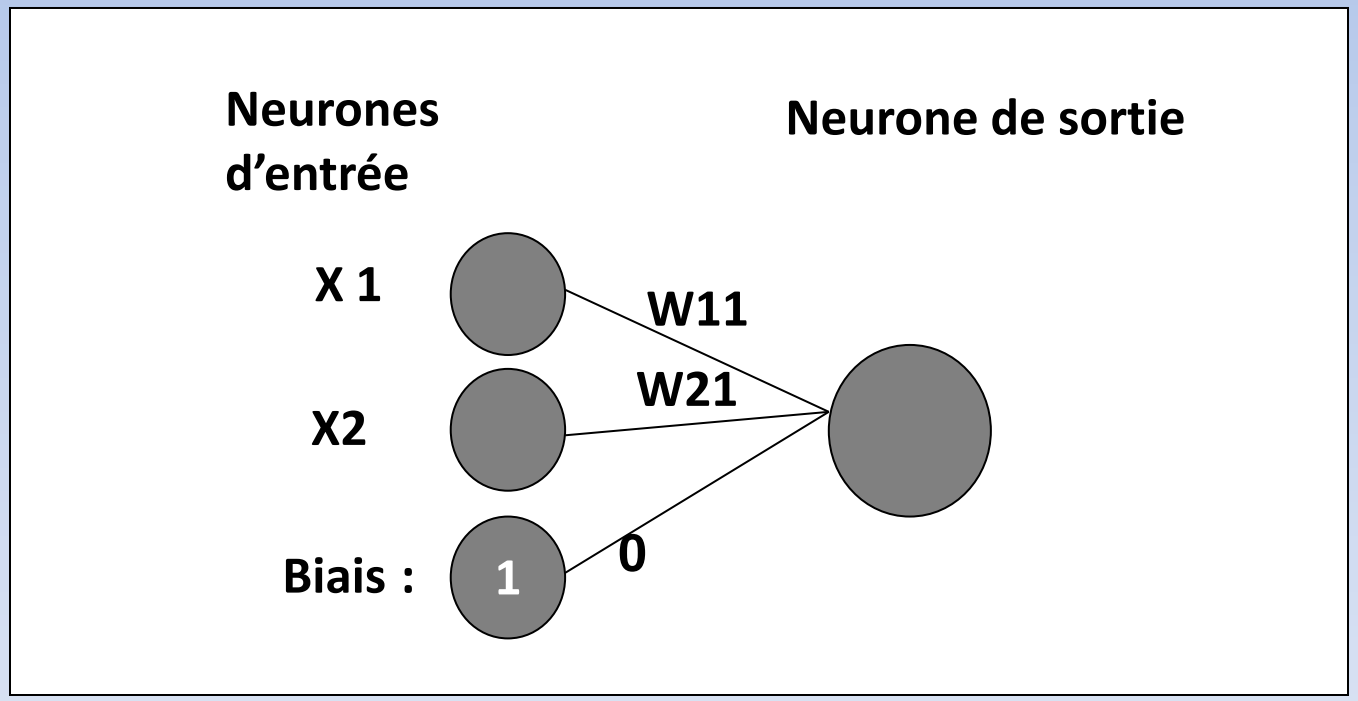
L'initialisation des poids dans cet exercice se fait d'une façon aléatoire dans un intervalle entre -1 et 1 sauf pour le biais qui prendra la valeur 0.

W11= -0.165955990594852

W21= 0.4406489868843162

Wb=0

Exemple



Initialisation des poids

Exemple

Etape 2: chargement des données de la première observation

Il s'agit d'alimenter les valeurs de X1 et X2 par 0 et 1 au lieu de de OUI et NON.

Réussite à l'examen de mathématiques	Réussite à l'examen d'informatique	Admis
1	0	0
1	1	1
0	1	0
0	0	0

Exemple

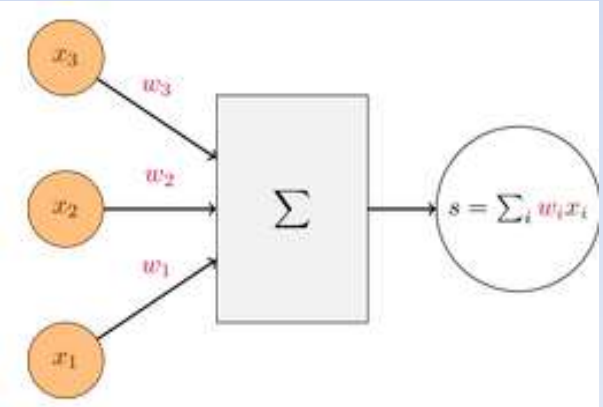
Etape 3 : Pré-activation :

La somme pondérée des différents poids est calculée comme suit :

SommePondérée = valeur du biais*Wb + (W11*X1+W21*X2)

SommePondérée = 1*0 + (-0.165955990594852*1 + 0.4406489868843162 *0)

SommePondérée = -0.165955990594852



Exemple

Etape 4 : utilisation d'une fonction d'activation

La fonction d'activation a pour rôle de réaliser la prédiction (nommée y). La fonction d'activation utilisée dans cet exemple est la fonction **sigmoïde**.

Y=1/ (1+ ***exp***(-SommePondérée)

Donc, Y= 1/(1+ ***exp*** (-(0.165955990594852)))

Y= 0.45860596

Exemple

Règle d'apprentissage du perceptron:

- Initialiser le perceptron avec des poids aléatoire.
- Traiter tous les exemples d'entrainement jusqu'à ce

que le perceptron les classifie tous correctement.

Exemple

Règle d'apprentissage du perceptron:

• Pour chaque exemple, les poids sont révisés à l'aide de la règle d'apprentissage **Delta**:

$$w_i = w_{i-1} + \Delta w$$

$$w_i = w_{i-1} + t_x * Err * g'(in) * x_i$$

$$Err = y - g(in)$$

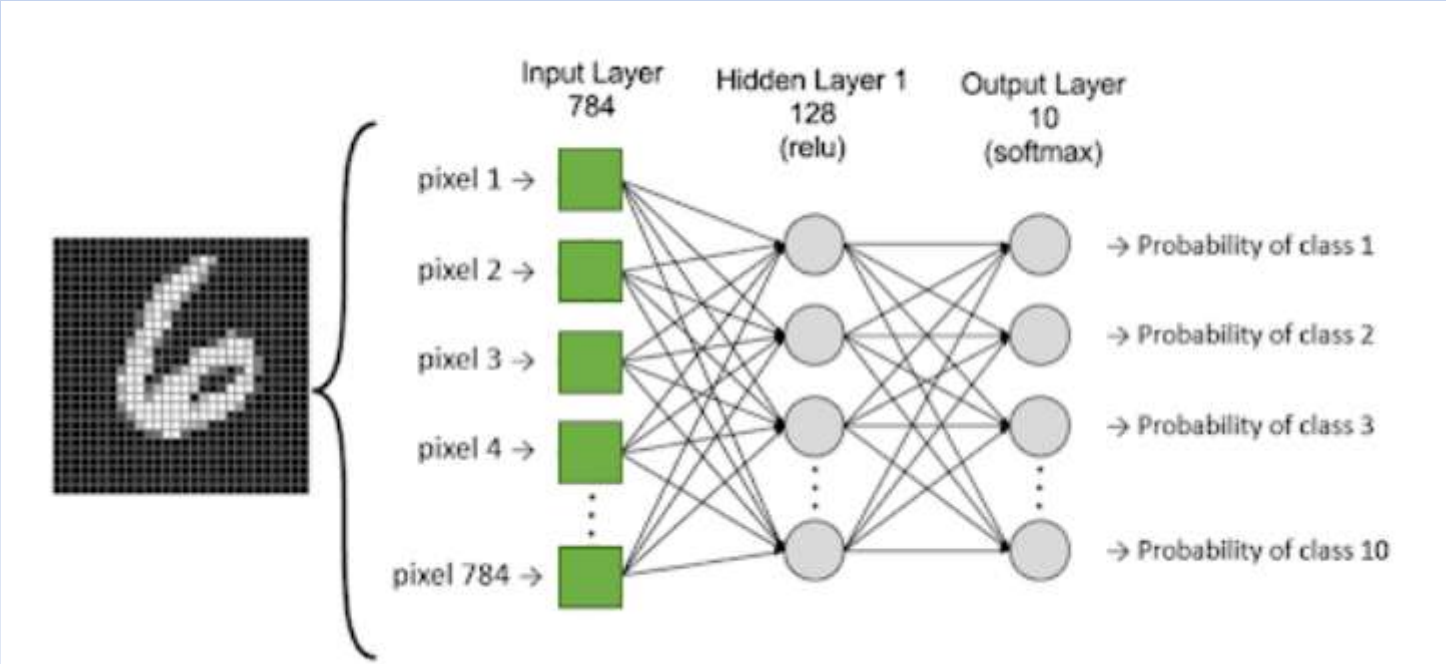
y: la sortie désirée / ***g***: la sortie obtenue / t_x le taux d'apprentissage

Pour une fonction **sigmoïde**: ***g'=g(1-g)***

Perceptron multicouches

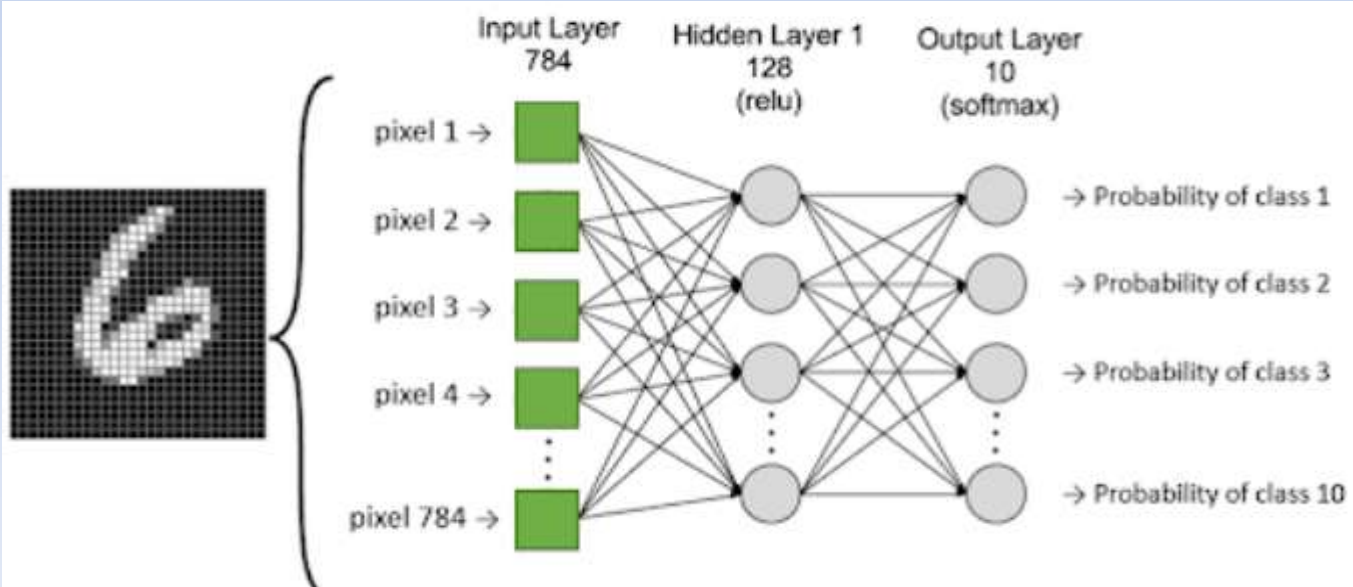
Perceptron multicouches

- Les perceptrons multicouches sont un empilement de perceptron qui sont mis sous forme de couche.



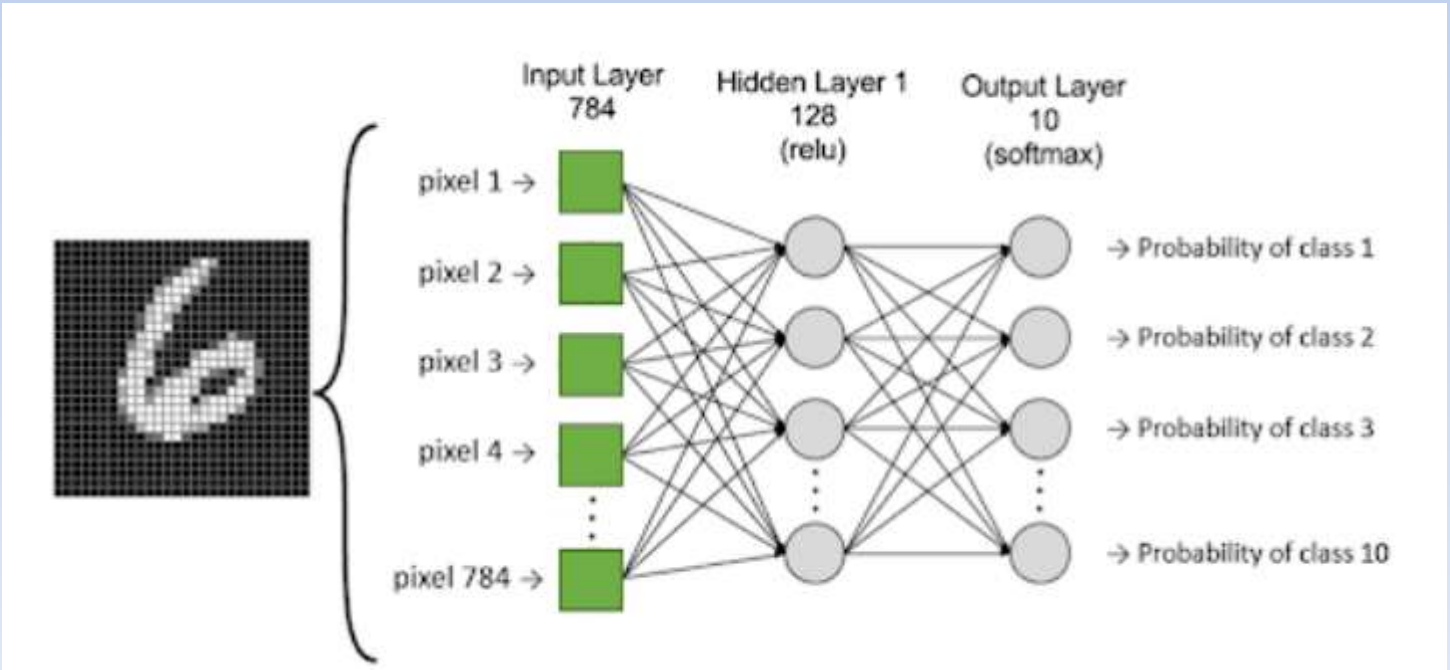
Perceptron multicouches

- Le modèle consiste à classifier (en 10 classes) des images de chiffres manuscrits.
- Les carrés verts sont la couche d'entrée.



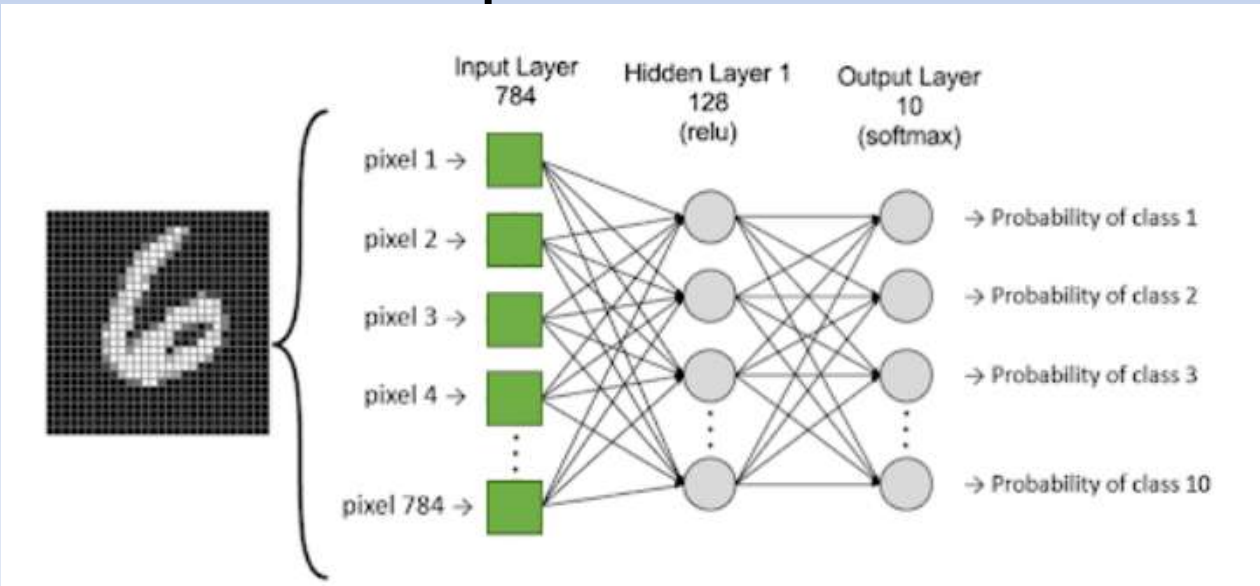
Perceptron multicouches

- La couche intermédiaire et la couche sortie sont des perceptrons représentés par des ronds gris.



Perceptron multicouches

- La dernière couche possède 10 neurones puisqu'il y a 10 classes et retourne la probabilité correspondante à chaque classe.



Architectures des réseaux de neurones

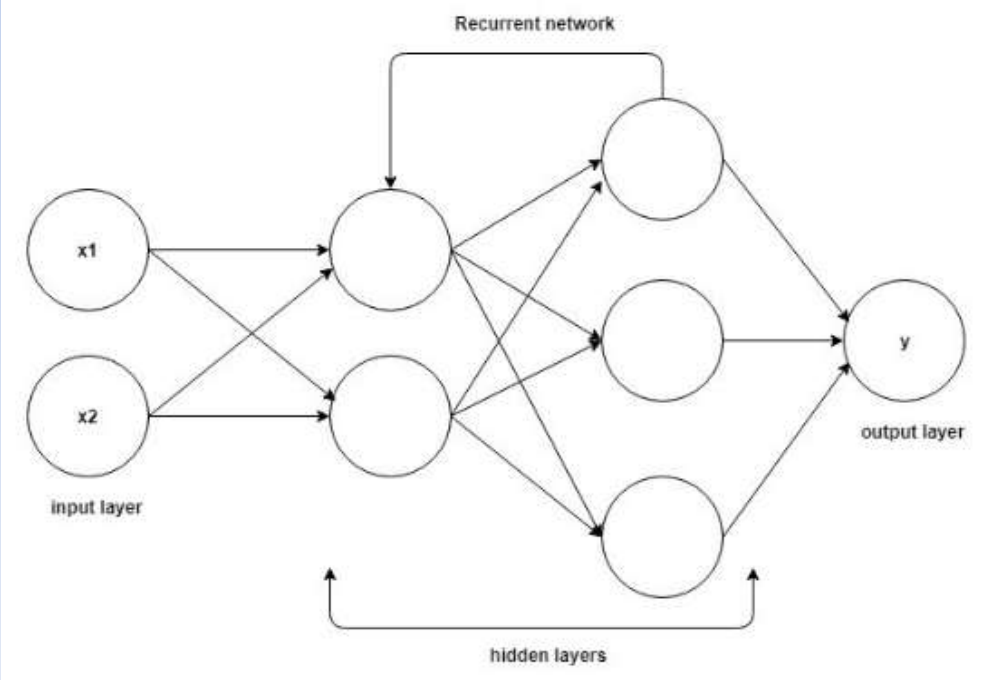
Réseau de neurones récurrents

Recurrent neural network

Réseau de neurones récurrents

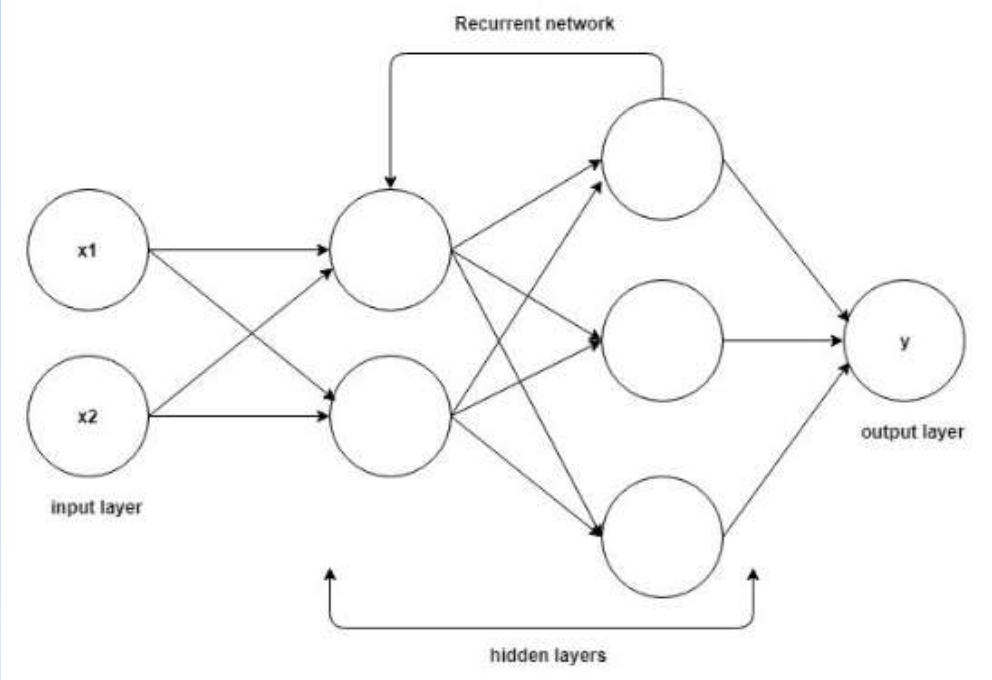
- Le réseau de neurones récurrent est:

Un type de réseau feedback ou bouclé.



Réseau de neurones récurrents

- Les **RNN** utilisent les sorties précédentes comme entrées supplémentaires



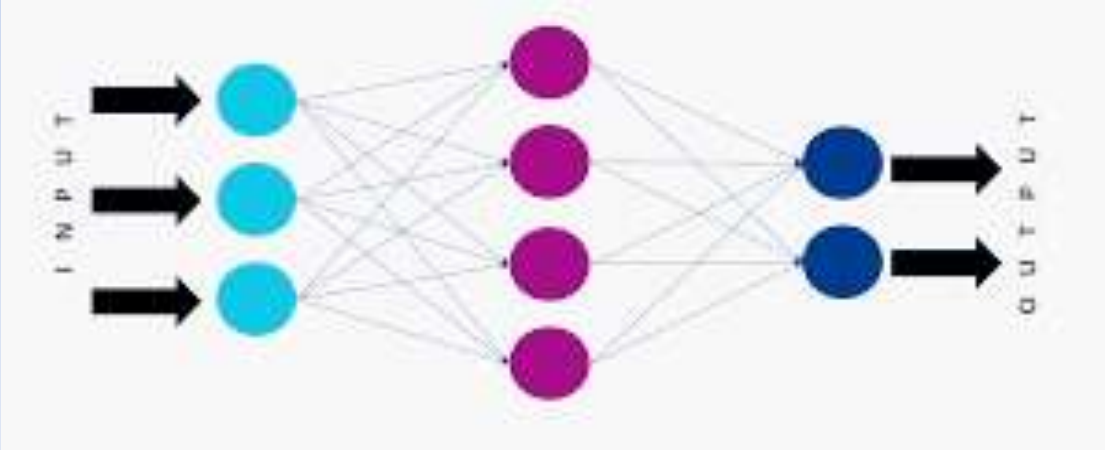
Architectures des réseaux de neurones

Réseau de neurones à propagation avant

Feed forward neural network

Réseau de neurones à propagation avant

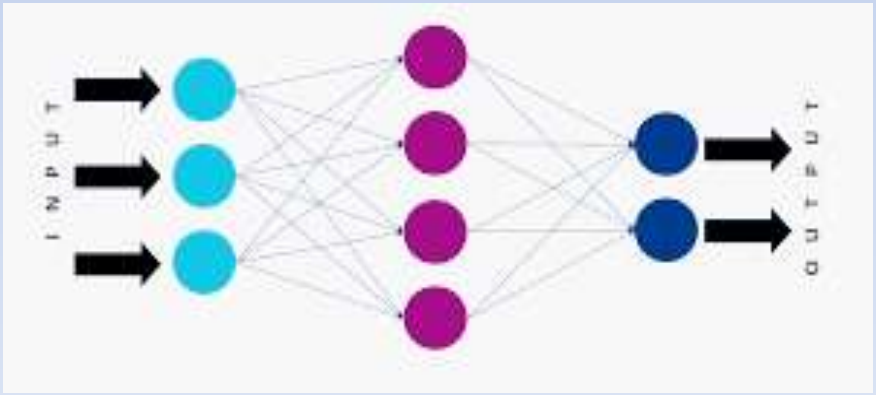
- **C**'est un **RN** acyclique, il n'y a pas de cycles ou de boucles dans le réseau. Exemple: Perceptron multi-couches.

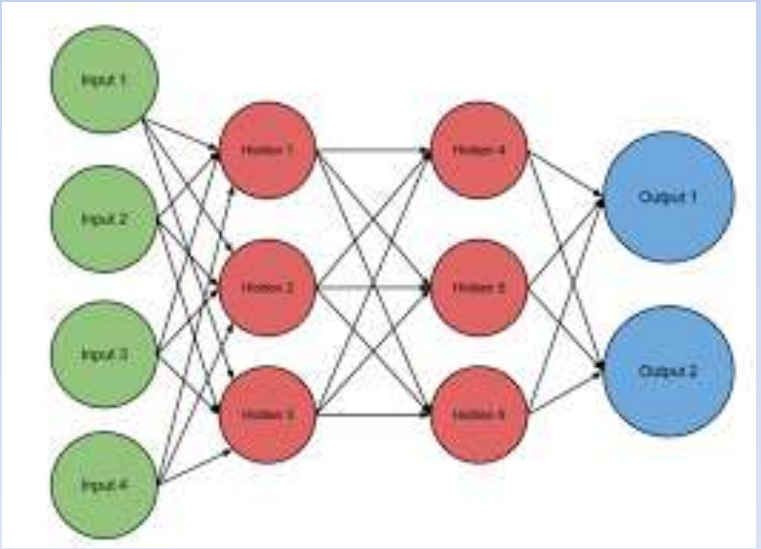


Réseau de neurones à propagation avant

- L'information ne se déplace que dans:

Une seule direction => Vers l'avant.





Merci pour votre attention

Concepts importants

Fonction objective

- Concrètement, l'apprentissage consiste à mettre à jour un ensemble de paramètres (poids + bias) de sorte à minimiser l'erreur entre la sortie du modèle et la sortie désirée (cible).

Fonction objective

Une fonction dite:

fonction de perte / fonction coût /fonction erreur /

fonction objective, est utilisée pour **mesurer la performance** d'un modèle (bon ou mauvais).