University Mohamed Boudiaf of M'sila
Faculty of Mathematics and Informatics

Introduction à l'Intelligence Artificielle

Chapitre II: Réseaux de Neurones Artificiels RNA

Dr. SAID KADRI

Associate Professor

Department of Computer Science, Faculty of Mathematics and Informatics,

University Mohamed Boudiaf of M'sila

E-mail: kadri.said28@gmail.com

Website: https://kadrisaid28.wixsite.com/sgadri

2021 – 2022

2. Réseau de neurones (Réseau neuronaux)

Motivation

- L'humain est capable de manipuler des grandes quantités de connaissances à l'aide son cerveau
- Le cerveau est constitué de neurones (cellules nerveuses) qui constituent les éléments fondamentaux du système nerveux.
- Un neurone est constitué de : noyau, axone, dendrites

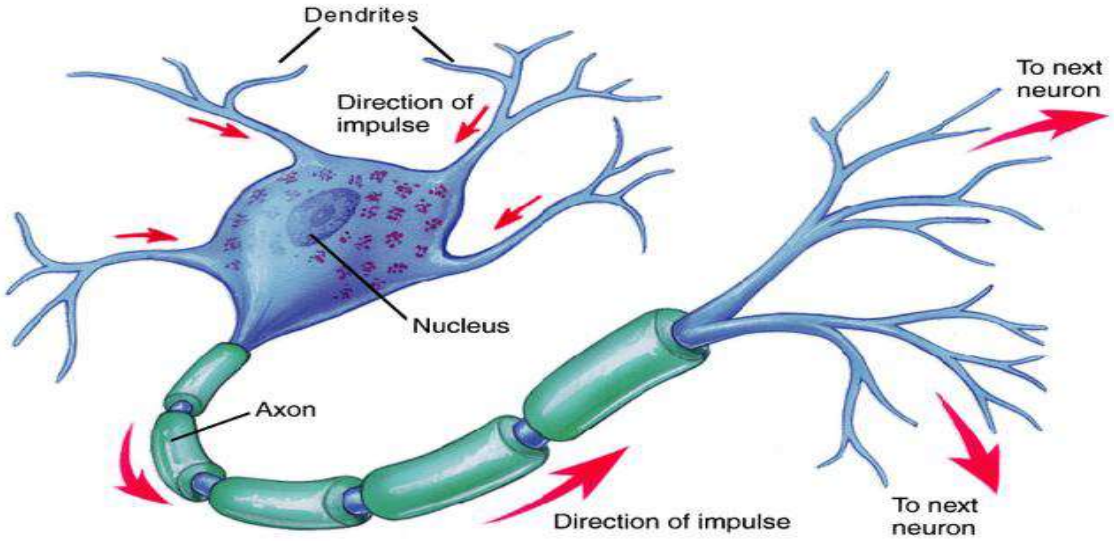


Fig II.5. Architecture du neurone biologique

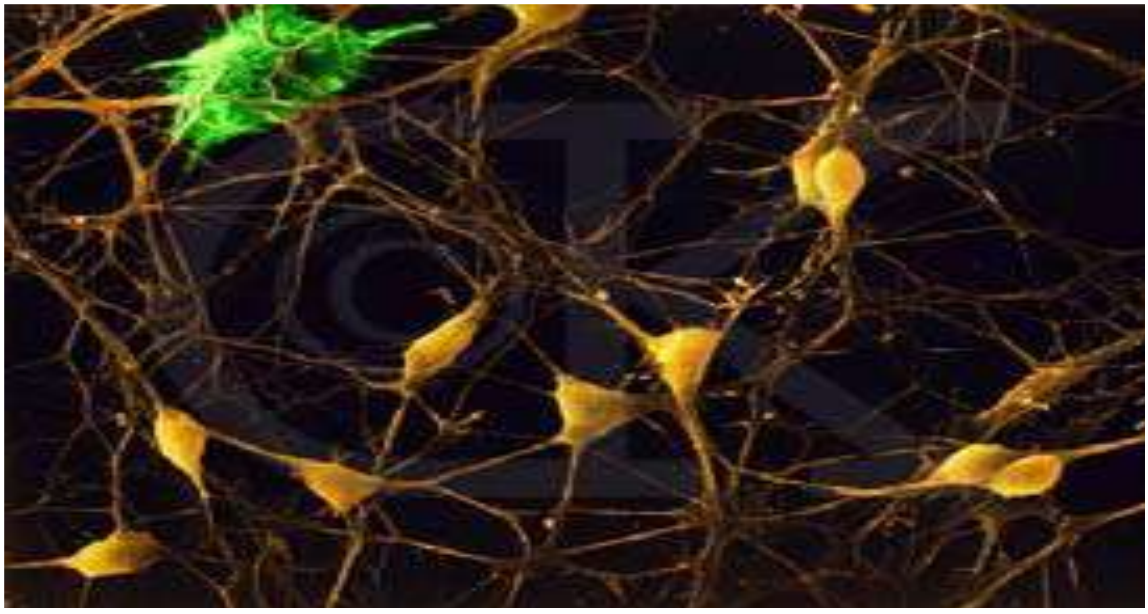
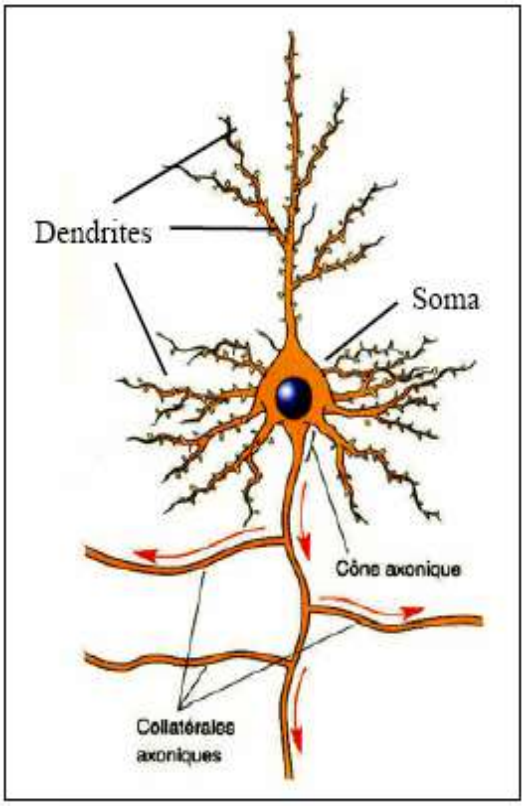


Fig II.6. Architecture du système nerveux

Caractéristiques du neurone biologique



- **Diamètre de la cellule:** 0.01 - 0.05 mm,
- **Soma (corps cellulaire)**
- Formes variables (gén. sphériques),
- 20 µm de diamètre,
- Contient le noyau,
- Entourée d 'une membrane de 5 nm d'ép.
- **Axone**
- Unique aux cellules nerveuses,
- Diamètre: 1-25 µm (humain),
- Longueur: 0.1 mm to 1 mm. (!!!),
- Connexion vers les autres neurones (synapses),
- Permet la transmission d'information,
- **Dendrites**
- Reçoivent les signaux des autres neurones,
- Chacune couverte de centaines de synapses.

Chiffres Importants

- Nombre de neurones dans le cerveau: ~ 10^{11}
- Nombre de connexions par neurone: ~ 10^{4} - 10^{5}
- Temps de cycle (switching time : début de signal → activation): ~ 10^{-3} seconde
- Temps moyen d'une activité cognitive: ~ 10^{-1} seconde

(ex. reconnaissance de visages)

⇨ Il n'y a donc de la place que pour 100 cycles de traitement ($10^{-1}/10^{-3}$)

ce qui est insuffisant pour une activité complexe !!!

⇨ Le cerveau doit donc effectuer des opérations en parallèle !!!

Caractéristiques du système nerveux

- Robuste et tolérant aux erreurs et aux "pannes".
- Flexible et doté de capacités d'apprentissage, pas besoin d'être explicitement "programmé" (plasticité synaptique).
- Capable de traiter de l'information partielle (floue), incertaine (probabiliste) ou incomplète.
- Très massivement parallèle.
- Les capacités (intelligence?) résident dans les connexions entre les milliards de neurones du cerveau humain.

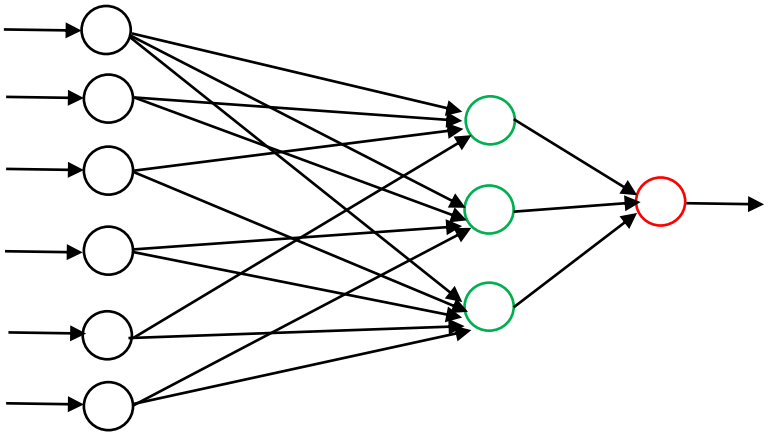
Principe de fonctionnement d'un neurone biologique

Chaque neurone reçoit des impulsions électriques à travers ses dendrites, si ces impulsions sont suffisamment importantes, l'axone transmet une impulsion électrique pour exciter les différents neurones connectés qui seront à leur tour activés en fonction de l'intensité des impulsions transmises.

Réseaux de neurones artificiels/Formels (RNA)

Principe

- Comme l'on a expliqué auparavant, un RNA simule le système nerveux biologique
- Un réseau de neurones est composé de plusieurs neurones interconnectés.
- Un poids est associé à chaque arc.
- A chaque neurone on associe une valeur de sortie.



Domaines d'application

1. **Contrôle**
   - Pilotage automatique:

     Alvinn: réseau de pilotage d'un véhicule à partir d'images vidéo,
   - Synthèse vocale:

     NETtalk: un réseau qui apprend à prononcer un texte en anglais
   - Processus de fabrication/production,
2. **Reconnaissance/classification/analyse**
   - Textes imprimés, caractères manuscrits (codes postaux), parole,
   - Images fixes (ex. visages), animées ou clips vidéo,
   - Risques (financiers, naturels, etc.)
3. **Prédiction**
   - Economie, finances, analyse de marchés, médecine,

Domaines de succès

- Reconnaissance de la parole (TDNNs)
- Contrôle robotique (ALVINN)
- Reconnaissance de caractères (OCR),
- Prévisions financières.

Relation avec le réseau de neurone biologique

- Parallélisme massif
- Calcul local
- Elément simple de calcul de type "neurone artificiel".

Quelques repères historiques

1943: J. McCulloch& W. Pitts

- Proposent un modèle simple de neurone capable de produire une machine de Turing,
- Démontrent qu'un assemblage synchrone de tels neurones est une machine universelle de calcul (c.à.d. n'importe quelle fonction logique peut être représentée par des unités à seuil).

Warren S. McCulloch

William Pitts

1948: D. Hebb : Propose une règle d'apprentissage pour des réseaux de neurones.

1958: F. Rosenblatt : Propose le modèle du Perceptron et démontre son théorème de convergence.

1969: M. Minsky& S. Papert : Démontrent les limitations du modèle du Perceptron.

1985: Apparition d'apprentissage par rétro-propagation pour les réseaux multi-couches.

Architecture du neurone formel

McCulloch et Pitts (1943) ont modélisé le fonctionnement d'un neurone biologique et imaginé une reproduction artificielle mimétique du raisonnement humain.

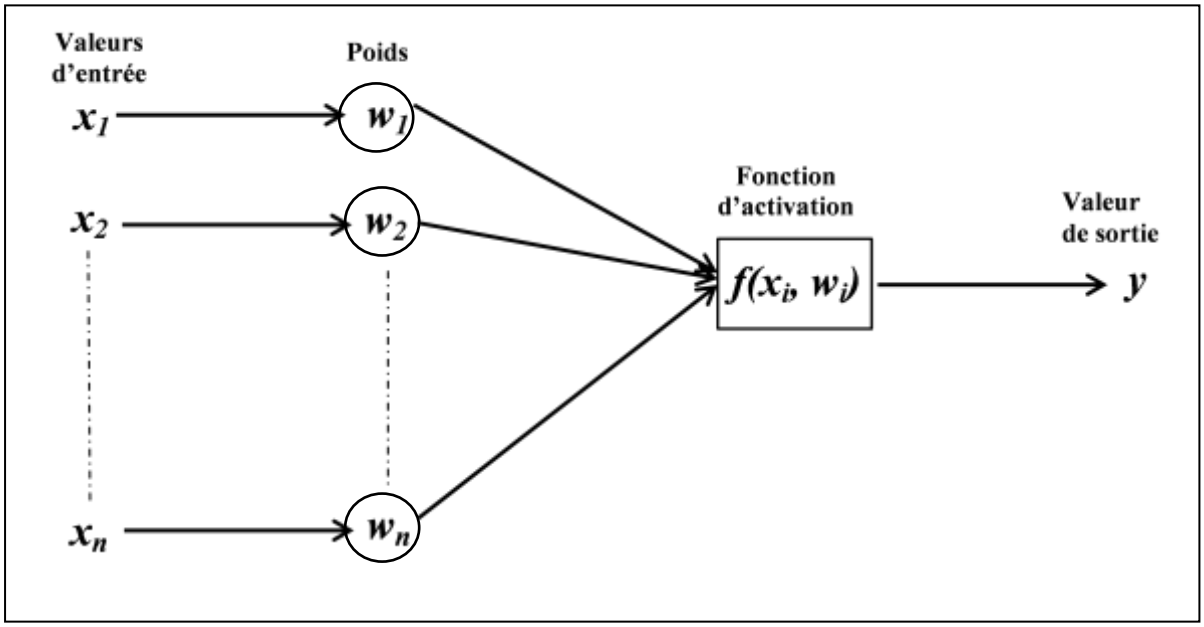
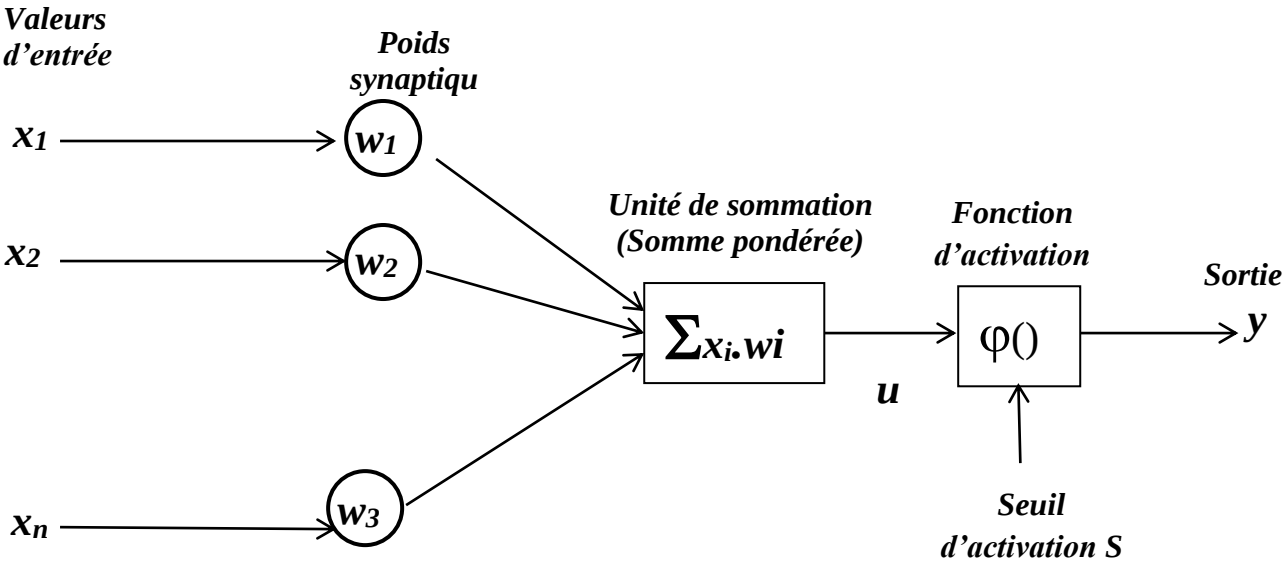


Fig II.6. Architecture du réseau de neurone formel

Principe de fonctionnement d'un neurone formel (modèle de McCulloch et Pitts/Modèle linéaire)

Un réseau de neurone formel ou aussi appelé automate à seuil base sur le même principe d'un réseau de neurone biologique comme c'était expliqué auparavant



Ce modèle base sur deux équations:

$$u = \Sigma x_i \, .wi \qquad (1) \quad \textit{(somme pondérée)}$$

$$y = \varphi(u - S) \qquad (2) \quad \textit{(fonction d'activation)}$$

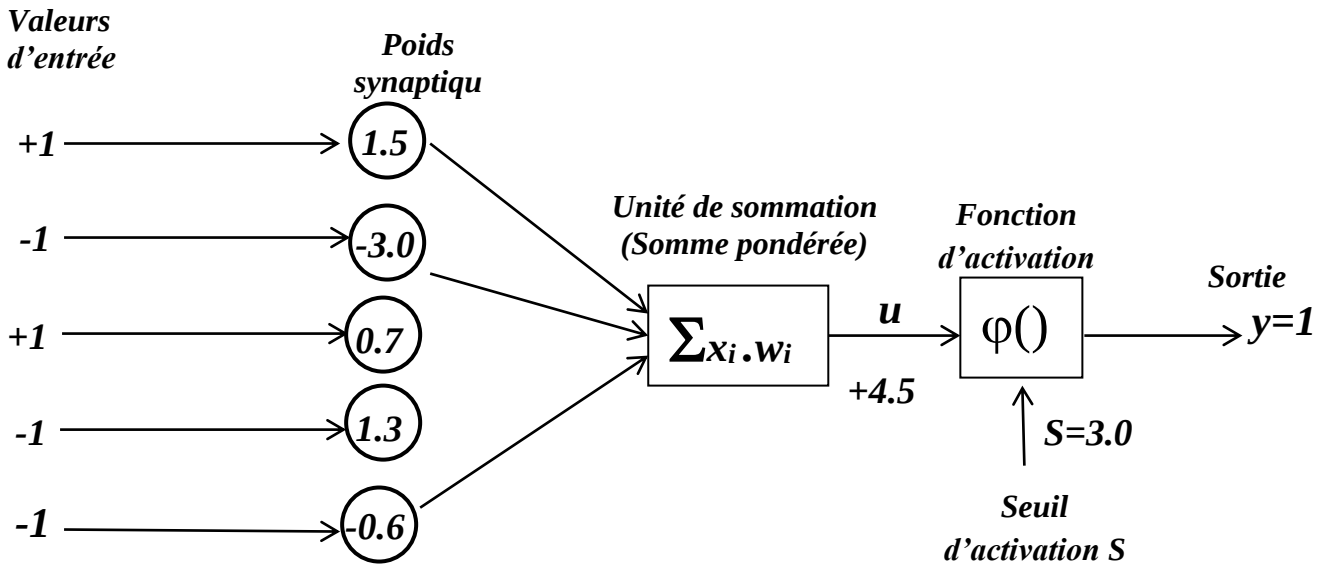
avec :

$x_1, x_2, ..., x_n$: sont les entrées,
$w_1, w_2, ..., w_n$: sont les poids synaptiques du neurone k,
u : la sortie de l'unité de sommation,
S : le seuil,
$\phi(.)$: la fonction d'activation,
y : le signal de sortie du neurone k.

Valeur de sortie : le neurone formel calcule la somme des entrées x_1, x_2, …, x_n pondérées par les poids synaptiques w_1, w_2, …, w_n et la comparer avec le seuil S, si le résultat est supérieur au seuil, alors la valeur renvoyée en sortie est +1 (activation du neurone), sinon la valeur renvoyée est 0

$$y = \begin{cases} +1 & si \sum xi \, . wi > S \\ 0 & sinon \end{cases}$$

Exemple : Fonctionnement d'un RNA (modèle de McCulloch & Pitts)



Caractéristiques des RNA

- Capacité d'apprentissage : apprendre et changer son comportement en fonction de toute nouvelle expérience.
- Permettent de découvrir automatiquement des modèles complexes.
- Plusieurs modèles de réseaux de neurones : PMC (Perceptron Multi-Couches), RBF (Radial Basis Function), Kohonen, ...
- Méthode de classification: Ajuster les poids en utilisant l'erreur

  Erreur= Valeur désirée–Valeur actuelle.

Quelques directives pour construire un modèle RNA

- Représentation des entrées
- Nombre de nœuds en entrée : correspond à la dimension des données du problème (attributs ou leurs codages).
- Déterminer le nombre de couches, nombre de nœuds dans chaque couche
- Nombre de couches cachées : Ajusté pendant l'apprentissage.
- Nombre de nœuds par couche : le nombre de nœuds par couche est au moins égal à deux et au plus égal au nombre de nœuds en entrée
- Nombre de nœuds en sortie : fonction du nombre de classes associées à l'application.
- Réseau riche ➔ pouvoir d'expression grand (Ex. 4-2-1 est moins puissant que 4-4-1)
- Attention : Choisir une architecture riche mais pas trop – chargé ==>Problème de sur-spécialisation (Overfitting)

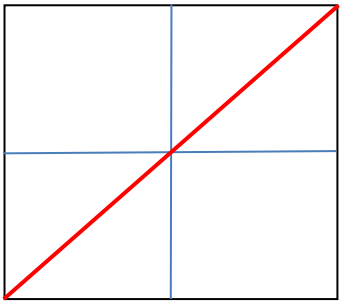
Méthodes d'apprentissage des RNA

- Méthode de rétropropagation du gradient (Back propagation)
- Méthode de Kohonen
- Méthode RBF (Radial Basis Function)
- Réseaux de neurones probabilistes
- ART (Adaptive Resonance Theory)

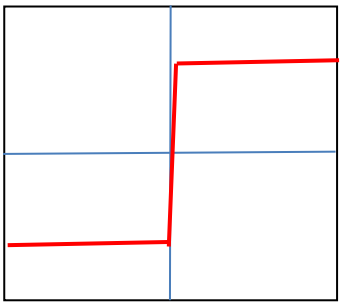
Fonction d'activation

La fonction d'activation définit la valeur de sortie d'un neurone en termes des niveaux d'activité de ses entrées. Cette fonction peut avoir plusieurs formes (linéaire, exponentielle, sigmoïde, …)

1. ***Fonction linéaire (Linear Function):*** Utilisée en couche de sortie surtout pour une régression. On peut la caractériser de nulle, puisque les unités de sortie seront identiques à leur niveau d'entrée. Son intervalle de sortie est (-∞;+∞) pour une fonction continue et [-1, +1] pour une fonction discrète.

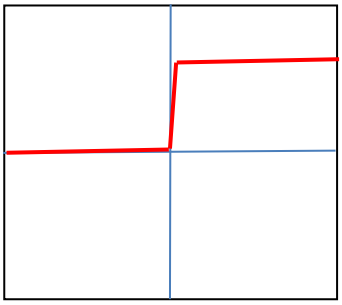


$$f(x) = ax + b$$



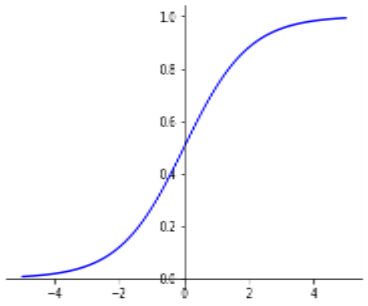
$$f(x) = \begin{cases} +1 & x > 0 \\ -1 & sinon \end{cases}$$

2. ***La fonction Step :*** Elle renvoi tout le temps 1 pour un signal positif, et 0 pour un signal négatif.



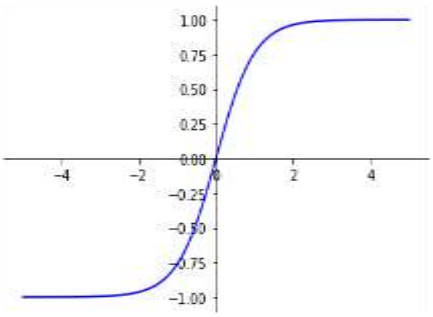
3. ***La fonction Sigmoid (logistic) :*** la fonction la plus populaire depuis des décennies. Mais aujourd'hui, elle devient beaucoup moins efficace par rapport à d'autres. Elle est utilisée surtout pour les couches cachées et aussi utilisée en couche de sortie pour de la classification binaire. Son intervalle de sortie est : [0,1] et sa formule est définie comme suit :

$$\boldsymbol{sigmoid(x) = \frac{1}{(1 + e^{-x})}}$$



4. ***La fonction tanh :*** est simplement la fonction de la tangente hyperbolique. Il s'agit en fait d'une version mathématiquement décalée de la fonction ***sigmoïde*** avec la seule différence c'est que la fonction sigmoïde donne un résultat entre 0 et 1 par contre la fonction tanh donne un résultat entre -1 et 1. 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. Cette fonction est (comme Sigmoïde) utilisé dans la classification binaire.

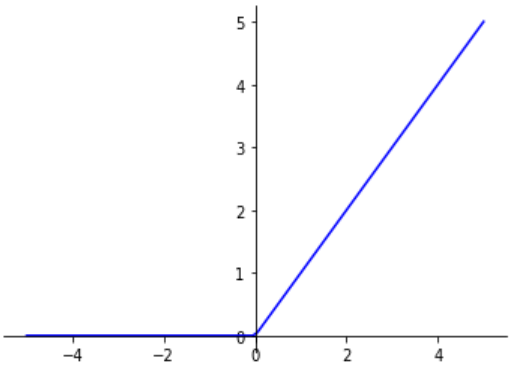
$$\boldsymbol{tanh(x) = \frac{1 - e^{-2x}}{(1 + e^{-2x})}}$$



5. ***La fonction Rectified Linear Unit (ReLU)*** : est la fonction d'activation la plus simple et la plus utilisée. Elle donne x si x est supérieur à 0, 0 sinon. Autrement dit, c'est le maximum entre x et 0 :

$$\boldsymbol{ReLU(x) = \begin{cases} x & x > 0 \\ 0 & sinon \end{cases}} \quad \textbf{OU} \quad \boldsymbol{ReLU(x) = max(x, 0)}$$

Cette fonction permet d'effectuer un filtre sur nos données. 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.

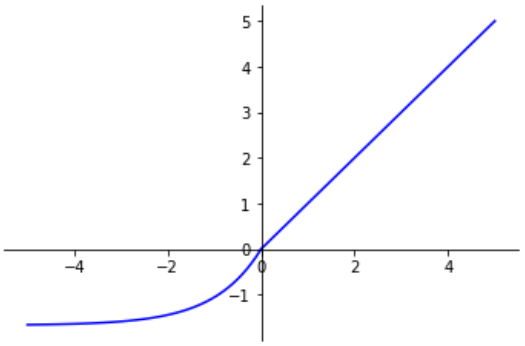


6. ***La fonction Exponential Linear Unit (ELU)*** : est une amélioration de ReLU car elle permet d'avoir des valeurs lisses lorsque x < 0.

C.à.d., lorsque x < 0, ELU a des valeurs négatives différents de 0 (ce qui n'est pas le cas pour ReLU). Cela permet de rapprocher la moyenne de la fonction de zéro. Une moyenne plus proche de zéro permet un apprentissage plus rapide.

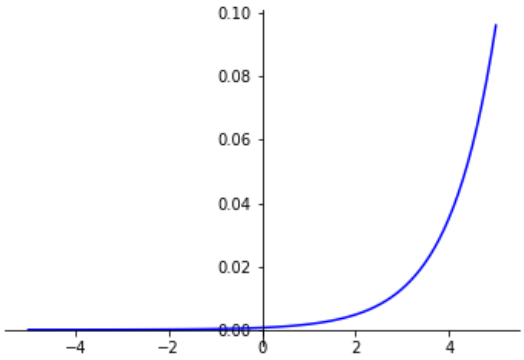
Effectivement, plus x diminue, plus ELU saturent à une valeur négative. Cette saturation implique qu'ELU a une petite dérivée ce qui diminue la variation du résultat et donc l'information qui est propagée vers la couche suivante.

$$ELU(x) = \begin{cases} x \quad si\ x > 0 \\ \alpha * (e^x - 1) \quad si\ x < 0 \end{cases} \qquad avec\ \ \alpha > 0$$



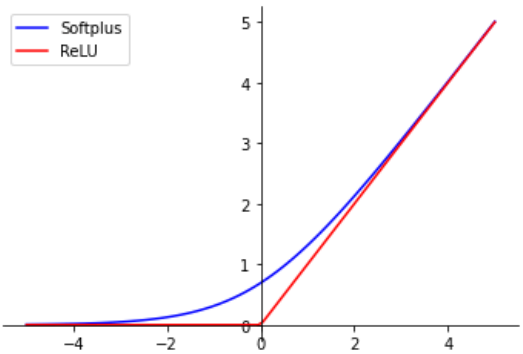
7. ***La fonction Softmax :*** La fonction Softmax permet de transformer un vecteur réel en un vecteur de probabilité. On l'utilise souvent dans la couche finale d'un modèle de classification, notamment pour les problèmes multi-classes. Dans la fonction Softmax, chaque vecteur est traité indépendamment.

$$\boldsymbol{Softmax(x) = \frac{e^x}{\sum_{i=1}^{n} e^{xi}}}$$



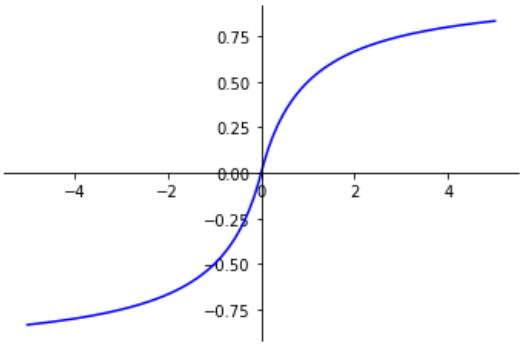
8. ***La fonction Softplus :*** est une approximation 'lisse' de la fonction ReLU. Cet aspect 'lisse' (ou soft) implique que la fonction est différentiable. En fait, cette fonction est intéressante par sa dérivée. Quand on dérive Softplus, on obtient la fonction logistique *f(x) = 1/(1+exp(-x))*.
   On rappelle que la dérivée est utilisée lors de la Backpropagation pour mettre à jour les poids.
   Elle était utilisée pour contraindre le résultat d'une couche à être toujours positif mais a été remplacé par ReLU qui est linéaire et donc beaucoup plus rapide à calculer.

$$\boldsymbol{Sofplus(x) = Log(e^x + 1)}$$



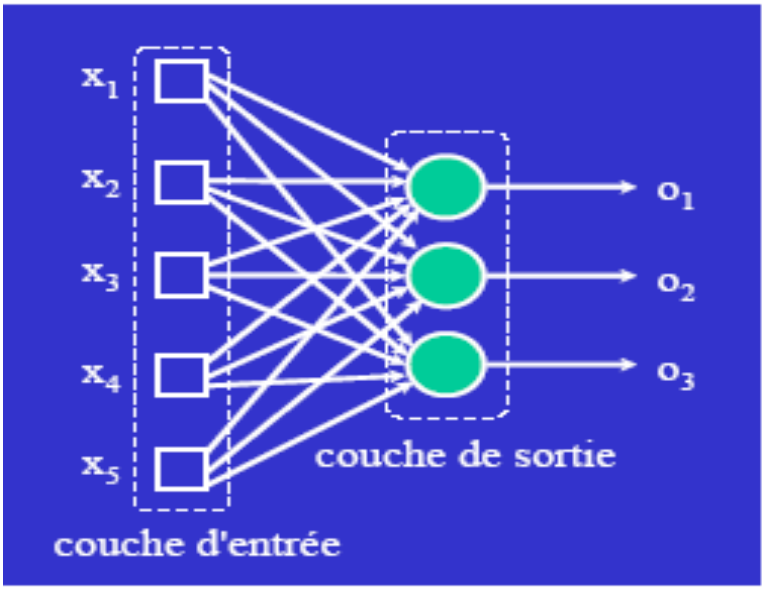
9. ***La fonction Softsign :*** La fonction est utile pour normaliser nos données car elle permet d'avoir un résultat entre -1 et 1 et garde en mémoire le signe des données (positif ou négatif). Autrement dit, les données sont recentrées sur zéro et bornées par -1 et 1.
   En fait, c'est la fonction signe lissé (Softsign) et donc différentiable Backpropagation oblige).

$$\boldsymbol{Sofsign(x) = \frac{x}{|x| + 1}}$$



Topologies de réseaux de neurones

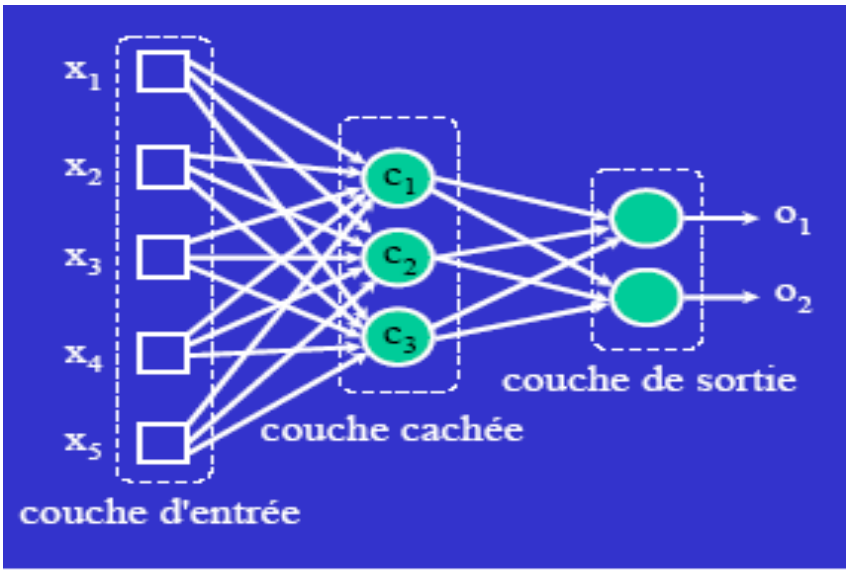
1. Réseau acyclique ("feedforward") à une seule couche



$$o_i = 1 \quad \text{si} \quad \sum_k w_{ik} x_k > 0$$

$$o_i = 0 \quad \text{sinon}$$

2. Réseau acyclique multi-couches



Valeurs de sortie :

Couche cachée

$c_j = 1$ si $\sum_k w_{jk} x_k > 0$

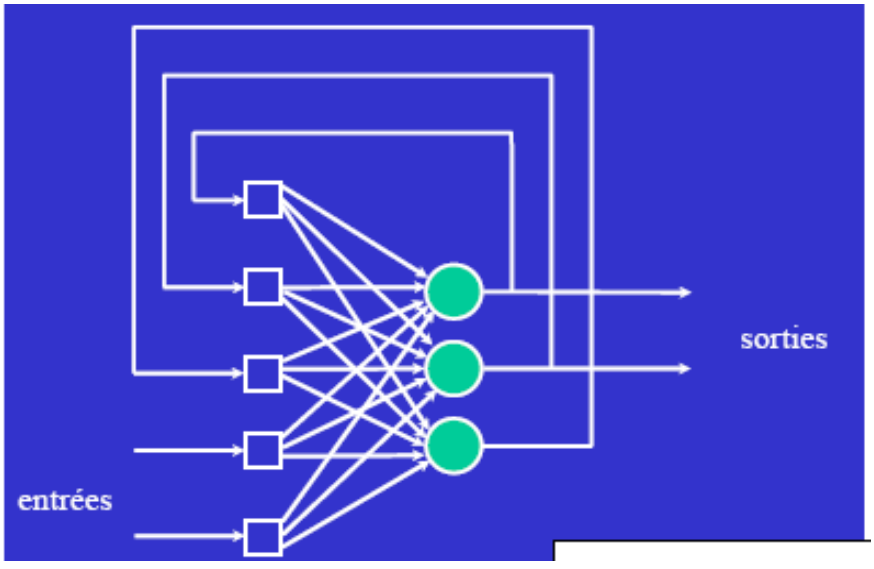
$c_j = 0$ sinon

Couche de sortie

$o_i = 1$ si $\sum_k w_{ik} c_k > 0$

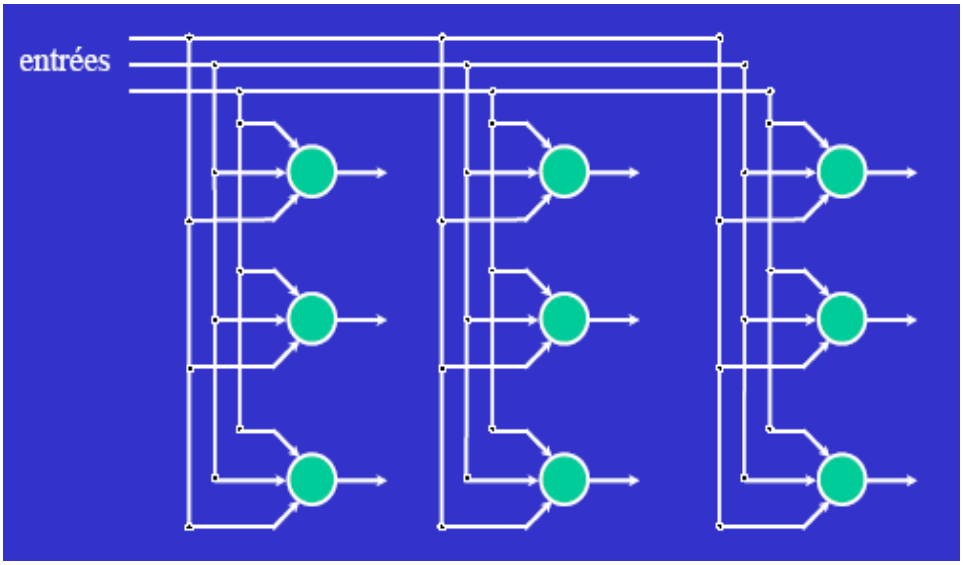
$o_i = 0$ sinon

3. Réseau récursif (réseau de Hopfield)



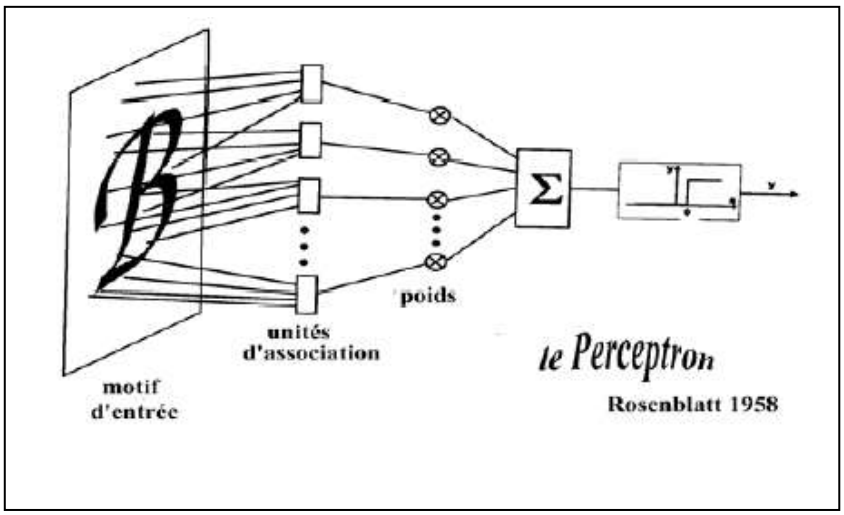
OBS : Chaque unité i est connectée à chaque autre unité j par un poids w_{ij} les poids sont supposés symétriques: $w_{ij} = w_{ji}$

4. Réseau "en treillis"



Réseau en treillis 3x3 bi-dimensionnel

Le Perceptron (F. Rosenblatt, 1958)

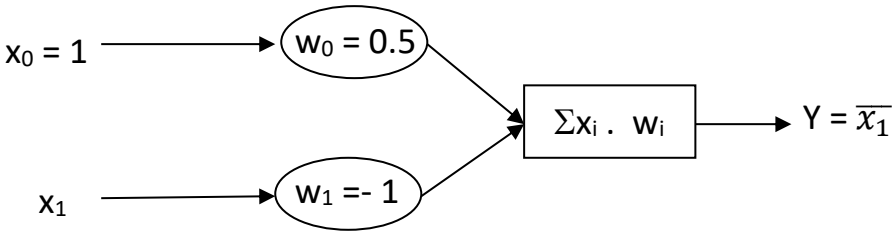


Avec :

$$x_0 = 1 \quad et \quad \begin{cases} 1 & si \ \sum x_i . w_i > 0 \\ 0 & sinon \end{cases}$$

Exemple : Réalisation d'un NON logique par un perceptron

Entrée x_1	Sortie
0	1
1	0



Exercice: Réaliser les portes logiques OR, AND, XOR en utilisant un perceptron (Voir TD)

Théorème d'apprentissage (F. Rosenblatt)

Étant donné suffisamment d'exemples d'apprentissage, il existe un algorithme qui apprendra n'importe quelle fonction linéairement séparable.

Algorithme d'apprentissage du Perceptron

- Obtenir un ensemble de poids qui font que la plupart des instances de l'ensemble d'apprentissage sont correctement classées.

- ***Etapes :***
  - Poids initiaux sont générés aléatoirement
  - Les vecteurs en entrée sont traités en séquentiel par le réseau
  - Calcul des valeurs d'activation des nœuds cachés
  - Calcul du vecteur de sortie
  - Calcul de l'erreur (sortie désirée –sortie actuelle).
  - Les poids sont mis à jour en utilisant l'erreur.

$$E(PMC) = \frac{1}{2}\sum_{x \in S}^{n} (d(x) - a(x))^2$$

Avec :

x : un exemple de la base d'apprentissage S

d(x) : sortie désirée a(x) : sortie calculé par le RNA

- Les poids sont mis à jour en utilisant l'erreur.
- L'apprentissage se fait selon la loi de retro-propagation de Hebb vue précédemment :

Entrées: *ensemble d'apprentissage {(x1,x2,…,xn,t)}*

Méthode

initialiser aléatoirement les poids w(i), 0<=i<=n

Répéter *jusqu'à convergence:*

Pour *chaque exemple*

Calculer la valeur de sortie o du réseau.

Ajuster les poids:

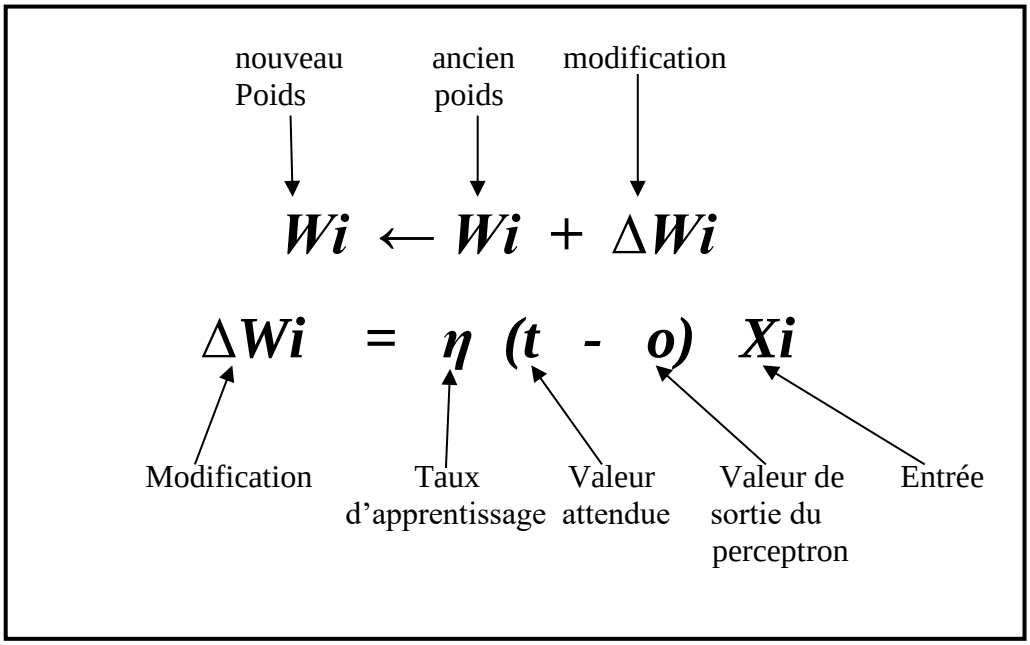
$\Delta wi = \eta(t - o)\, xi$ /* *t : sortie attendue, O : sortie calculée*

$wi \leftarrow wi + \Delta wi$ /* *par le perceptron*

FPour

Fin Répéter

Fin Méthode



- Le paramètre taux d'apprentissage $\eta \in [0,1]$ influe sur la modification des poids. (Valeur grande➔ modification forte ; Valeur petite ➔ modification minime)
- Critère d'arrêt : la tolérance définit l'erreur cible.
- et/ou Nombre d'instances bien classées (seuil)

Elagage du réseau

- *N* nœuds en entrée, *h* couches cachées, et *m* nœuds en sortie *h(m+n)* arcs (poids)

⇨ ***Elagage :*** Supprimer les arcs et les nœuds qui n'affectent pas le taux d'erreur du réseau. Eviter le problème de sur-spécialisation (over-fitting).

- Ceci permet de générer des règles concises et claires.

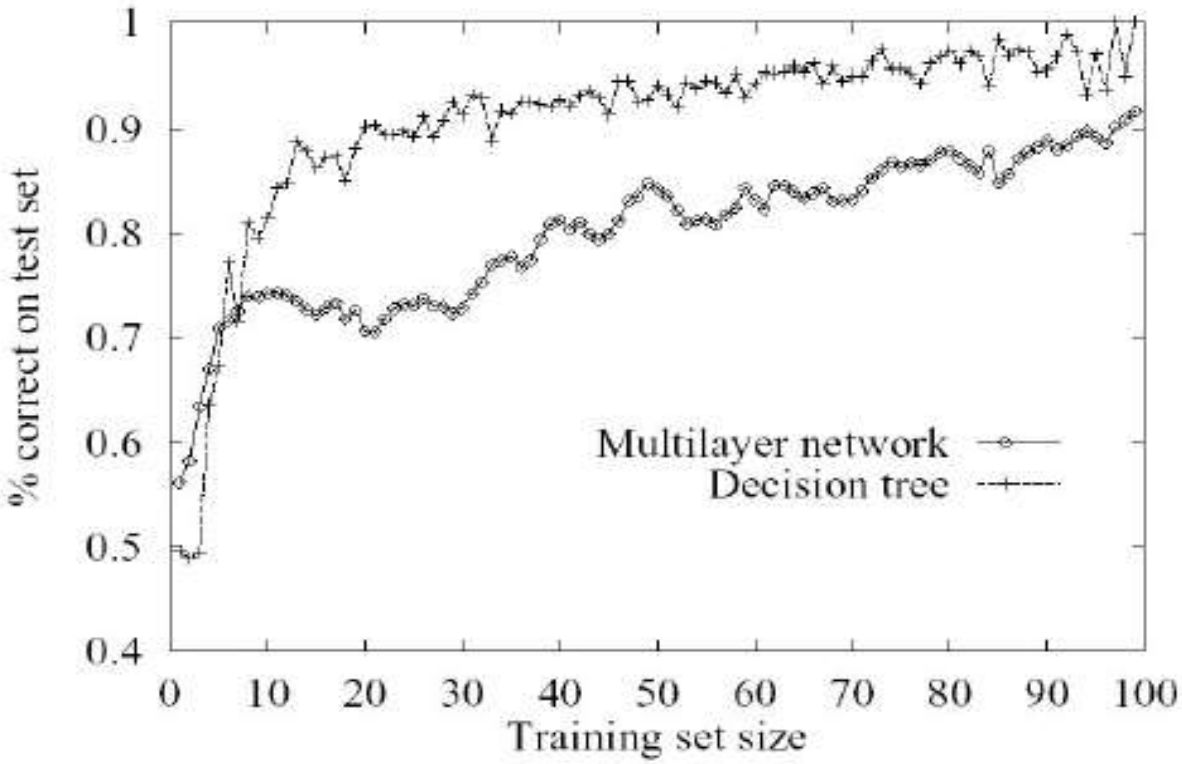
Réseaux de neurones RNA : Avantages

- Taux d'erreur généralement bon
- Outil disponible dans les environnements de data mining
- Résistance au bruit ➔ reconnaissance de formes (son, images sur une rétine, …)
- Classification rapide (réseau étant construit)
- Combinaison avec d'autres méthodes (ex : arbre de décision pour sélection d'attributs)

Réseaux de neurones RNA : Inconvénients

- Apprentissage très long
- Plusieurs paramètres (architecture, coefficients synaptiques, …)
- Pouvoir explicatif faible (boite noire)
- Pas facile d'incorporer les connaissances du domaine.
- Evolutivité dans le temps (phase d'apprentissage)

Réseaux Vs arbre de décision

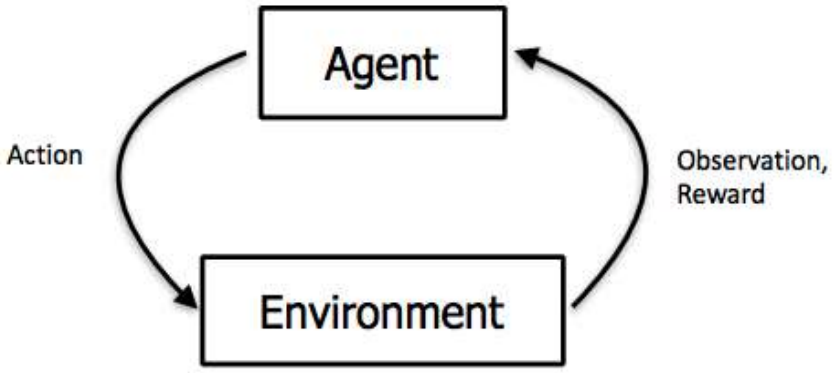


Quand utiliser des réseaux de neurones artificiels pour résoudre des problèmes d'apprentissage ?

- Lorsque les données sont représentées par des paires attributs-valeur,
- Lorsque les exemples d'apprentissage sont bruités,
- Lorsque des temps d'apprentissage (très) longs sont acceptables,
- Lorsqu'une évaluation rapide de la fonction apprise est nécessaire,
- Lorsque la compréhension par l'utilisateur de la fonction apprise est sans importance.

Apprentissage par renforcement (reinforcement learning)

- Est un processus cognitif résultant de la rencontre de la psychologie expérimentale et les neurosciences computationnelles.
- Il est inspiré directement du mécanisme de l'apprentissage humain et animal.
- Pour le cas de l'humain, il peut apprendre des choses par l'expérience, si cette expérience aboutit à un résultat positif et induit des récompenses, on juge que cette expérience est positive et qu'elle doit être répétée. Au contraire, si le résultat de l'expérience est négatif, on la mémorise comme mauvaise expérience pour ne plus la faire dans le futur (ce qui renforce la décision de l'agent humain dans les deux cas).
- Pour le cas de l'animal, il est impossible de lui apprendre des actions en lui expliquant verbalement, mais il est possible de provoquer sa réaction en lui créant une situation. Si sa réaction est bonne, on le récompense (par exemple en lui donnant une tranche de viande), sinon on le punit. Alors, grâce aux récompenses, l'animal sera progressivement conditionné et motivé de faire la même action dès qu'il se retrouvera confronté avec la même situation pour recevoir la récompense (le morceau de viande).
- Dans le cas d'un agent intelligent (programme, robot), il interagit avec son environnement pour trouver la solution optimale de la situation dans laquelle il s'est retrouvé.



- Une différence fondamentale entre les trois types d'apprentissage : supervisé, non supervisé, et l'apprentissage par renforcement et que le dernier est plus interactif (avec l'environnement) et itératif (se répète dans le temps) et ne demande pas de données préalables pour s'entrainer mais il est guidé par les réactions de l'environnement (positives ou négatives).
- Effectivement, l'agent (humain, animal, agent intelligent) essaie plusieurs solutions, observe la réaction de son environnement (positive ou négative) et adapte son comportement pour trouve la solution optimale.

Domaines d'application de l'apprentissage par renforcement

L'apprentissage par renforcement a plusieurs applications, à noter :

- Apprendre un robot à marcher sur un terrain difficile (environnement ouvert).
- Voitures auto-conduites (Self-Driving Cars)
- Pilotage d'un agent artificiel à travers un labyrinthe.
- Les jeux
- Contrôle automatique des feux tri-couleurs.
- Systèmes de recommandation (qui s'adaptent au changement des préférences des utilisateurs. Alors, la récompense dans ce cas et donnée par la réaction positive de l'utilisateur)

Conclusion :

- L'apprentissage par renforcement consiste à apprendre à partir d'expériences successives en contact avec l'environnement de manière à trouver la meilleure solution.
- Il s'agit donc d'un apprentissage guidé par la réaction de l'environnement et comporte généralement trois taches principales :
  - ✓ Observation des effets des actions de l'agent sur l'environnement.
  - ✓ Déduire la qualité de ces actions en se basant sur les observations (positives/négatives)
  - ✓ Améliorer les actions futures