Introduction:

Un réseau de neurones artificiels, ou Artificial Neural Network en anglais, est un système informatique matériel et / ou logiciels issus de l'idée de reproduire le fonctionnement du cerveau humain. Les réseaux biologiques, permettent d'effectuer des fonctions habituelles comme la reconnaissance, la perception ou le contrôle moteur. Ces capacités sont, grâce à la haute complexité de leur structure, formées par un nombre important de cellules nerveuses (neurones) massivement interconnectées et fonctionnant en parallèle.

Le concept des réseaux de neurones artificiels apparait en 1943, quand Walter Pitts et Warren McCulloch proposeront une « représentation » du cerveau base e en cellules interconnectées. Le 1959, Bernard Widrow et MarcialHoff sont les premiers à utiliser les réseaux neuronaux dans un cas réel.

En 1957, le perceptron fut inventé. Il s'agit du **plus ancien algorithme deMachine Learning**, conçu pour effectuer des tâches de reconnaissance de patterns complexes. C'est cet algorithme qui permettra plus tard aux machines d'apprendre à reconnaître des objets sur des images.
Malheureusement, à l'époque, les réseaux de neurones étaient limités par les ressources techniques. Par exemple, **les ordinateurs n'étaient pas assezpuissants**pour traiter les données nécessaires au fonctionnement des réseaux de neurones. C'est la raison pour laquelle la recherche dans le domaine des Neural Networks est restée en sommeil durant de longues années.
Il aura fallu attendre le début des années 2010, **avec l'essor du Big Data etdu traitement massivement parallèle**, pour que les Data Scientists disposent des données et de la puissance de calcul nécessaires pour exécuter des réseaux de neurones complexes. En 2012, lors d'une compétition organisée par ImageNet, un Neural Network est parvenu pour la première fois à surpasser un humain dans la reconnaissance d'image.

C'est la raison pour laquelle cette technologie est de nouveau au coeur des préoccupations des scientifiques. A présent, les réseaux de neurones artificiels ne cessent de s'améliorer et d'évoluer de jour en jour.

1.DU NEURONE BIOLOGIQUE AU NEURONE ARTIFICIEL

Les réseaux de neurones biologiques qui constituent le cerveau humain réalisent simplement de nombreuses applications telles que la reconnaissance de formes, le traitement du signal, la mémorisation, la généralisation, l'apprentissage par l'exemple, etc. Or ces applications sont actuellement à la limite de leurs possibilités actuelles, même en tenant compte de tous les efforts déployés en algorithmique ou en intelligence artificielle. C'est à partir de l'hypothèse que le comportement intelligent humain est le résultat de la structure et des éléments de bases du système nerveux central (que sont les neurones), que l'on a développé les réseaux de neurones artificiels. Cette inspiration à partir du modèle biologique provient du fait que le cerveau humain est un système apprenant basé sur une structure contenant environ 100 milliards de neurones reliés entre eux par 10000 contacts synaptiques ce qui représente un million de milliards de synapses.
Les réseaux de neurones artificiels sont donc un moyen de modéliser le mécanisme d'apprentissage et de traitement de l'information qui se produit dans le cerveau humain.

• Le cerveau humain est compose´ de 10^{12} mille milliard de neurone , avec 1000 à 10000 connexions pour chaque neurone.

Neurones :

Ce sont des cellules reliées entre elles par des axones et des dendrites

En première approche

• on peut considérer que ces sortes de filaments sont conductrices d'électricité et peuvent ainsi véhiculer des messages (dopamines) depuis un neurone vers un autre

Un neurone biologique tel que présentée par la figure est constitué de :

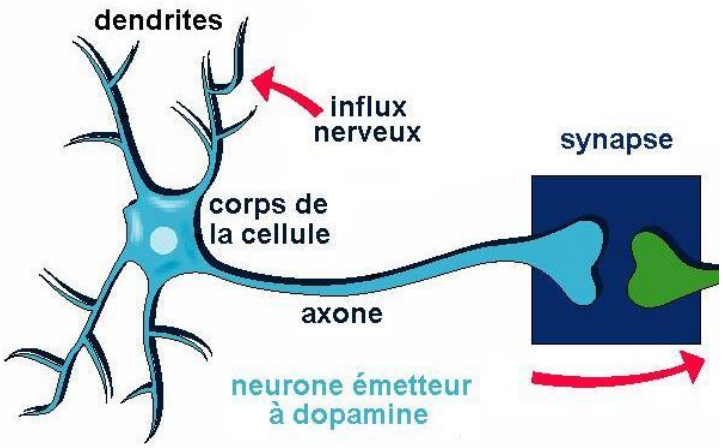


Figure.2 Neurone biologique

Remarque :(La dopamine est une molécule, un ***neurotransmetteur*** *parmi de nombreux neurotransmetteurs...C'est un* ***message chimique*** *u système nerveux)*

Correspondance neurone biologique/formel

On peut résumer la modélisation du neurone formel à partir du neurone biologique à l'aide du tableau suivant.

Neurone biologique	Neurone artificiel
Axones	Signal de sortie
Dendrites	Signal d'entrée
Synapses	Poids de la connexion

Tab1.Correspondance neurone biologique/formel

2.NEURONE ARTIFICIEL

Un réseau de neurones est un outil puissant de la modélisation des relations complexes entre les données d'entrée et de sortie. Le développement de cette technologie a été motivé par l'ambition d'avoir un système artificiel capable d'accomplir des tâches "intelligentes" de manière semblable à celles qui sont exécutées par le cerveau humain. Un réseau de neurones ressemble au cerveau humain dans le sens où il apprend des connaissances par apprentissage, et ses connaissances sont stockées dans des connexions inter-neurones connues sous le nom « poids synaptiques ».
L'avantage des réseaux de neurones se situe dans leur capacité d'apprendre à résoudre des problèmes complexes à partir la modélisation des exemples d'apprentissage réels.

Selon Mc Culloch et Pitts :

un Modèle mathématique très simple ,dérivé de la réalité biologique.

Un Neurone: est un automate composé de:

1 Un ensemble d'entrées, x_i

2 Un ensemble de poids, ω_i

3 Un seuil, s

4 Une fonction d'activation (Une fonction de transfert),f pour chaque noeud qui détermine l'état d'un noeud en fonction des poids de ses liens entrants, son biais et les états des noeuds reliés à ce noeud. Cette fonction est habituellement une fonction sigmoïde

5. Un biais associé à chaque noeud.

6. Une sortie, y

Un réseau de neurones possède les avantages suivants :
Un apprentissage adaptatif : qui est sa capacité d'apprendre comment résoudre de nouvelles tâches basée sur l'expérience initiale ou les données d'apprentissage.
l'auto-Organisation : qui est la capacité d'un réseau de neurones à créer sa propre architecture ou sa propre représentation d'information qu'il reçoit au cours de la phase d'apprentissage.
exécution en temps réel : les calculs d'un réseau de neurones se font en temps réel comme ils peuvent être effectués en parallèle à l'aide des dispositifs particuliers

Un neurone artificiel est une unité élémentaire qui reçoit un nombre d'entrées ou des sorties en provenance des autres neurones du réseau, un poids est associé à chacune de ces entrées qui représente la puissance de sa connexion avec un neurone.
La figure 2 montre le modèle du neurone artificiel.

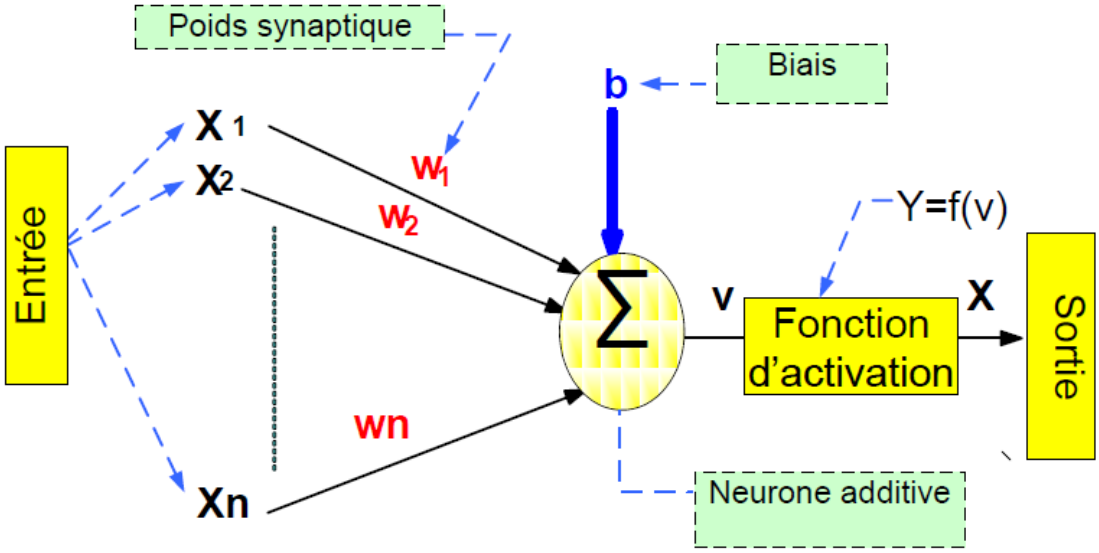


Fig. 2 Le neurone artificiel

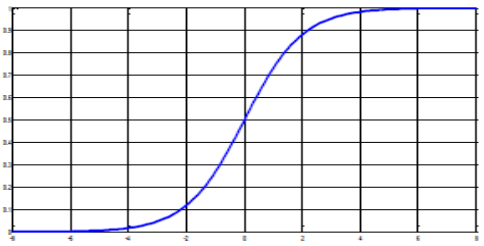
La sortie du neurone est une somme pondérée, de ses entrées plus un biais, propagée par une fonction d'activation qui peut être (sigmoïde, tangente ou hyperbolique, etc.) [11][16] [17]

$$y = \sum_{j=1}^{n} w_j \; x_j + b$$
$$x = f(y)$$

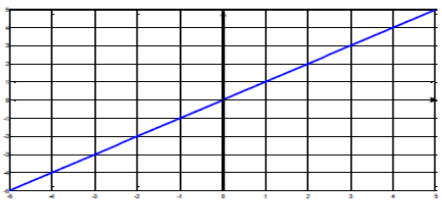
: Est la valeur de la jeme entrée du neurone, Est le poids synaptique correspondant au neurone j, désigne le biais et la fonction d'activation. La fonction la plus utilisée est du type sigmoïde. Elle est définie par :

$$f(y) = \frac{1}{1+e^{-\delta y}}$$

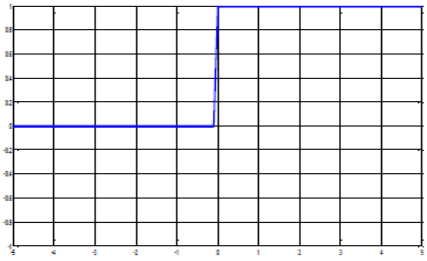
Où δ dénote le paramètre de la sigmoïde qui définit le degré de non-linéarité.
L'utilisation d'une fonction d'activation non-linéaire permet au RNA de modéliser des équations dont la sortie n'est pas une combinaison linéaire des entrées.
Cette caractéristique confère au RNA de grandes capacités de modélisation fortement appréciées pour la résolution de problèmes non-linéaires. Voici quelques exemples de fonctions d'activation :



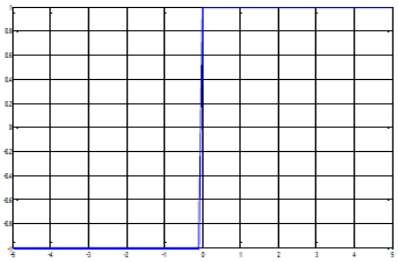
Fonction sigmoïde: $f(x) = \frac{1}{1+e^{-x}}$



Fonction linéair: $f(x) = x$



$$Fonction\ seuil: f(x) = \begin{cases} 1\ Si\ x \geq 0 \\ 0\ Si\ x < 0 \end{cases}$$



$$Fonction\ seuil\ symétrique: f(x) = \begin{cases} 1\ Si\ x \geq 0 \\ 0\ Si\ x < 0 \end{cases}$$

a/Calcul de la sortie

Si a > s (seuil) $\Rightarrow$ x = 1
sinon x = 0, -1 ou autre valeur $\neq$ 1 selon l'application .

Ou bien :

$a = \Sigma\ (\omega_i.e_i) - s$
(la valeur de seuil est introduite ici dans le calcul de la somme pondérée)

x = signe(a) (si a > 0 alors x = +1 sinon x = -1)

3.TOPOLOGIES

Un réseau de neurones résulte de l'arrangement du modèle de base du neurone individuel dans diverses configurations. Plusieurs architectures sont donc envisageables.
On doit cependant réaliser que les algorithmes utilisés pour entraîner les RNA ont une adéquation très forte avec une architecture donnée, c'est-à-dire qu'ils leurs sont intimement liés. Plusieurs topologies et algorithmes d'apprentissage ont été imaginés et conçus.

3.1 LE PERCEPTRON MONO-COUCHE (ROSENBLATT, 1958) :

Développé par Rosenblatt en 1958, c'est le premier réseau à avoir vu le jour. De type feedforward, il ne contient qu'une couche de neurones d'entrée et une couche de sortie. Tous les neurones de la couche d'entrée sont reliés à ceux de la couche de sortie. Ils ne manipulent que des valeurs binaires et la fonction d'activation consiste en un simple seuil.

Il est capable de simuler des opérations logiques simples comme les portes logiques **And** ou **Or** qui sont linéairement séparables. Son apprentissage s'effectue selon la règle de renforcement de Hebb.

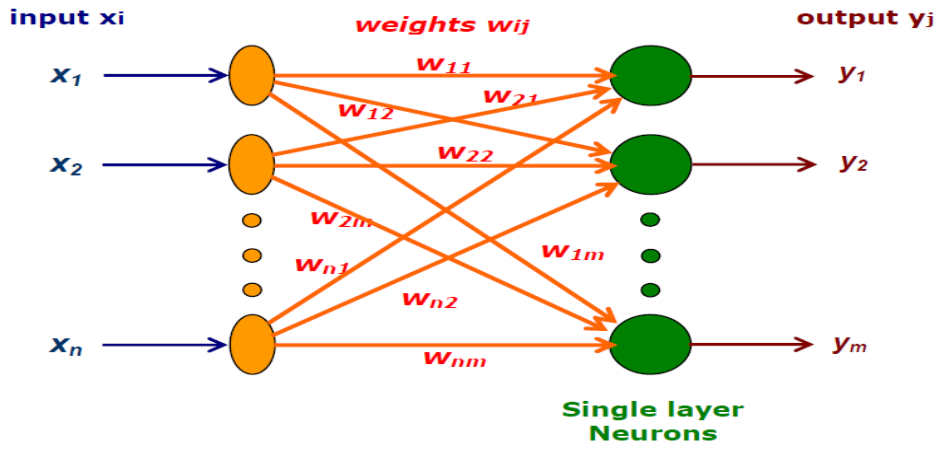


Fig.3 Le perceptron de Rosenblatt

3.2 LE PERCEPTRON MULTI COUCHES (M.MINSKY AND S.PAPERT, 1969)

Un réseau de neurones est un ensemble de processeurs élémentaires, les neurones qui sont largement connectés les uns aux autres et qui sont capables d'échanger des informations au moyen des connexions qui les relient. Les connexions sont directionnelles et à chacune d'elle est associé un réel appelé poids de la connexion.
L'information est ainsi transmise de manière unidirectionnelle du neurone j vers le neurone i, affectée du coefficient pondérateur wij.
Un neurone calcule son état d'informations venues de l'extérieur, ou bien il détermine son entrée à partir des neurones auxquels il est connecté et calcule son état comme une transformation souvent non linéaire de son entrée.
Il transmet à son tour son état vers d'autres neurones ou vers l'environnement extérieur.

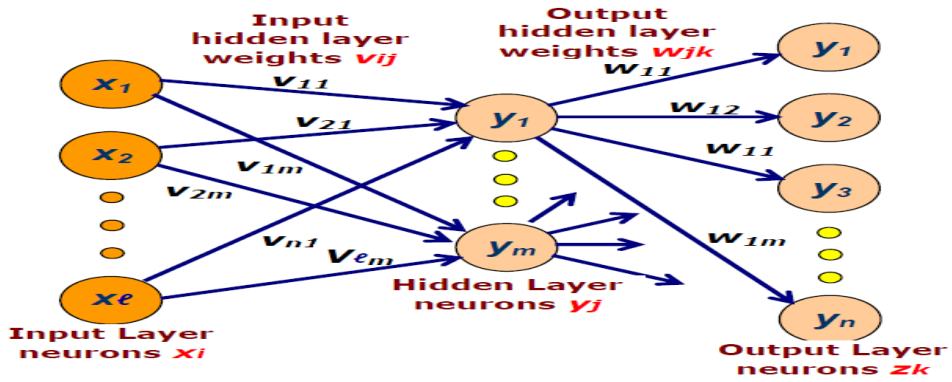


Fig.4 Topologie d'un réseau multicouche (MLP)

3.3 RESEAU A CONNEXIONS RECURRENTES

Les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouches. Ces connexions sont le plus souvent locales (Figure 5).

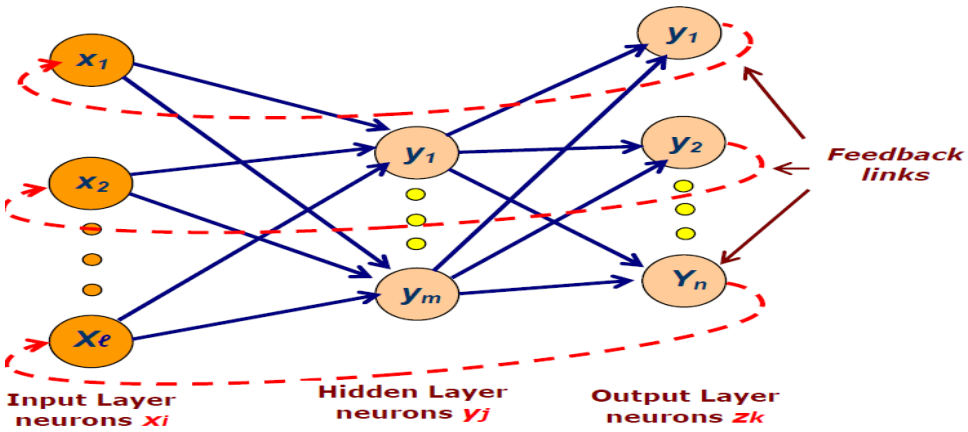


Fig. 5: Réseau à connexions récurrentes

3.4 RESEAU A CONNEXION COMPLETE

C'est la structure d'interconnexion la plus générale (Figure **6**).Chaque neurone est connecté à tous les neurones du réseau (et à lui-même).

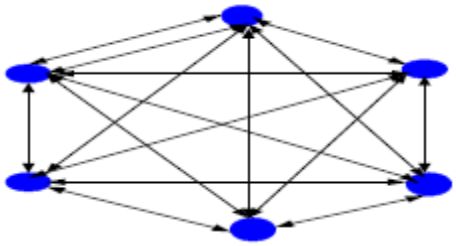


Fig. 6: Réseau à connexions complète

4.APPRENTISSAGE D'UN RESEAU DE NEURONES

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré.
L'ajustement du poids des liens entre les neurones peut s'effectuer selon diverses équations mathématiques, dont la plus populaire est sans aucun doute l'apprentissage hebian (la loi de Hebb).

- le choix de l'équation d'adaptation des poids dépend en grande partie de la topologie du réseau de neurones utilisé.
- Il est aussi possible de faire évoluer l'architecture du réseau, soit le nombre de neurones et les interconnexions, mais avec d'autres types d'algorithmes d'apprentissage.

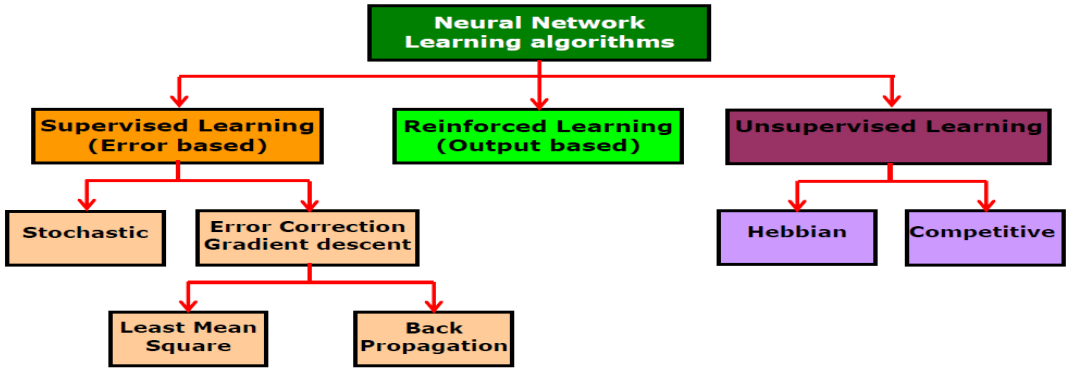
Il existe plusieurs méthodes d'apprentissage dont les plus répandues sont :

1. L'apprentissage supervisé
2. L'apprentissage non supervisé
3. L'apprentissage renfoncé

Ces trois types sont classés selon :

- La présence ou l'absence d'un « enseignant » apprenant
- L'information procurée par le système à étudier.

La figure suivante montre le classement hiérarchique des méthodes d'apprentissage sus citées.



4.1/APPRENTISSAGE SUPERVISE (La rétropropagation) :

Cet algorithme d'apprentissage ne peut être utilisé que lorsque les combinaisons d'entrées-sorties désirés sont connues.

1. L'enseignant (entrée /sortie) est présent durant le processus d'apprentissage et présente les résultats attendus.
2. Chaque modèle d'entrée est utilisé pour entrainer le réseau.
3. Le processus d'apprentissage est basé sur la comparaison entre la sortie obtenue par le réseau et la sortie désirée générant ainsi une erreur.
4. L'erreur est utilisée pour ajuster les paramètres du réseau (les poids), ce qui améliorera les performances.

En résumé : Un apprentissage est dit supervisé lorsqu'on **force** le réseau à **converger** vers un **état final précis**.

Supervised Learning: ⇒ System identification and control

Teacher Input/ Target data

Network weight correction Learning algorithm

Minimize an error function

Mean-squared error (MSE)

4.2 APPRENTISSAGE NON-SUPERVISE :

Il n'y pas de connaissances à priori des sorties désirés pour des entrées données. En fait, c'est de l'apprentissage par exploration où l'algorithme d'apprentissage ajuste les poids des liens entre neurones de façon à maximiser la qualité de classification des entrées. [18] [20] [23].

Les RNA qui utilisent ce type d'apprentissage sont appelés «**auto-organisatrice**» où ce type d'apprentissage possède souvent une **moindre complexité** dans le calcul par rapport à l'apprentissage supervisé

Unsupervised Learning ⟹ Pattern recognition

Only input pattern provided

4.3 APPRENTISSAGE PAR RENFORCEMENT *:*

Dans ce cas, bien que les sorties idéales ne soient pas connues directement, il y a un moyen quelconque de connaître si les sorties du RNA s'approchent ou s'éloignent du but visé. Ainsi, les poids sont ajustés de façons plus ou moins aléatoire et la modification est conservée si l'impact est positif ou rejetée sinon.

5. LE PERCEPTRON MULTICOUCHE (MLP)

Le perceptron multicouche (MLP) est l'exemple le plus répandu des classificateurs neuronaux employés dans le domaine de la reconnaissance. Un MLP est composé d'une couche d'entrée dont la taille est égale à la taille des données d'entrée, une ou plusieurs couches cachées dont la taille est déterminée par essai et la couche de sortie dont la taille est égale au nombre de classes du problème à résoudre.

Dans les réseaux MLP, chaque neurone est relié à un certain nombre d'entrées qui peuvent être les données d'entrée ou les sorties des couches précédentes (Figure **6**).
Chacune de ces entrées est pondérée par un poids synaptique ; le poids total d'un neurone est le produit scalaire entre les entrées et leurs poids correspondant avec addition d'un biais (b) :

$$Le\ Poids\ total = X_1\ W_1 + \cdots \ldots \ldots \ldots \ldots + X_n\ W_n + b$$

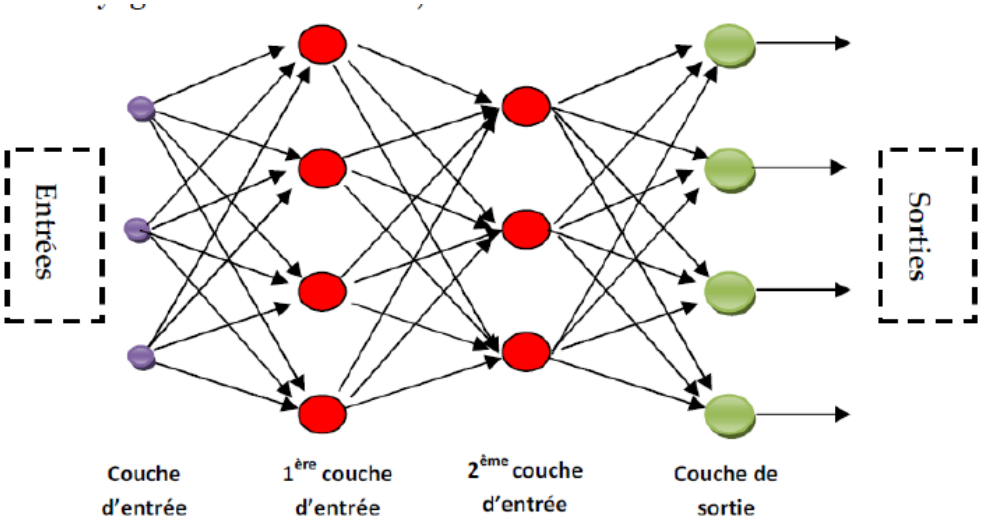


Fig 6: Structure d'un MLP

Un algorithme d'apprentissage tel que la rétropropagation est appliqué pour ajuster les poids en réduisant au minimum une fonction d'erreur qui est, en général, la somme des erreurs quadratiques entre la sortie du réseau de neurones et la sortie attendue.

6. L'algorithme de rétro-propagation

•La création d'un perceptron multicouche pour résoudre un problème passe par l'inférence de la meilleure application possible telle que définie par un en semble de données d'apprentissage constituées de paires de vecteurs d'entrées et de sorties désirées.

•Cette inférence peut se faire par l'algorithme dit de **rétro-propagation**.

•L'algorithme de rétro-propagation, va réaliser l'apprentissage du réseau en **modifiant** les **poids** des connexions neurone par neurone en commençant par la couche de sortie.

•Le critère d'apprentissage étant la **minimisation** de la racine de **l'erreur** quadratique moyenne.

7. ALGORITHME D'APPRENTISSAGE PAR CORRECTION D'ERREUR

Le problème consiste à trouver les poids d'un perceptron qui classifie correctement un ensemble S d'apprentissage.

On note **S** la **base d'apprentissage** composée de couples **(x, c)** où :

x est le vecteur associé à l'entrée **(x_0, x_1, …, x_n)**

c est la sortie correspondante souhaitée

On cherche à déterminer les coefficients **(w_0, w_1, …, w_n)**

Initialiser aléatoirement les coefficients wi

Répéter

Calculer la sortie o du réseau pour l'entrée **x**

Mettre à jour les poids :

Pour i de 0 à n faire

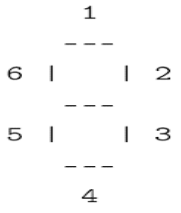
$w_i = w_i + \varepsilon * (c - o) * xi$

Fin Pour

Fin Répéter

Exemple

On veut apprendre si un chiffre est pair ou impair. Le chiffre est représenté par une rétine de 7 leds.



On considère un ensemble complet

$$S = \left\{ \begin{array}{l} (1111110, 0), (0110000, 1), (1101101, 0), \\ (1111001, 1), (0110011, 0), (1011011, 1), \\ (0011111, 0), (1110000, 1), (1111111, 0), (1111011, 1) \end{array} \right\}$$

Trace de l'algorithme

Etape	w	x	c	o
1	$(1,1,1,1,1,1,1,1)$	$(1,1,1,1,1,1,1,0)$	0	1
2	$(0,0,0,0,0,0,0,1)$	$(1,0,1,1,0,0,0,0)$	1	0
3	$(1,0,1,1,0,0,0,1)$	$(1,1,1,0,1,1,0,1)$	0	1
4	$(0,-1,0,1,-1,-1,0,0)$	$(1,1,1,1,1,0,0,1)$	1	0
5	$(1,0,1,2,0,-1,0,1)$	$(1,0,1,1,0,0,1,1)$	0	1
6	$(0,0,0,1,0,-1,-1,0)$	$(1,1,0,1,1,0,1,1)$	1	0
7	$(1,1,0,2,1,-1,0,1)$	$(1,0,0,1,1,1,1,1)$	0	1
8	$(0,1,0,1,0,-2,-1,0)$	$(1,1,1,1,0,0,0,0)$	1	1
9	$(0,1,0,1,0,-2,-1,0)$	$(1,1,1,1,1,1,1,1)$	0	0
10	$(0,1,0,1,0,-2,-1,0)$	$(1,1,1,1,1,0,1,1)$	1	1

Vérification

$$\text{Coefficients} = (0,1,0,1,0,-2,-1,0)$$

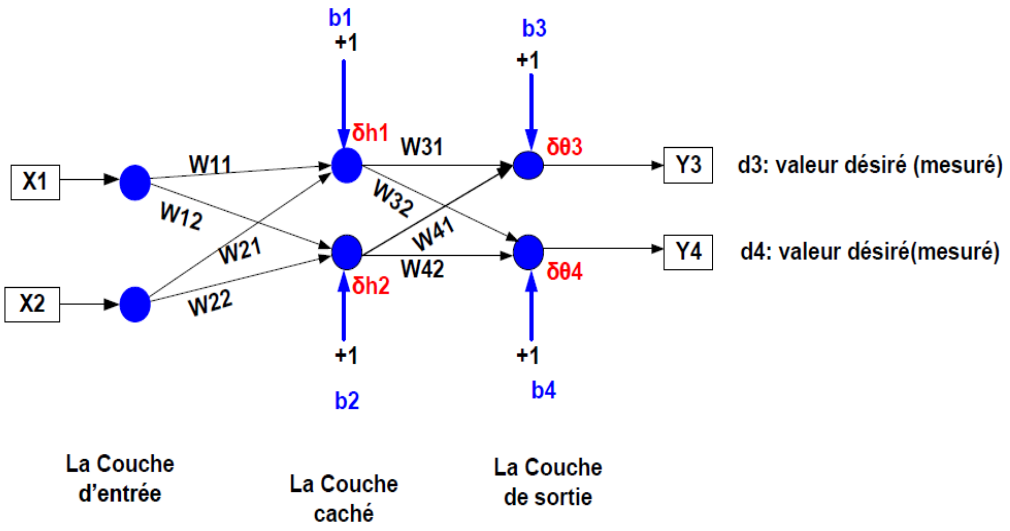
Entrée	Sortie
$(1,1,1,1,1,1,1,0)$	0
$(1,0,1,1,0,0,0,0)$	1
$(1,1,1,0,1,1,0,1)$	0
$(1,1,1,1,1,0,0,1)$	1
$(1,0,1,1,0,0,1,1)$	0
$(1,1,0,1,1,0,1,1)$	1
$(1,0,0,1,1,1,1,1)$	0
$(1,1,1,1,0,0,0,0)$	1
$(1,1,1,1,1,1,1,1)$	0
$(1,1,1,1,1,0,1,1)$	1

8. ALGORITHME PAR DESCENTE DE GRADIENT

Cet algorithme évalue l'erreur de gradient pour chaque neurone dans le réseau depuis la dernière couche jusqu'à la première. Les poids produisant une erreur significative seront modifiés en grande valeur par rapport aux poids produisant une erreur marginale. La rétro-propagation par descente de gradient a été efficacement employée pour l'apprentissage d'un MLP ; son principe est de converger de manière itérative vers un ensemble optimal des poids synaptiques. Dans le cas de l'apprentissage supervisé, les sorties sont fournies à l'avance ; la différence entre ces sorties et les sorties du réseau (après propagation de l'information par une fonction d'activation) constitue l'erreur à réduire au minimum par rétro-propagation tout en ajustant les poids. Cette étape est répétée plusieurs fois jusqu'à ce que le réseau puisse offrir la meilleure prévision.
Dans certains cas l'algorithme de la rétro-propagation de gradient ne peut pas échapper des optimums locaux, pour cette raison, un poids moment a été ajouté à la formule de rétro-propagation permettant à l'algorithme d'échapper aux optimums locaux.

L'algorithme de rétro-propagation par descente de gradient suit les étapes suivantes :

- De chaque neurone, l'information est propagée à la couche suivante en utilisant une fonction d'activation (habituellement une fonction sigmoïde) appliquée au produit scalaire des entrées et les poids correspondants :



$$v_1 = 1 * b_1 + X_1 * W_{11} + X_2 * W_{21}$$

$$y_1 = f(v_1) = \frac{1}{1+e^{-\delta v_1}} \text{ (Fonction sigmoïde)}$$

δ: dégrée de non linéarité=1

$$y_1 = f(v_1) = \frac{1}{1+e^{-v_1}}$$

$$v_2 = 1 * b_2 + X_1 * W_{12} + X_2 * W_{22}$$

$$y_2 = f(v_2) = \frac{1}{1+e^{-v_2}}$$

$$v_3 = 1 * b_3 + y_1 * W_{31} + y_2 * W_{41}$$

$$y_3 = f(v_3) = \frac{1}{1+e^{-v_3}}$$

$$v_4 = 1 * b_4 + y_1 * W_{32} + y_2 * W_{42}$$

$$y_4 = f(v_4) = \frac{1}{1+e^{-v_4}}$$

$$e_3 = d_3 - y_3 \text{(Erreur)}$$

$$e_4 = d_4 - y_4 \text{(Erreur)}$$

$$\delta\theta_3 = f'(v_3) * e_3 \text{(Gradient)}$$

$$y_3{}' = f'(v_3) = f(v_3) * \big(1 - f(v_3)\big)$$

$$\delta\theta_3 = \Big(f(v_3) * \big(1 - f(v_3)\big)\Big) * e_3$$

$$\delta\theta_4 = f'(v_4) * e_4 \text{(Gradient)}$$

$$y_4' = f'(v_4) = f(v_4) * (1 - f(v_4))$$
$$\delta\theta_4 = \Big(f(v_4) * \big(1 - f(v_4)\big)\Big) * e_4$$
$$\delta h_{1=} f'(v_1) * (\delta\theta_3 * W_{31} + \delta\theta_4 W_{32}) \text{ (Gradient)}$$
$$\delta h_{1=} (f(v_1) * \big(1 - f(v_1)\big) * (\delta\theta_3 * W_{31} + \delta\theta_4 W_{32})$$
$$\delta h_{2=} f'(v_2) * (\delta\theta_3 * W_{41} + \delta\theta_4 W_{42}) \text{ (Gradient)}$$
$$\delta h_{1=} (f(v_2) * \big(1 - f(v_2)\big) * (\delta\theta_3 * W_{41} + \delta\theta_4 W_{42})$$

8.1 AJUSTEMENT DES POIDS

$$\mathrm{W}(\mathrm{n}+1) = W(n) + \alpha * W(n-1) + \tau * \delta(n) * y$$

α est le moment, c'est un nombre réel positif dont la valeur est comprise entre 0 et 1.
τ: taux d'apprentissage son augmentation améliore les performances du processus d'apprentissage (augmenter la rapidité de convergence).

$$\mathrm{W}_{31}(\mathrm{n}+1) = \mathrm{W}_{31}(n) + \alpha * \mathrm{W}_{31}(n-1) + \tau * \delta(\theta_3) * y_1$$
$$\mathrm{W}_{32}(\mathrm{n}+1) = \mathrm{W}_{32}(n) + \alpha * \mathrm{W}_{32}(n-1) + \tau * \delta(\theta_4) * y_1$$
$$\mathrm{W}_{41}(\mathrm{n}+1) = \mathrm{W}_{41}(n) + \alpha * \mathrm{W}_{41}(n-1) + \tau * \delta(\theta_3) * y_2$$
$$\mathrm{W}_{42}(\mathrm{n}+1) = \mathrm{W}_{42}(n) + \alpha * \mathrm{W}_{42}(n-1) + \tau * \delta(\theta_4) * y_2$$
$$\mathrm{W}_{11}(\mathrm{n}+1) = \mathrm{W}_{11}(n) + \alpha * \mathrm{W}_{11}(n-1) + \tau * \delta(h_1) * x_1$$
$$\mathrm{W}_{12}(\mathrm{n}+1) = \mathrm{W}_{12}(n) + \alpha * \mathrm{W}_{12}(n-1) + \tau * \delta(h_2) * x_1$$
$$\mathrm{W}_{21}(\mathrm{n}+1) = \mathrm{W}_{42}(n) + \alpha * \mathrm{W}_{42}(n-1) + \tau * \delta(h_1) * x_2$$
$$\mathrm{W}_{22}(\mathrm{n}+1) = \mathrm{W}_{22}(n) + \alpha * \mathrm{W}_{22}(n-1) + \tau * \delta(h_2) * x_2$$

8.2. AJUSTEMENT LE SEUIL D'ACTIVITATION (BIAIS)

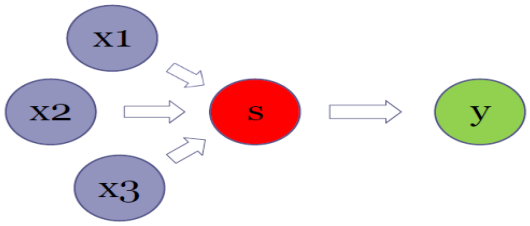
$$\mathrm{b}(\mathrm{n}+1) = b(n) + \alpha * b(n-1) + \tau * \delta(n) * 1$$
$$\mathrm{b}_1(\mathrm{n}+1) = \mathrm{b}_1(n) + \alpha * \mathrm{b}_1(n-1) + \tau * \delta(h_1) * 1$$
$$\mathrm{b}_2(\mathrm{n}+1) = \mathrm{b}_2(n) + \alpha * \mathrm{b}_2(n-1) + \tau * \delta(h_2) * 1$$
$$\mathrm{b}_3(\mathrm{n}+1) = \mathrm{b}_3(n) + \alpha * \mathrm{b}_3(n-1) + \tau * \delta(\theta_3) * 1$$
$$\mathrm{b}_4(\mathrm{n}+1) = \mathrm{b}_4(n) + \alpha * \mathrm{b}_4(n-1) + \tau * \delta(\theta_4) * 1$$

9. EXEMPLES D'APPLICATION

9.1 EXEMPLE1

Soit le neurone artificiel de la figure ci-dessous avec:
•Une fonction de combinaison : **produit scalaire**.

•Une fonction d'activation: **linéaire**.



Calculez la réponse **y** à l'excitation **x=[0 1 5]** sachant que les poids **w=[1 -2 6].**
Solution:

$\psi(s)=s$

$d\psi/ds=1$

x=[0 1 5]
w=[1 -2 6]'
s=x*w
s=28
y=s=28

9.2 EXEMPLE 2

Programmez un neurone artificiel à 3 entrées et 1 sortie, avec:
•Une fonction de combinaison : **distance euclidienne**.

•Une fonction d'activation: **radiale de base,** avec γ=1

En utilisant Matlab, calculez la réponse **y** à l'excitation
x= (0 1 5), x= (-2 6.25 -5) et les poids **w= (1 -2 6).**

La fonction radiale de base est donnée par :

$$\psi(s) = \exp(-s^2/\gamma^2)$$

$$d\psi/ds = -(2/\gamma^2)\cdot s\cdot \exp(-s^2/\gamma^2) = -(2/\gamma^2)\cdot s\cdot \psi(s)$$

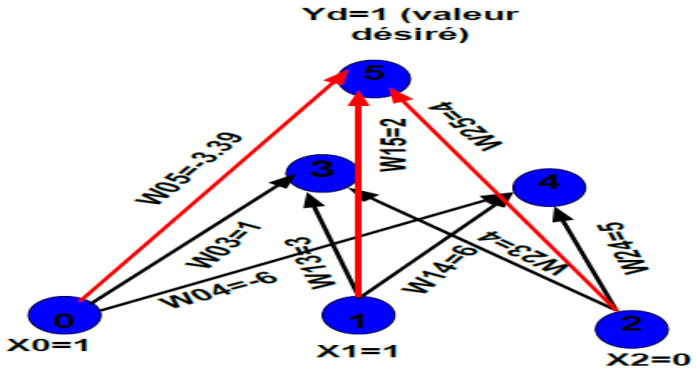
avec $\gamma \in \Re^n$.

Solution :
x=[0 1 5]
w=[1 -2 6]
s=abs(x-w)
s=sqrt(sum(s.*s))
s =3.3166
y=exp(-s*s)
y =1.6702e-005

9.3 EXEMPLE 3

Voici le réseau MLP constitué d'une couche d'entrée à trois neurones une couche cachée ayant deux neurones, et la couche de sortie constituée d'un seul neurone comme montré par la figure ci-dessous:
Avec δ=1 (dégrée de non linéarité)

Calculez l'erreur.



Solution :
La sortie du neurone 3

$$a_3 = X_0 * W_{03} + X_1 * W_{13} + X_2 * W_{23} = 1 * 1 + 1 * 3 + 0 * 4 = 4$$

La fonction d'activation de la sortie du neurone 3 :

$$y_3 = f(a_3) = \frac{1}{1 + e^{-\delta a_3}} = \frac{1}{1 + e^{-(4)}} = 0.982$$

La sortie du neurone 4

$$a_4 = X_0 * W_{04} + X_1 * W_{14} + X_2 * W_{24} = 1 * -6 + 1 * 6 + 0 * 5 = 0$$

La fonction d'activation de la sortie du neurone 4 :

$$y_4 = f(a_4) = \frac{1}{1 + e^{-\delta a_4}} = \frac{1}{1 + e^{-(0)}} = 0.5$$

La sortie du neurone 5 :

$$a_5 = X_0 * W_{05} + X_1 * W_{15} + X_2 * W_{25} = 1 * -3.39 + 1 * 2 + 0 * 4 = 0.04$$

La fonction d'activation de la sortie du neurone 5 :

$$y_5 = f(a_5) = \frac{1}{1 + e^{-\delta a_5}} = \frac{1}{1 + e^{-(0)}} = 0.510$$

Donc l'erreur est :

$e = y_d - y_5 = 1 - 0.510 = 0.490$ (Erreur=valeur désiré-valeur de sortie)

9.4 EXEMPLE 4

Voici une perception MLP constitué d'une couche d'entrée portant deux neurones , une couche cachée aura deux neurones, et la couche de sortie constitue un seul neurone comme montré par la figure ci-dessous:

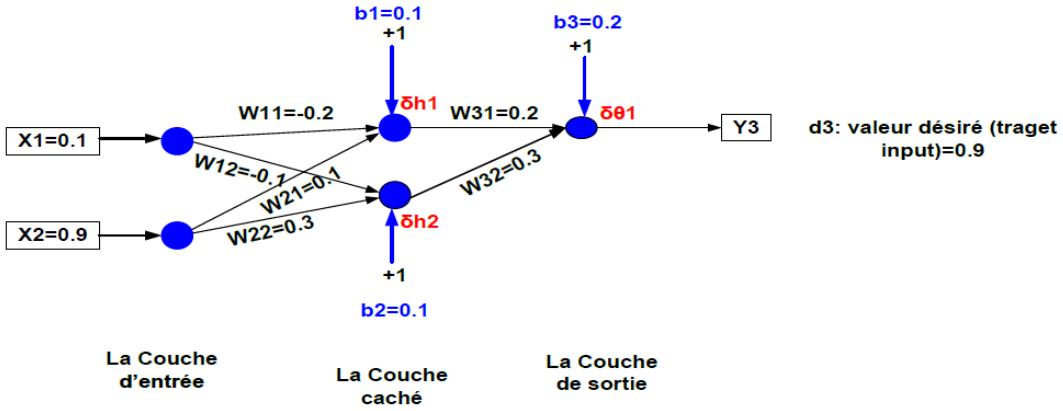
On donne :

δ=1 (degré de non linéarité)

d3=0.9 (valeur désirée)

τ=0.25 (taux d'apprentissage)

α=0.0001



Trouver la valeur minimale de l'erreur après trois itérations ?

Solution

- 1ère itération

$$v_1 = 1 * b_1 + X_1 * W_{11} + X_2 * W_{21}$$

$$v_1 = 1 * 0.1 + 0.1 * (-0.2) + 0.9 * 0.1 = 0.17$$

$$y_1 = f(v_1) = \frac{1}{1+e^{-\delta v_1}} \text{ (Fonction sigmoïde)}$$

$$y_1 = \frac{1}{1 + e^{-(0.17)}} = 0.542$$

$$v_2 = 1 * b_2 + X_1 * W_{12} + X_2 * W_{22} = 1 * 0.1 + 0.1 * (-0.1) + 0.9 * 0.3 = 0.36$$

$$y_2 = f(v_2) = \frac{1}{1 + e^{-\delta v_2}} = \frac{1}{1 + e^{-(0.36)}} = 0.590$$

$$v_3 = 1 * b_3 + y_1 * W_{31} + y_2 * W_{32} = 1 * 0.2 + 0.542 * 0.2 + 0.590 * 0.3 = 0.485$$

$$y_3 = f(v_3) = \frac{1}{1 + e^{-v_3}} = \frac{1}{1 + e^{-(0.485)}} = 0.619$$

$e = d_3 - y_3 = 0.9 - 0.619 = 0.281$ ⇒ **$e = 0.281$**

- Calcule le gradient de chaque neurone :

$$\delta\theta_1 = f'(v_3) * e = \left(f(v_3) * \left(1 - f(v_3)\right)\right) * e$$
$$= \left(0.619 * (1 - 0.619)\right) * 0.281 = 0.0662$$
$$\delta h_1 = f'(v_1) * (\delta\theta_1 * W_{31}) = (f(v_1) * \left(1 - f(v_1)\right) * (\delta\theta_1 * W_{31})$$
$$= \left(0.542 * (1 - 0.542)\right) * (0.0663 * 0.2)$$
$$= 0.0033$$
$$\delta h_2 = f'(v_2) * (\delta\theta_1 * W_{32}) = (f(v_2) * \left(1 - f(v_2)\right) * (\delta\theta_1 * W_{32})$$
$$= \left(0.590 * (1 - 0.590)\right) * (0.0663 * 0.3) = 0.00492$$

- **AJUSTE LE POIDS**

$$W_{31}(n+1) = W_{31}(n) + \alpha * W_{31}(n-1) + \tau * \delta\theta_1 * y_1$$
$$0.2 + 0.0001 * 0.2 + 0.25 * 0.0662 * 0.542 = 0.2090$$

$W_{31}(n+1) = 0.2090$

$W_{32}(n+1) = W_{32}(n) + \alpha * W_{32}(n-1) + \tau * \delta\theta_1 * y_2 = 0.3 + 0.0001 * 0.3 + 0.25 * 0.0663 * 0.589 = 0.3098$

$$\mathbf{W_{32}(n+1) = 0.3098}$$

$W_{11}(n+1) = W_{11}(n) + \alpha * W_{11}(n-1) + \tau * \delta h_1 * X_1$
$= (-0.2) + 0.0001 * (-0.2) + 0.25 * 0.0033 * 0.1 = -0.1999$

$$\mathbf{W_{11}(n+1) = -0.1999}$$

$W_{12}(n+1) = W_{12}(n) + \alpha * W_{12}(n-1) + \tau * \delta h_2 * X_1$
$= (-0.1) + 0.0001 * (-0.1) + 0.25 * 0.00492 * 0.1 = -0.0998$

$$\mathbf{W_{12}(n+1) = -0.0998}$$

$W_{21}(n+1) = W_{21}(n) + \alpha * W_{21}(n-1) + \tau * \delta h_1 * X_2$
$= 0.1 + 0.0001 * 0.1 + 0.25 * 0.0033 * 0.9 = 0.108$

$$\mathbf{W_{21}(n+1) = 0.108}$$

$W_{22}(n+1) = W_{22}(n) + \alpha * W_{22}(n-1) + \tau * \delta h_2 * X_2 = 0.3 + 0.0001 * 0.3 + 0.25 * 0.0049 * 0.9 = 0.3011$

$$\mathbf{W_{22}(n+1) = 0.3011}$$

AJUSTE LE SEUIL D'ACTIVITATION

$b_3(n+1) = b_3(n) + \alpha * b_3(n-1) + \tau * \delta(\theta_1) * 1$
$= 0.2 + 0.0001 * 0.2 + 0.25 * 0.0662 * 1 = 0.2016$

$$\mathbf{b_3(n+1) = 0.2066}$$

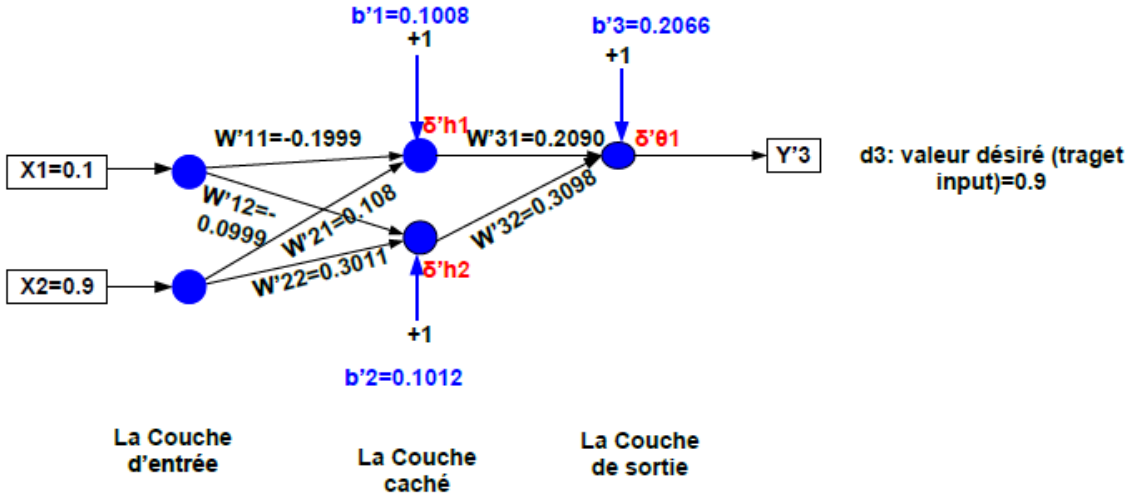
$b_2(n+1) = b_2(n) + \alpha * b_2(n-1) + \tau * \delta(h_2) * 1$
$= 0.1 + 0.0001 * 0.1 + 0.25 * 0.00492 * 1 = 0.1012$

$$\mathbf{b_2(n+1) = 0.1012}$$

$b_1(n+1) = b_1(n) + \alpha * b_1(n-1) + \tau * \delta(h_1) * 1 = 0.1 + 0.0001 * 0.1 + 0.25 * 0.0033 * 1 = 0.1008$

$$\mathbf{b_1(n+1) = 0.1008}$$

Après la première itération le réseau devient avec les nouveaux poids et seuil d'activation on passe à la deuxième itération.



$$v_1' = 1 * b_1' + X_1 * W_{11}' + X_2 * W_{21}'$$

$$v_1' = 1 * 0.1008 + 0.1 * 0.1999 + 0.9 * 0.108 = 0.18521$$

$$y_1' = f(v_1') = \frac{1}{1+e^{-\delta v_1'}} \text{ (Fonction sigmoïde)}$$

$$y_1' = f(v_1') = \frac{1}{1+e^{-v_1'}} = \frac{1}{1+e^{-0.18521}} = 0.5461$$

$$y_1' = 0.5461$$

$$v_2' = 1 * b_2' + X_1 * W_{12}' + X_2 * W_{22}'$$

$$v_2' = 1 * 0.1012 + 0.1 * 0.0999 + 0.9 * 0.3011 = 0.3622$$

$$y_2' = f(v_2') = \frac{1}{1+e^{-\delta v_2'}} = \frac{1}{1+e^{-0.3622}} = 0.5895$$

$$y_2' = 0.5895$$

$$v'_3 = 1 * b'_3 + y'_1 * W'_{31} + y'_2 * W'_{32}$$

$$v'_3 = 1 * 0.2066 + 0.5461 * 0.2090 + 0.5895 * 0.3098 = 0.5033$$

$$y'_3 = f(v'_3) = \frac{1}{1 + e^{-v'_3}} = \frac{1}{1 + e^{-0.5033}} = 0.6232$$

$$y'_3 = 0.6232$$

$$e' = d_3 - y'_3 = 0.9 - 0.6255 = 0.2745 \Rightarrow \boxed{e' = 0.2745}$$

Calcul du gradient pour chaque neurone :

$$\delta'\theta_1 = f'(v'_3) * e' = \left(f(v'_3) * \left(1 - f(v'_3)\right)\right) * e'$$
$$= (0.6255 - 0.3745) * 0.276 = 0.0692$$
$$\delta'h_1 = f'(v'_1) * (\delta'\theta_1 * W'_{31}) = (f(v'_1) * \left(1 - f(v'_1)\right) * (\delta'\theta_1 * W'_{31})$$
$$= (0.5461 - 0.4539) * (0.0692 * 0.2090) = 0.0922 * 0.01440$$
$$= 0.00133$$
$$\delta'h_2 = f'(v'_2) * (\delta'\theta_1 * W'_{32}) = (f(v'_2) * \left(1 - f(v'_2)\right) * (\delta'\theta_1 * W'_{32})$$
$$= (0.5895 - 0.4105) * (0.0692 * 0.3098) = 0.179 * 0.02134$$
$$= 0.00383$$

Ajustement des poids :

$$W'_{31}(n+1) = W'_{31}(n) + \alpha * W'_{31}(n-1) + \tau * \delta\theta'_1 * y'_1$$
$$= 0.2090 + 0.0001 * 0.2090 + 0.25 * 0.0692 * 0.5461 = 0.2184$$

$$\boxed{W'_{31}(n+1) = 0.2184}$$

$$W'_{32}(n+1) = W'_{32}(n) + \alpha * W'_{32}(n-1) + \tau * \delta'\theta_1 * y'_2 = 0.3098 + 0.0001 * 0.3098 + 0.25 * 0.0692 * 0.5895 = 0.3200$$

$$\boxed{W'_{32}(n+1) = 0.3200}$$

$$W'_{11}(n+1) = W'_{11}(n) + \alpha * W'_{11}(n-1) + \tau * \delta'h_1 * X_1 = (-0.1999) + 0.0001 * (-0.1999) + 0.25 * 0.00133 * 0.1 = -0.1999$$

$$\boxed{W'_{11}(n+1) = -0.1999}$$

$$W'_{12}(n+1) = W'_{12}(n) + \alpha * W'_{12}(n-1) + \tau * \delta' h_2 * X_1$$
$$= (-0.0999) + 0.0001 * (-0.0999) + 0.25 * 0.00383 * 0.1$$
$$= -0.0998$$

$$\mathbf{W'_{12}(n+1) = -0.0998}$$

$$W'_{21}(n+1) = W'_{21}(n) + \alpha * W'_{21}(n-1) + \tau * \delta' h_1 * X_2$$
$$= 0.108 + 0.0001 * 0.108 + 0.25 * 0.00132 * 0.9 = \mathbf{0.1083}$$

$$\mathbf{W'_{21}(n+1) = 0.1083}$$

$$W'_{22}(n+1) = W'_{22}(n) + \alpha * W'_{22}(n-1) + \tau * \delta' h_2 * X_2 = 0.3011 + 0.0001 * 0.3011 + 0.25 * 0.00381 * 0.9 = 0.3019$$

$$\mathbf{W'_{22}(n+1) = 0.3019}$$

Ajustement des seuils d'activation

$$b'_3(n+1) = b'_3(n) + \alpha * b'_3(n-1) + \tau * \delta'(\theta_1) * 1$$
$$= 0.2166 + 0.0001 * 0.2166 + 0.25 * 0.0692 * 1 = 0.234$$

$$\mathbf{b'_3(n+1) = 0.234}$$

$$b'_2(n+1) = b'_2(n) + \alpha * b'_2(n-1) + \tau * \delta'(h_2) * 1$$
$$= 0.1012 + 0.0001 * 0.1012 + 0.25 * 0.00381 * 1 = 0.1021$$

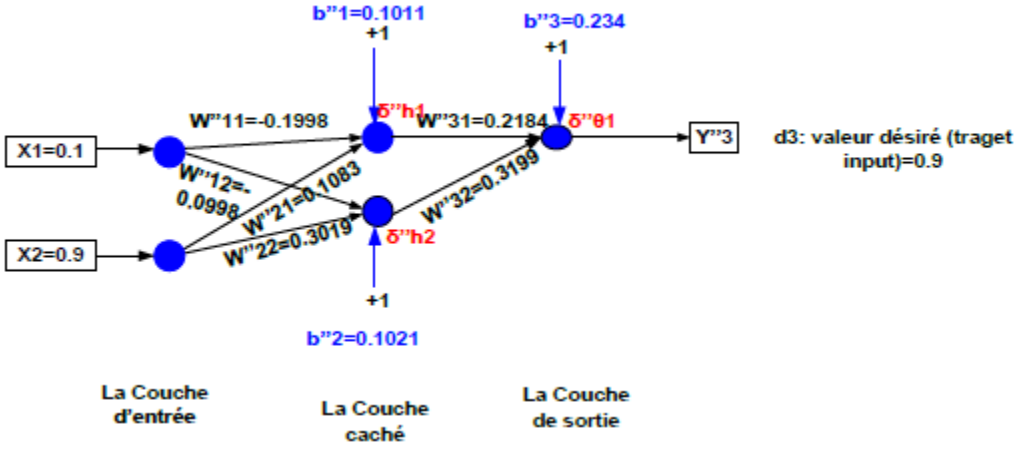
$$\mathbf{b'_2(n+1) = 0.1021}$$

$$b'_1(n+1) = b'_1(n) + \alpha * b'_1(n-1) + \tau * \delta'(h_1) * 1$$
$$= 0.1008 + 0.0001 * 0.1008 + 0.25 * 0.00133 * 1 = 0.1011$$

$$\mathbf{b'_1(n+1) = 0.1011}$$

Troisième itération

Arriver à cette phase on peut passer à la 3ème itération pour calculer l'erreur, on doit alors porter les nouveaux poids et biais ajustés lors de la 2ème itération dans la structure du MLP.



$$v_1'' = 1 * b_1'' + X_1 * W_{11}'' + X_2 * W_{21}''$$

$$v_1'' = 1 * 0.1011 + 0.1 * (-0.1998) + 0.9 * 0.1083 = 0.1785$$

$$y_1'' = f(v_1'') = \frac{1}{1 + e^{-v_1''}} = \frac{1}{1 + e^{-0.1785}} = 0.5445$$

$$v_2'' = 1 * b_2'' + X_1 * W_{12}'' + X_2 * W_{22}''$$

$$v_2'' = 1 * 0.1021 + 0.1 * (-0.0998) + 0.9 * 0.3019 = 0.3638$$

$$y_2'' = f(v'_2) = \frac{1}{1 + e^{-v_2''}} = \frac{1}{1 + e^{-0.3638}} = 0.5899$$

$$v_3'' = 1 * b_3'' + y_1'' * W_{31}'' + y_2'' * W_{32}''$$

$$v_3'' = 1 * 0.234 + 0.5445 * 0.2184 + 0.5899 * 0.3199 = 0.5416$$

$$y_3'' = f(v_3'') = \frac{1}{1+e^{-v_3''}} = \frac{1}{1+e^{-0.5416}} = 0.6321$$

$$e'' = d_3 - y''_3 = 0.9 - 0.6321 = 0.2679$$

$$e'' = 0.2679$$

Conclusion

Pour pouvoir dire que le réseau converge il faut :

Si l'erreur est jugée suffisamment **faible**, on considère que **l'apprentissage** est **terminé** dans 1itération.
Sinon, on doit faire la **Retro-propagation (nouvelle itération),**
On doit calculer le **signal d'erreur sur chaque neurone** de la couche de **sortie.**