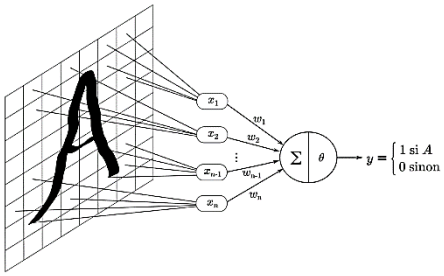
Introduction au DEEP LEARNING

Le Deep Learning est un domaine de Machine Learning, elle-même domaine de l'IA Intelligence Artificielle.

La machine learning est capable développer un modèle en se servant d'un algorithme d'optimisation pour minimiser les erreurs entre le modèle et les données.

Le deep learning est une machine learning composé de réseau de neurones artificiels (correspondant à un réseau de fonctions mathématiques connectées entre elles).

Grâce à un stade d'apprentissage profond, le deep learning réalise son propre « algorithme d'optimisation ».



Le perceptron de Rosenblatt

Le premier neurone capable d'apprendre de ***manière supervisé*** fut proposé par Rosenblatt en 1957, pour modéliser la reconnaissance visuelle de caractère. Il est basé sur un neurone formel inventé en 1943 par McCulloch et Pitts.

Le deep learning est un PERCEPTRON multi couches MLP (Multi Layers Perceptron) fonctionnant avec 5 étapes :

1. Propagation_*Forward propagation* : phase d'apprentissage à partir d'une base de données connue
2. Fonction coût_*Cost Fonction* : évaluation de l'erreur entre la sortie du deep learning et la valeur true
3. Rétropropagation_*Backward propogation* : mesure de l'influence de chacune des couches de neurone en remontant de la dernière à la première couche
4. Gradients_*Gradient descent* : correction de chaque paramètre du modèle
5. Retour à l'étape 1

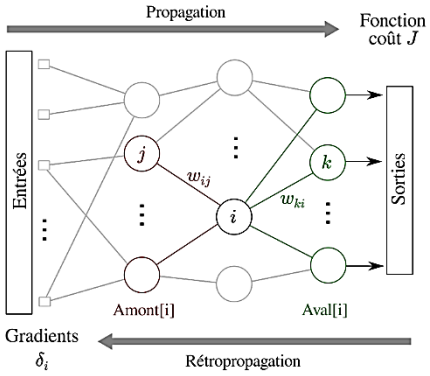
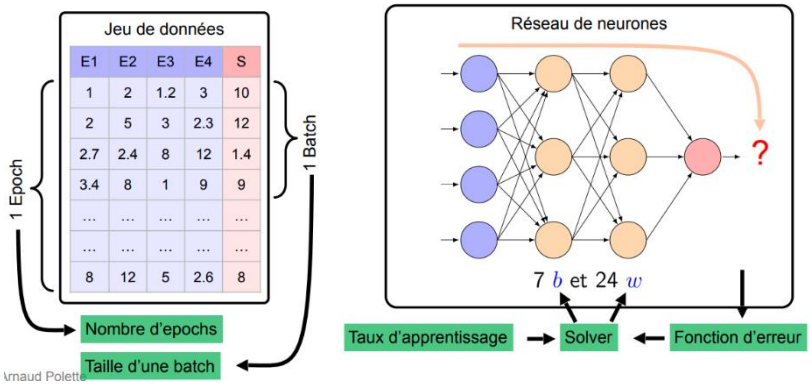


FIGURE 17 – Schématisation de l'apprentissage d'un MLP

Lorsque le deep learning réalise une fois toutes ces étapes on dit qu'il réalise un **cycle d'entrainement :** une itération.

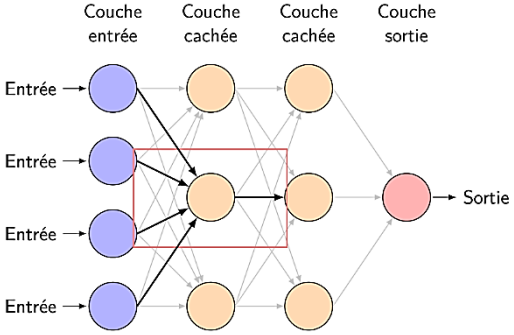
Lorsque le volume de données est massif, le jeu de données, nommé EPOCH est décomposé en BATCH.

Le nombre d'epoch et la taille des batch doit être choisi correctement pour que le deep learning soit efficace.



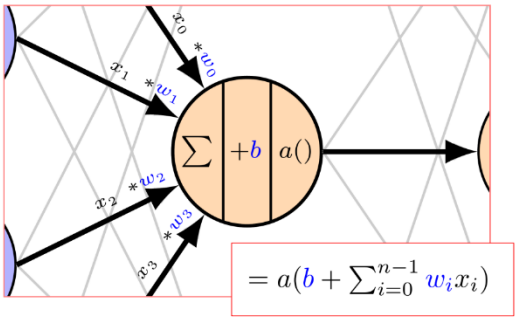
Structure d'un réseau de neurones :

Le deep learning est composé de plusieurs couches (entrée, cachées, sortie) et chaque couche comporte un ou plusieurs neurones (fonction mathématique).



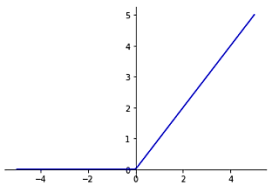
Chaque neurone est relié aux autres neurones par des synapses : ***variables*** x_i affectées à ***un poids (wight)*** w_i **.**

Le neurone va traiter ces informations par l'intermédiaire d'une fonction d'activation $a(b + \sum_{i=0}^{n-1} w_i x_i)$ où b est le ***biais*** et $\sum_{i=0}^{n-1} w_i x_i$ est le ***potentiel synaptique***



Les fonctions d'activation les plus usuelles sont les suivantes :

fonction_ReLU(x) = max(x, 0)



Fonction ReLU – Rectified Linear Unit

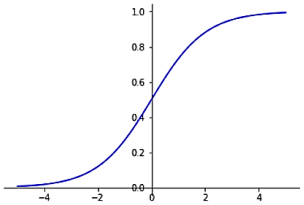
- La fonction **ReLU** : Rectified Linear Unit : elle laisse passer les valeurs positives (x>0) dans les couches suivantes du réseau de neurones

  Elle est utilisée presque partout mais surtout pas dans la couche finale, elle est utilisée dans les couches intermédiaires

fonction_Sigmoid(x) = 1 / (1 + exp(-x))

- La fonction **Sigmoid** : elle donne une valeur entre 0 et 1, une probabilité.

  Elle est donc très utilisée pour les classifications binaires, lorsqu'un modèle doit déterminer seulement deux labels

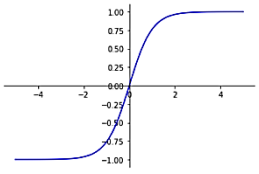


Fonction Sigmoïde

fonction_tanh(x) = sinh(x)/cosh(x)

fonction_tanh(x) = ((exp(x) – exp(-x))/(exp(x) + exp(-x)))

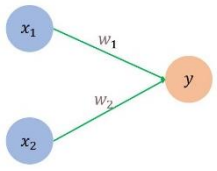
La fonction **tanh**: Il s'agit en fait d'une version mathématiquement décalée de la fonction sigmoïde.



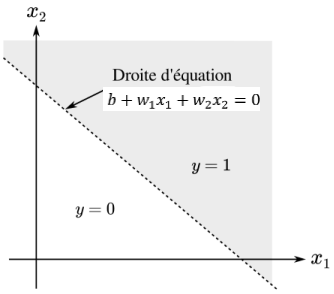
Fonction tanh

L'avantage de tanh est que les entrées négatives seront bien répertoriées comme négatives là où, avec sigmoïde, les entrées négatives peuvent être confondus avec les valeurs proche de nulles.

Influence des paramètres w_i et b



Cas simple avec deux liens on aura $y = a(b + w_1 x_1 + w_2 x_2)$ avec la fonction d'activation Heaviside suivante : y $\begin{cases} 1 \; si \; b + w_1 x_1 + w_2 x_2 \geq 0 \\ 0 \; sinon \end{cases}$



La délimitation entre les deux zones de classification est déterminée par la droite d'équation

$$b + w_1 x_1 + w_2 x_2 = 0 \text{ soit } x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2} = 0.$$

Donc en gérant les paramètres de poids w_1 et w_2, on modifie la pente de la droite.

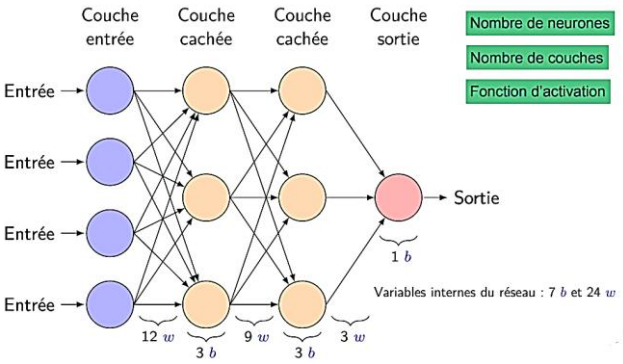
Et en gérant le paramètre de biais b, on modifie l'ordonnée à l'origine.

Ici la fonction d'activation affecte à y la valeur 0 ou 1 mais en utilisant une autre fonction d'activation on pourra affecter une valeur différente à y.

L'avantage principal des fonctions sigmoid (probabilité) et tanh par rapport à la fonction Heaviside est le fait qu'elles soient dérivables ce qui permet d'implémenter les algorithmes d'apprentissages basés sur une descente de gradient.

Donc les paramètres ajustables lors de la phase d'apprentissage sont :

- Les poids w_i affectés à chaque lien
- Les biais b_i de chaque neurone



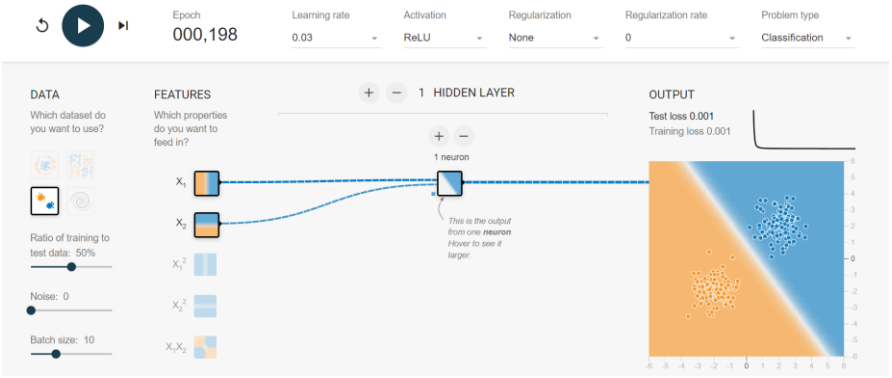
Ces paramètres choisis au départ au hasard vont être ajustés par le deep learning lors de la phase d'apprentissage.

Ils sont modifiés à chaque itération lors du *Rétropropagation* et surtout du *Gradient*

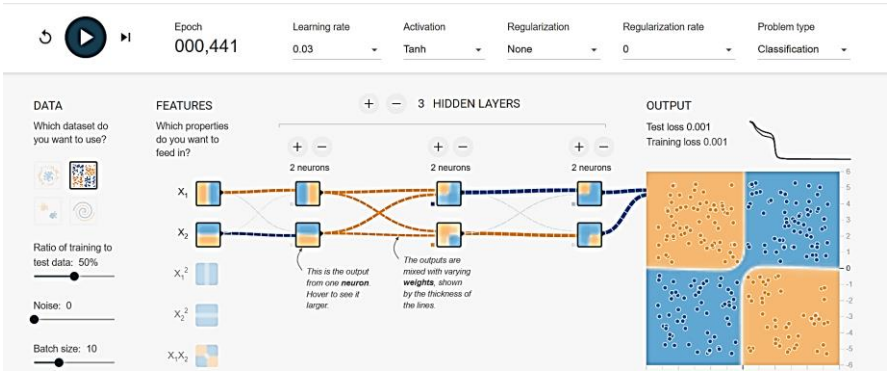
Utilisation de Tensor flow pour réaliser des « *machine learning* »

https://playground.tensorflow.org/

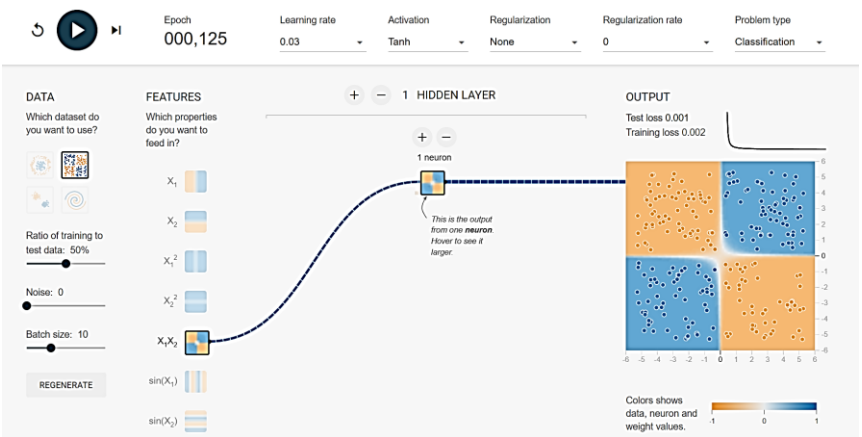
Cas simple de classification de données séparable par une droite : Un neurone est suffisant.



Cas plus compliqué de données : il va falloir plus de couches cachées et plus de neurones

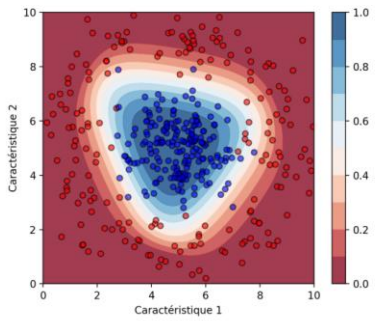


Mais on aurait pu changer la façon de traiter les données d'entrée

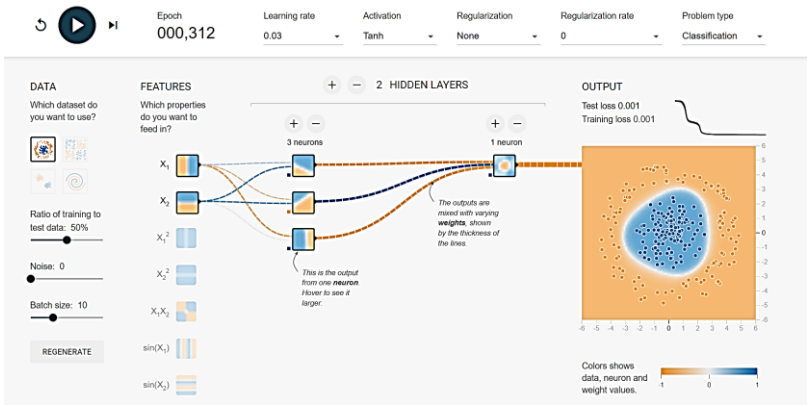


Quelle deep learning peut-on mettre en place pour d'autre type de données ?

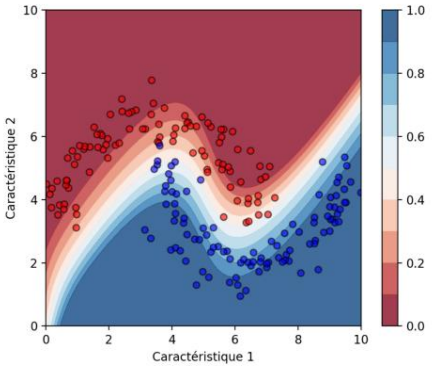
Les données ne sont pas toujours parfaitement séparables :



Classification avec 3 neurones et activation Tanh().

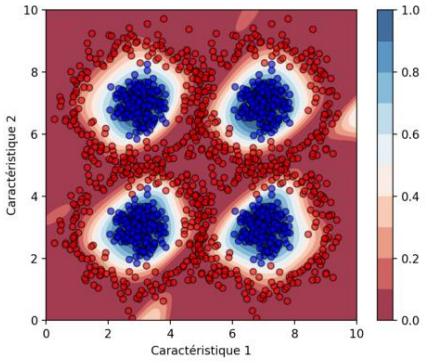


Les données ne sont pas toujours parfaitement séparables :



Classification avec 4 neurones et activation Tanh().

Les données ne sont pas toujours parfaitement séparables :



Classification avec 15 neurones et activation Tanh().

On peut mesurer la qualité d'un système de classification par sa ***matrice de confusion***.

En apprentissage supervisé, la matrice de confusion est une matrice où :

- chaque ligne correspond à une classe réelle ;
- chaque colonne correspond à une classe estimée.

Chaque donnée mal classée est appelée un faux positif/négatif, chaque donnée bien classée est appelée un vrai positif/négatif.

Exemple :

Exemple

Pour la régression logistique obtenue figure 16, possédant 50 exemples de 0 et 50 exemples de 1, la matrice de confusion est la suivante :

		Classe estimée (par le classifieur)	
		1	0
Classe réelle (données exemples)	1	**48** (vrais positifs)	**2** (faux négatifs)
	0	**3** (faux positifs)	**47** (vrais négatifs)

L'intérêt de cette matrice est qu'elle montre rapidement si un système de classification parvient à classifier correctement ou non. Ici, sur 100 exemples seulement 5 sont mal classés : ce système de classification est correct à 95 %.

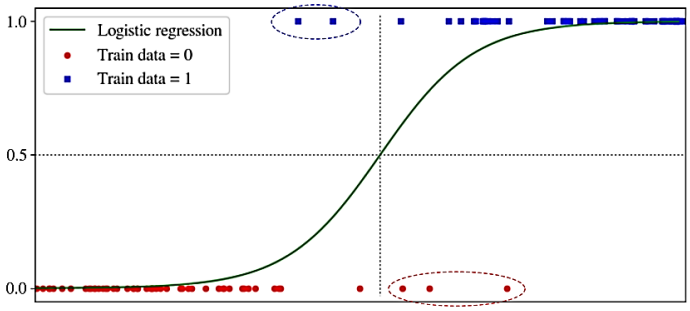
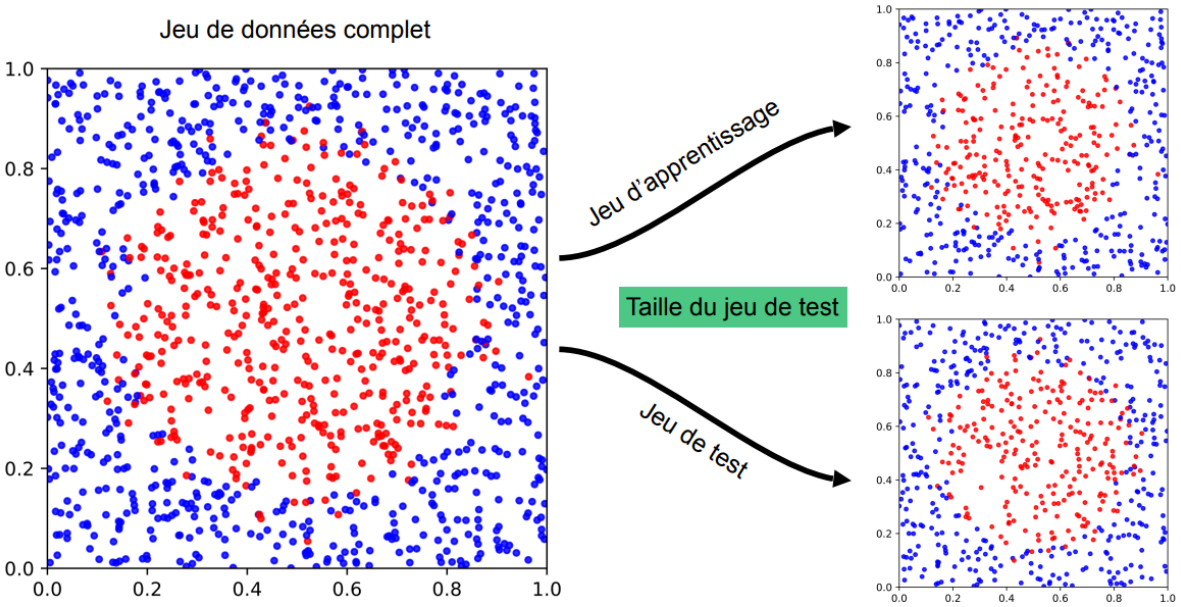
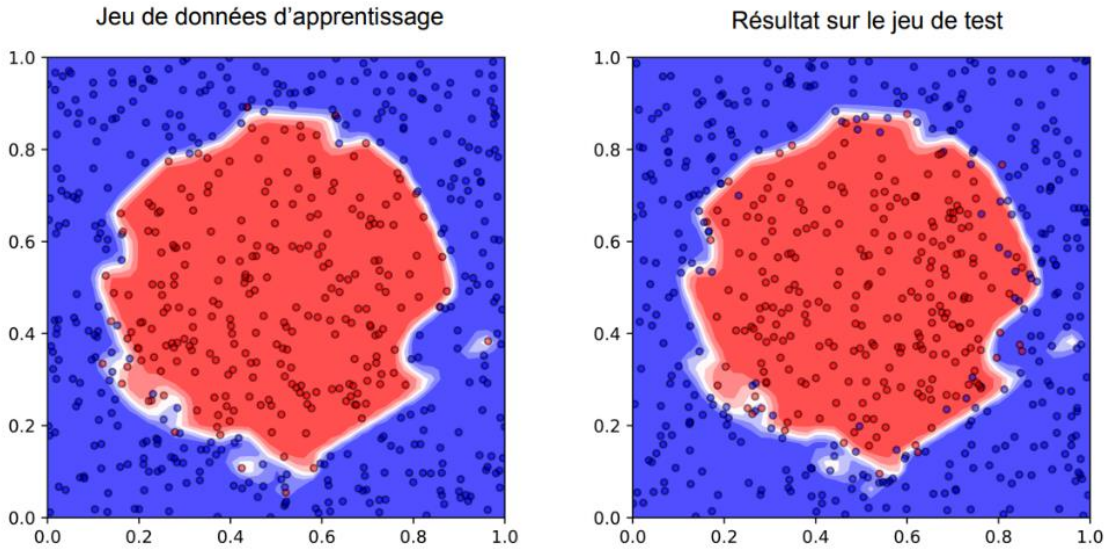


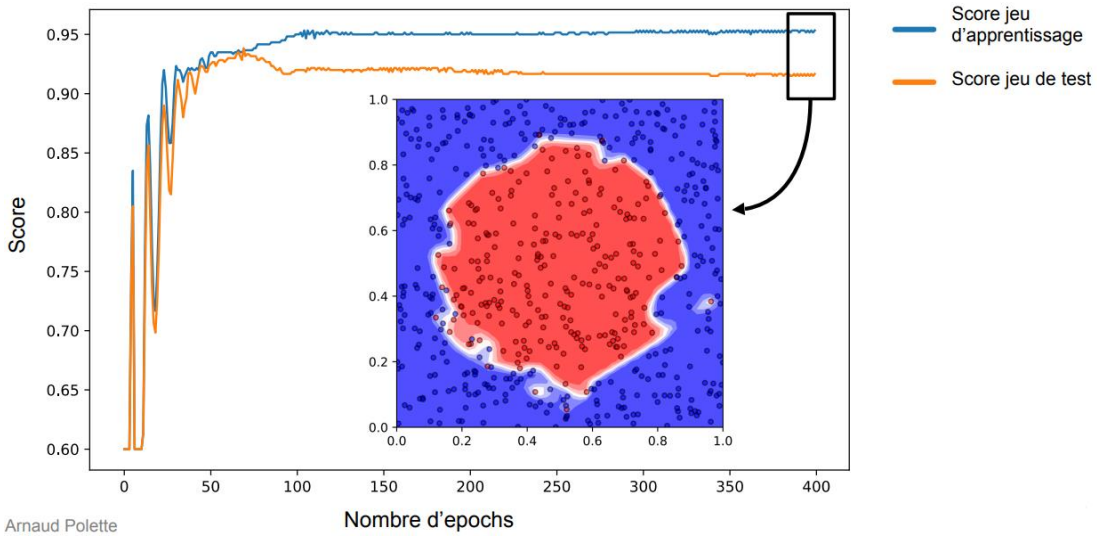
FIGURE 16 – Régression logistique par utilisation d'un Perceptron

Un classificateur sera d'autant plus correct que sa matrice de confusion s'approchera d'une ***matrice diagonale***.

Après la phase d'apprentissage, il faut vérifier que le deep learning est bien entrainé et donc faire des tests de vérification

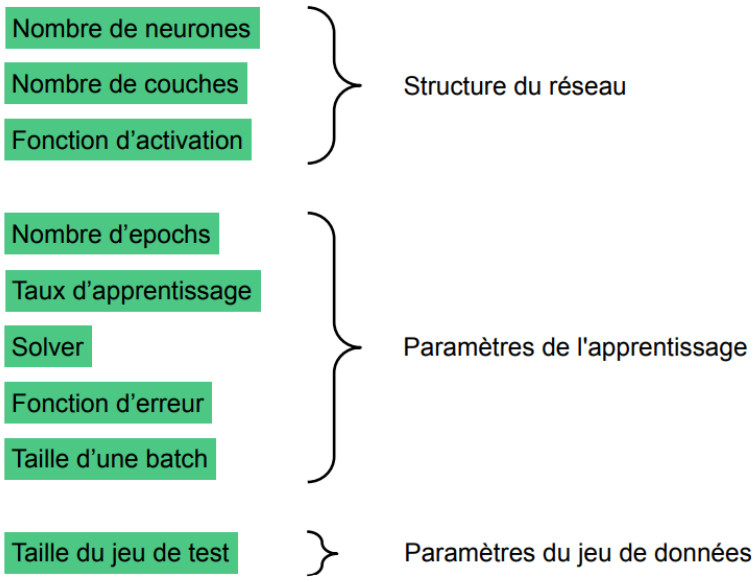






Attention au phénomène de sur apprentissage.

Bilan



Apprentissage SUPERVISE ou NON SUPERVISE

On appelle apprentissage des réseaux de neurones la procédure qui consiste à estimer les paramètres des neurones du réseau, afin que celui-ci remplisse au mieux la tâche qui lui est affectée.

L'apprentissage est dit ***supervisé***, car cela signifie qu'un « *professeur* » peut fournir au réseau des « *exemples* » de ce que celui-ci doit faire (métaphore).

L'apprentissage est dit ***non supervisé,*** à l'inverse de l'apprentissage supervisé, il n'y a pas là de « *professeur* » puisque c'est au réseau de découvrir les ressemblances entre les éléments de la base de données, et de les traduire par une proximité ou ressemblance dans la « carte » de dimension 2 qu'il doit produire.

Ici la notion de « professeur » correspond au fait que dans les bases de données servant pour l'apprentissage et la vérification, à toute entrée est affectée une sortie vraie connue.