

Chapitre 3

Notions sur les réseaux de neurones artificiels

1. Introduction :

Depuis l'apparition de l'informatique, les chercheurs ont tenté de reproduire un mécanisme qui est propre à l'homme, le raisonnement et au sens le plus large du terme, le pensé. Ainsi, est née l'intelligence artificielle, dont les tendances de recherche se sont orientées rapidement vers l'approche symbolique. Cette représentation des connaissances est une formalisation du raisonnement, sous forme de règles, inspirée de la logique mathématique [8].

L'un des plus grands succès de l'intelligence artificielle est sans doute le système expert avec lequel on a cru un moment pouvoir "*faire penser*" un ordinateur en lui fournissant des connaissances en vrac et quelques algorithmes pour les manipuler. Seulement, et même si ces méthodes s'avèrent prolifiques dans certaines applications, ce n'est décidément pas comme ça que le cerveau fonctionne. Portant, il est légitime de penser que tout être est un ensemble de circuits câblés, mais d'une complexité, des milliards de fois, plus importante que celle des ordinateurs les plus complexes que l'être humain peut envisager.

D'où est venue l'idée d'envisager d'imiter les réseaux de neurones, souvent simulés sur ordinateur. Cependant, un réseau de neurones fonctionne sans programme, n'exécute pas d'instructions, ne contient pas de mémoire pour y stocker des données, ne manipule pas de nombres, de plus la destruction d'une partie de ses circuits ne l'empêche pas de fonctionner [9].

En fait, un réseau de neurones sert à donner des résultats qui sont difficilement calculables, comme par exemple donner un nom à une image d'un visage. Les réseaux neuronaux sont capables de le faire, car ils sont capables du raisonnement, de l'apprentissage voir même de la création.

En ce sens, il faut prendre cette technologie comme complémentaire à celle des ordinateurs qui restent, encore, les meilleurs outils pour traiter de grands ensembles de données ou pour faire des calculs.

Un réseau de neurones ou réseau neuromimétique est, en fait, un système de calcul basé sur un modèle biologique prouve que l'homme cherche sans cesse la possibilité de construire une machine intelligente [9].

2. Historique

Les réseaux de neurones artificiels ont été construits au départ sur le modèle du cerveau humain. Les premiers travaux ont été écrits en 1943 où W.Mcculloch et Pitts introduisent les réseaux de neurones et donnent ainsi naissance au connexionnisme. Puis, en 1957, Rasemblatt présente le premier modèle opérationnel le réseaux de neurones, le perceptron inspiré du système visuel et capable d'apprendre à calculer certaines fonctions logiques en modifiant ses connexions synaptiques. Déjà, en 1949 Hebb parle de cette propriété de plasticité synoptique en proposant une explication à l'apprentissage [8].

Cependant, dans les années 60, les réseaux de neurones tombent dans l'oubli par un désintéressement des chercheurs et les investisseurs. Ce n'est que depuis les années 80 que l'on assiste au renouveau des réseaux de neurones grâce à des chercheurs comme Hophield qui en 1982 a montré l'analogie des réseaux de neurones avec certains systèmes physiques et a permis de leur appliquer un formalisme mieux maîtrisé. Depuis, les recherches et les applications ont progressé [8].

1943	Mcculloch et pitts	Définition du neurone formel
1949	Hebb	Règle d'apprentissage pour les neurones
1957	Rosemblatt	Modèle de réseaux de neurones : le perceptron
1969	Minsky et Papert	Optimisation sur les perceptrons
1982	Hophield	Modèle en relation avec les systèmes physiques
1989	Kohenen	Auto-organisation et mémoires associatives
1992	Boughanem	Réseaux de neurones appliqués à la recherche d'information

Tableau 3.1. Historique des réseaux de neurones

3. Réseaux de neurones artificiels

3.1. Du neurone biologique au neurone formel

La reconnaissance du fait que le cerveau fonctionne de manière entièrement différente de celle d'un ordinateur conventionnel a joué un rôle très important dans le développement des réseaux de neurones artificiels. Les travaux effectués pour essayer de comprendre le comportement du cerveau humain ont mené à représenter celui-ci par un ensemble de composants structurels appelés neurones, massivement interconnectés entre eux. Le cerveau

humain en contiendrait plusieurs centaines de milliards, et chacun de ceux ci serait, en moyenne, connecté à dix mille autres [10].

Le neurone biologique est composé de quatre parties distinctes (figure 3.1).

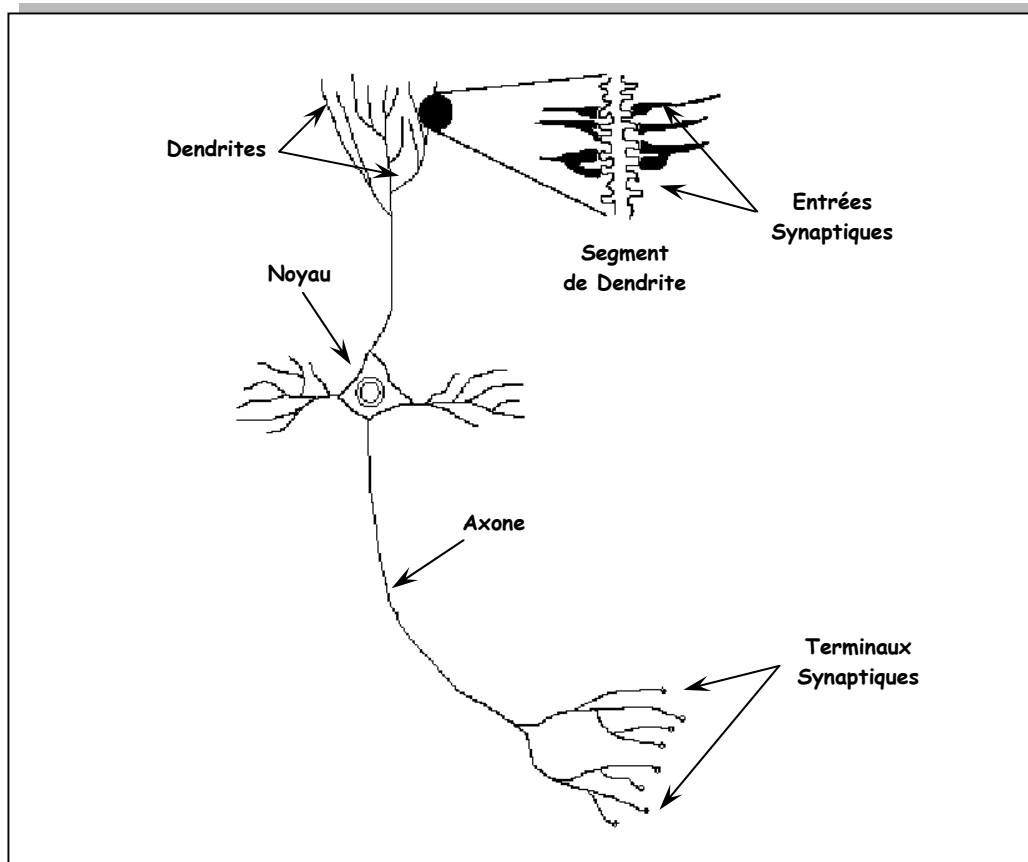


Figure 3.1. Schéma d'un neurone biologique

Le corps cellulaire : qui contient le noyau de la cellule nerveuse, est l'endroit où prend naissance l'influx, qui représente l'état d'activité du neurone.

Les dendrites : sont des ramifications tubulaires courtes formant une espèce d'arborescence autour du corps cellulaire. Elles sont les entrées principales du neurone, qui captent l'information venant d'autres neurones.

L'axone : est une longue fibre nerveuse qui se ramifie à son extrémité. C'est la sortie du neurone et le support de l'information vers les autres neurones.

La synapse : communique l'information, en la pondérant par un poids synaptique, à un autre neurone. Elle est essentielle dans le fonctionnement du système nerveux [10].

L'information contenue dans le cerveau, quant à elle, est représentée par les poids synaptique attribués aux entrées de chaque neurone. Le cerveau est capable d'organiser ces neurones, selon un assemblage complexe, non linéaire et extrêmement parallèle de manière à pouvoir accomplir des tâches très élaborées. Du fait du grand nombre de neurones et de leurs

interconnexions, ce système possède une propriété de tolérance aux fautes. Ainsi, la défectuosité d'un élément mémoire (neurone) n'entraînera aucune perte réelle d'information,

Mais seulement une faible dégradation, en qualité, de toute l'information contenue dans le système. C'est pourquoi, nous pouvons reconnaître le visage d'une personne même si celle-ci est vieille, par exemple, alors que cette tâche est quasiment impossible pour un ordinateur classique [10]. C'est la tentative de donner à l'ordinateur les qualités du perceptron du cerveau humain qui a conduit à une modélisation électrique de celui-ci. C'est cette modélisation que tentent de réaliser les réseaux de neurones artificiels, dont l'élaboration repose sur la base de la définition suivante, proposé par hykin [11].

Un réseau de neurones est un processeur distribué de manière massivement parallèle, qui a une propension naturelle à mémoriser des connaissances de façon expérimentale et de les rendre disponibles pour utilisation. Il ressemble au cerveau en deux points :

1. la connaissance est acquise au travers d'un processeur d'apprentissage.
2. les poids de connections entre les neurones sont utilisés pour mémoriser la connaissance.

La première étude systématique du neurone artificiel est due au neuropsychiatre McCulloch et au logiciel Pitts [12] qui, s'inspirant de leurs travaux sur les neurones biologiques, proposèrent en 1943 le modèle illustré à la figure III.2 [10]. Ce neurone formel est un processeur élémentaire qui réalise une somme pondérée des signaux qui lui parviennent. La valeur de cette sommation est ensuite comparée à un seuil, et la sortie du neurone est une fonction non linéaire du résultat.

$$U = \sum_j^N w_j x_j - \theta \quad \text{III.1}$$

$$y = f(u) \quad \text{III.2}$$

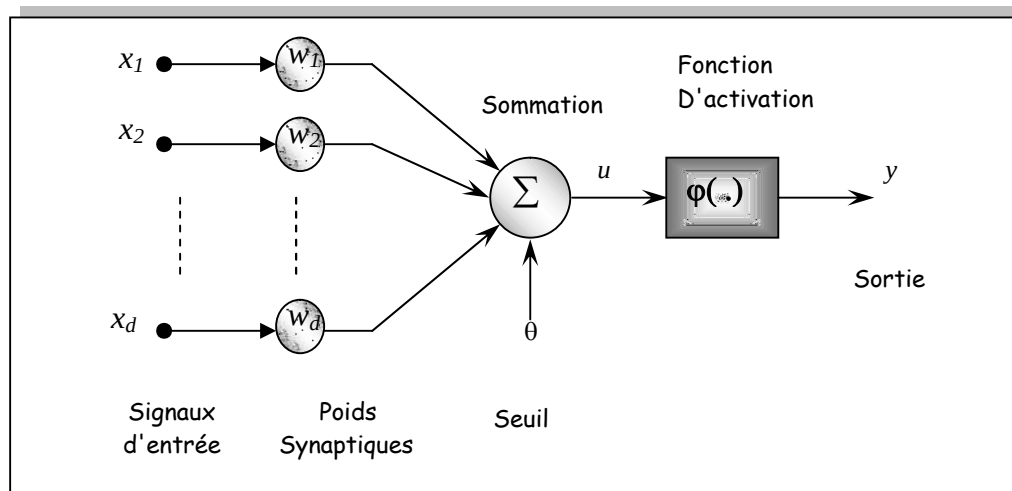


Figure 3.2. Le modèle de neurone formel

En notant par $x = [-1, x_1, \dots, x_j]^T$ le vecteur d'entrée augmenté et par $w = [\theta, w_1, \dots, w_j]^T$ le vecteur des poids augmenté, il vient [Bo00].

$$y = f(w^T, x)$$

III.3

La fonction $\phi(\cdot)$ peut prendre une des formes suivantes:

a) Une fonction binaire à seuil :

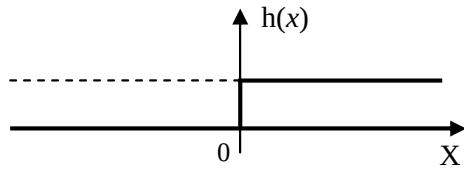


Figure 3.3. Fonction de Heaviside

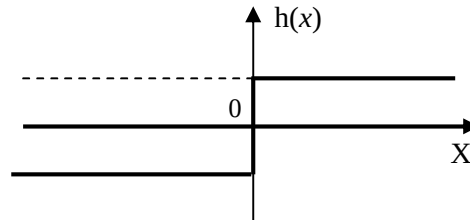


Figure 3.4. Fonction signe

b) Fonction Linéaire à seuil ou à multi seuils:

$$\text{Sature}(X) = \begin{cases} X & \text{si } X \in [U, V] \\ U & \text{si } X \leq U \\ V & \text{si } X > V \end{cases}$$

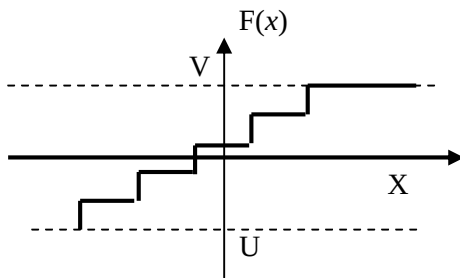


Figure 3.5. Fonction Linéaire multi seuils

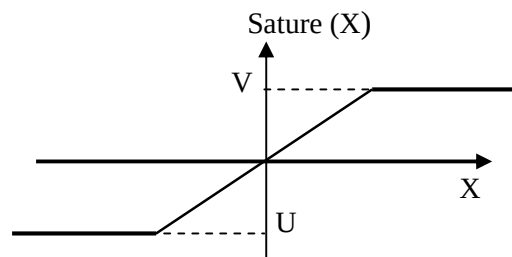
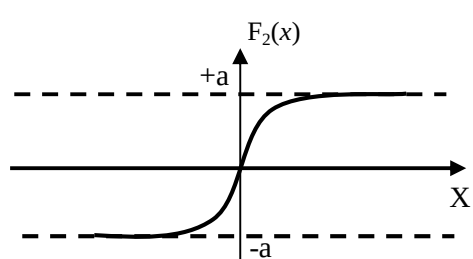
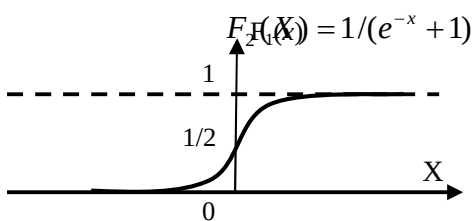


Figure 3.6. Fonction à Seuils

c) La Fonction sigmoïde:

$$F_1(X) = a(e^{kx} - 1) / e^{kx} + 1$$



III.6

Figure 3.7. Fonctions sigmoïdes

4. L'apprentissage d'un réseau de neurone

Le problème principal pour les réseaux de neurones est d'arriver à trouver un ensemble de valeurs (poids) aux synapses, qui sont les porteurs de l'information, tel que les configurations d'entrée se traduisent par les réponses voulues, et cela en portant d'une valeur particulière des poids des connections. Le réseau améliore ses réponses en ajustant ces coefficients selon un algorithme ou une règle d'apprentissage.

Généralement, il existe trois grandes classes d'apprentissage.

4.1. Apprentissage supervisé

Pour ce type d'apprentissage (perceptron, adaline, etc....), le réseau doit savoir qu'il a commis une erreur et il doit connaître la réponse qu'il aurait du donner. Pour un stimulus présenté à la couche d'entrée du réseau, la réponse obtenue est comparée avec celle désirée et la modification et les ajustements à apporter aux poids sont déterminés en fonction de l'erreur commise par le réseau. Généralement, les règles d'apprentissage supervisé sont des formes de descente du gradient.

4.2. Apprentissage non supervisé

Pour ce type d'apprentissage (hophield, kohonen, etc...), le but est toujours le même d'atteindre l'ensemble des poids synaptiques pour lesquels le comportement du réseau est optimal. La modification et l'ajustement des poids se font en fonction d'un critère interne, indépendant de la relation entre le comportement du réseau et la tâche qui doit effectuer.

4.3. Apprentissage semi-supervisé

Ce type d'apprentissage suppose qu'un comportement de référence précis n'est pas disponible, mais qu'en revanche, il est possible d'obtenir des indications qualitatives (correcte/incorrecte) sur les performances du réseau.

5. Les règles d'apprentissage

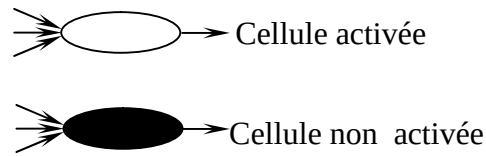
5.1. La règle de Hebb

La règle de Hebb est le premier mécanisme d'évolution proposé sur les synapses. Son interprétation pour les réseaux de neurone formels est la suivante.

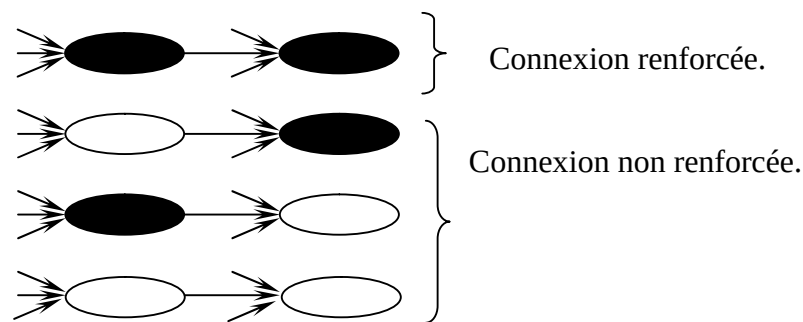
On considère que si deux neurones connectés entre eux sont activés aux mêmes moments, la connexion qui les relie doit être renforcée et elle n'est pas modifiée, dans le cas

contraire. C'est-à-dire que le poids w_{ij} d'une connexion entre un neurone i et un neurone j augmente quand les deux neurones sont activés en même temps et il n'est pas modifié, dans le cas contraire.

Si nous prenons, à titre d'exemple, les connexions suivantes :



La règle de Hebb donne alors:



Ceci se traduit par :

Lorsque la connexion entre deux cellules est très forte, quand la cellule émettrice s'active, la cellule réceptrice s'active aussi, il faut augmenter le poids de cette connexion lors de l'apprentissage.

Si la cellule émettrice s'active sans que la cellule réceptrice ne le soit, ou si la cellule réceptrice s'active alors que la cellule émettrice ne s'était pas activée, cela traduit bien le fait que la connexion entre les deux n'est pas prépondérante dans le comportement de la cellule réceptrice on peut donc dans la phase d'apprentissage laisser un poids faible à cette connexion [19].

En cas se basant sur ce principe, Hebb a donné la règle d'apprentissage suivante :

$$w_{ij}(t + \partial t) = w_{ij}(t) + \mu \cdot A_i A_j \quad 3.7$$

Avec :

$w_{ij}(t)$ et $w_{ij}(t + \partial t)$: les poids de la connexion entre le neurone i et le neurone j aux instants " t " et " $t + \partial t$ ".

A_i et A_j : l'activation du neurone i et l'activation de neurone j ($\mu(\mu > 0)$), sont des paramètres de l'intensité de l'apprentissage.

5.2. Règle de Widrow Hoff

La règle de Widrow Hoff ou règle delta proposée en 1960, consiste à modifier, à chaque pas, les poids et les biais afin de minimiser la somme des carrés des erreurs en sortie en utilisant la règle suivante [20] :

$$w(k+1) = w(k) + \eta(t_k - y_k)x_k^T \quad 3.8$$

A chaque pas d'apprentissage k , l'erreur en sortie est calculée comme la différence entre la cible recherchée t et la sortie y du réseau.

$$E_k = e_k^T e_k = (t_k - y_k)^T (t_k - y_k) = \frac{1}{2}(t_k^T t_k + y_k^T y_k - 2y_k^T t_k) \quad 3.9$$

Le gradient de cette quantité par rapport à la matrice de poids w est donné par :

$$\frac{\nabla E_k}{w} = \frac{1}{2} \cdot \frac{\nabla(y_k y_k^T - 2y_k^T t_k)}{w} \quad 3.10$$

Le gradient se calcul comme suit :

$$\frac{\nabla E_k}{w} = \frac{\partial E_k}{\partial w} = \frac{\partial E_k}{\partial y_k} \cdot \frac{\partial y_k}{\partial w} \quad 3.11$$

D'après l'expression de E_k et avec $y_k = wx_k + b$ les dérivées partielles sont :

$$\frac{\partial E_k}{\partial y_k} = y_k - t_k \quad 3.12$$

$$\frac{\partial (wx_k + b)}{\partial w} = x_k^T \quad 3.13$$

La mise à jour des poids se fait par l'équation :

$$w(k+1) = w(k) - \eta \cdot \frac{\nabla E_k}{w} \quad 3.14$$

Avec η : le gain d'apprentissage ($0 < \eta < 1$).

De même, on obtient l'expression de la modification du biais :

$$b(k+1) = b(k) + \eta(t_k - y_k) = b(k) - \eta \cdot \frac{\nabla E_k}{w} \quad 3.15$$

6. Les différentes architectures

Un neurone réalise simplement une fonction non linéaire, paramétrée sur variables d'entrée. L'intérêt des neurones réside dans les propriétés que résultent de leur association en réseaux, c'est-à-dire de la composition des fonctions non linéaires réalisées par chacun des neurones. On distingue deux types de réseaux de neurones; les réseaux à couche (non bouclés) et les réseaux entièrement connectés (bouclés) [13].

6.1. Les Réseaux à couches

Les neurones sont rangés en couches distinctes. Chaque élément ne pouvant recevoir d'impulsion que des éléments de la couche précédente et ne pouvant en transmettre qu'aux neurones de la couche suivante. A l'intérieur d'une même couche, les neurones ne sont pas liés entre eux. (Figure 3.8)

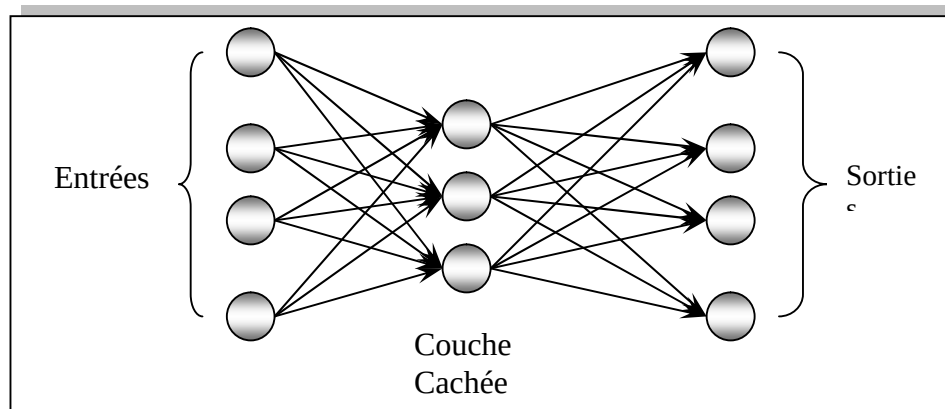


Figure 3.8. Réseaux à couche (non bouclé)

6.2. Les réseaux entièrement connectés "Feed back network"

Un neurone reçoit des impulsions de tous ses compagnons et même de lui-même. Puisque tous les éléments sont connectés entre eux, il n'y a en général qu'une seule couche vue et au mieux deux; une d'entrée et une de sortie [9].

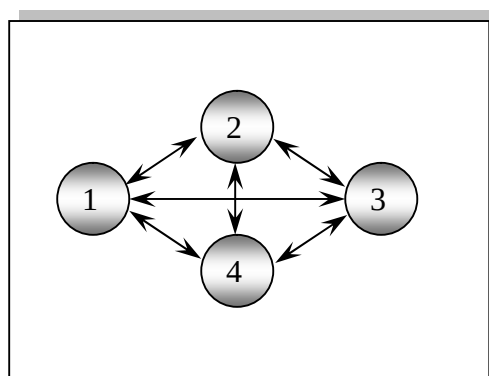


Figure 3.9. Réseaux entièrement connectés (bouclés).

7. Mise en œuvre de réseaux de neurones

Les réseaux de neurones réalisent des fonctions non linéaires paramétrées. Ils peuvent approcher toute fonction de régression si l'on trouve un réseau de neurone qui rend minimum

la fonction de coût des moindres carrés (Somme des carrés des différences entre les valeurs calculées par le réseau de neurones et les valeurs mesurées). La mise en œuvre d'un réseau de neurone nécessite donc [13] :

- De déterminer les entrées pertinentes, c'est-à-dire les grandeurs qui ont une influence significative sur le phénomène que l'on cherche à modéliser.
- De choisir l'architecture du réseau, c'est-à-dire les entrées externes, le nombre de neurones cachés et l'agencement des neurones entre eux, de telle manière que le réseau soit en mesure de reproduire ce qui est déterministe dans les données. Notons que le nombre de poids ajustables est un des facteurs fondamentaux de la réussite d'une application ; si le réseau possède un trop grand nombre de poids, c'est-à-dire si le réseau est très « souple », il risque de s'ajuster au bruit qui est présenté dans les données de l'ensemble d'apprentissage, et même en absence de bruit, il risque de présenter des oscillations non significatives entre les points d'apprentissage donc de présenter de mauvaises propriétés de généralisation. Si ce nombre est trop petit, le réseau est trop « rigide » et ne peut conduire que la partie déterministe de la fonction. Le problème de la détermination de l'architecture optimale est restée pendant longtemps un problème ouvert, mais il existe actuellement diverses méthodes, mettant notamment en jeu des tests statistiques qui permettent de déterminer cette architecture pour une vaste classe de réseaux [14].
- De calculer les poids du réseau ou, en d'autres termes, d'estimer les paramètres de la régression non linéaire, à partir des exemples, en minimisant l'erreur d'approximation, de telle manière que le réseau réalise la tâche désirée.
- D'estimer la qualité du réseau obtenu en lui présentant des exemples qui ne font pas partie de l'ensemble d'apprentissage.

8. Domaine d'application des réseaux de neurones

Les réseaux de neurones artificiels ne sont plus de simples outils de filtrage réservés à des spécialistes de la programmation, mais deviennent des outils courant dans un grand nombre de domaines.

Beaucoup d'applications ont vu le jour [15]. Par exemple, dans le domaine financier:

- Comportement humain vis-à-vis du remboursement de crédit (analyse de risque, crédit scoring). Dans ce cas, on cherche à partir d'une description du client, à créer un modèle qui prédit la probabilité de "casse" du prêt.

- Comportement d'achat (ciblage de marketing, fidélisation, vente croisée), il s'agit également ici d'associer un comportement à un dossier de client (description du client et historique de la consommation). Le modèle construit permettra d'obtenir des prédictions sur le comportement de nouveaux prospects jamais rencontrés auparavant.
- Comportements de systèmes financiers (prédiction de cours, prise de décision d'achat ou de vente), ici les cours passés servent de base à une prédiction des cours futurs, par l'intermédiaire d'un réseau de neurones construit sur les premiers et appliqué aux derniers.

Dans le domaine aérospatial :

- Développement de pilotes automatiques performants.
- Simulation de trajectoires de vols.

Dans le domaine automobile :

- Systèmes de guidage automatique de véhicule.

Dans le domaine de la défense :

- Poursuite de cibles
- Discrimination d'objets
- Reconnaissance des traits du visage
- Extraction de caractéristiques, suppression de bruit, compression de données dans des images radars et sonores.

Dans le domaine du traitement du signal et de la classification:

- Reconnaissance de caractères manuscrits
- Reconnaissance de la parole
- Classification et extraction de signaux d'électroencéphalogrammes (EEG).
- Classification des cellules sanguines
- Inspection de cartes de circuits imprimés.

Dans le domaine du contrôle/commande:

- Contrôle de bras manipulateur
- Contrôle de robots mobiles
- Commande et régulation de réacteurs chimiques
- Analyse et maintenance de machines

La liste est loin d'être exhaustive, mais elle témoigne de la vitalité de ce secteur de recherche. C'est surtout ces deux dernières décennies que les applications ont vu le jour, et cela ne fait que continuer. En témoigne aussi le fait que les réseaux de neurones artificiels, de

plus en plus formalisés, sont devenus une discipline d'enseignement en écoles d'ingénieurs et dans les cursus de deuxième et troisième cycles universitaires.

9. Limites et avantages des réseaux de neurones

Les réseaux de neurones ont des limites et des avantages.

9.1 Limites

Comme pour tout outil, il est important de réaliser que les réseaux ont leurs limites d'applications

- Difficulté d'utilisation du parallélisme inhérent aux réseaux de neurones : la plupart des réseaux sont simulés sur des machines séquentielles ce qui entraîne des temps de calcul important
- Le passage de la simulation à l'implantation des réseaux neurones sur hardware n'est qu'à ses débuts alors que c'est la seule façon d'exploiter réellement leur parallélisme.
- Les performances des réseaux neuronaux dépendent de la quantité et de la pertinence du prétraitement effectuée lors de l'apprentissage.
- L'un des principaux reproches fait aux réseaux neuronaux est leurs incapacité à expliquer les résultats qu'ils fournissent.

Ces réseaux se présentent comme des boîtes noires dont les règles de fonctionnement sont inconnues, la qualité de leurs performances ne peut être mesurée que par des méthodes statistiques, ce qui cause une certaine méfiance de la part des utilisateurs.

9.2 Avantages

Ils ont la capacité de représenter n'importe quelle dépendance fonctionnelle. Le réseau découvre (apprend, modélise) la dépendance lui-même sans avoir besoin qu'on lui "souffle" quoi que ce soit. Il n'a pas besoin de lui postuler un modèle, de l'amender, etc.

On passe directement des données au prédicteur, sans intermédiaire, sans recodage, sans discrétisation, sans simplification ou interprétation sujette à caution.

Ils résistent au bruit ou au manque de fiabilité des données.

Ils disposent d'une grande variété possible pour la variable prédite : oui/non, valeur continue, une ou plusieurs classes parmi n, etc.

Ils sont simples à manier, beaucoup moins de travail personnel à fournir que dans l'analyse statistique classique. Aucune compétence en maths, informatique ou statistiques requise. On n'a pas besoin de détecter les colinéarités, non plus.

En segmentation, le réseau de neurones détermine de lui-même combien de clusters distincts renferment chaque classe.

Leur comportement est moins mauvais en cas de faible quantité de données.

Pour l'utilisateur novice, l'idée d'apprentissage est plus simple à comprendre que les complexités des statistiques multivariées.

Consultation rapide (10 microsecondes câblé-50 millisecondes sur IBM-PC).

Relations spatiales (localisation géographique etc.) faciles à analyser et modéliser.

Le modèle fourni est continu et dérivable et se prête bien à la visualisation, à une recherche de profils-type (par ascension du gradient de probabilité prédite), au calcul de sensibilités (par dérivation partielle), etc.

10. Quelques modèles de réseaux de neurones

10.1 Le modèle du perceptron

Présenté originellement par Rosenblatt, en 1958, le perceptron est la forme la plus simple du réseau de neurones. Il permet de classer correctement des objets appartenant à deux classes linéairement séparables.

10.1.1 Architecture du réseau

Le perceptron se compose de deux couches de neurones (fig. 3.9), la première était appelée à l'origine la rétine du perceptron, quand la deuxième couche donne la réponse du perceptron correspondant à la stimulation présentée en entrée.

Les neurones de la première couche répondent par oui ou non, la réponse oui correspond à une valeur 1 en sortie pour le neurone, la réponse non correspond à une valeur 0. Les cellules de la rétine sont reliées aux cellules de sortie grâce à des synapses d'intensité variable.

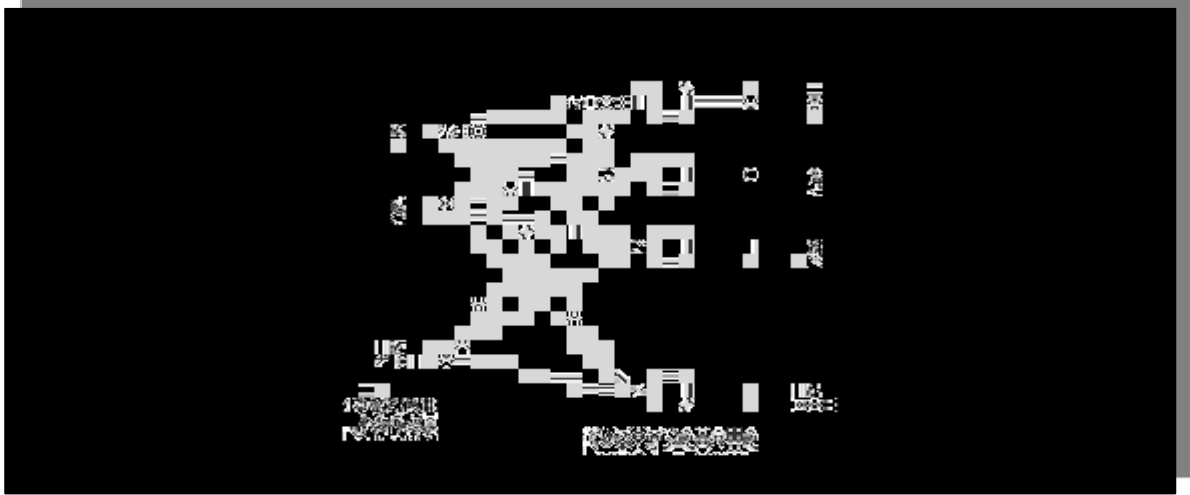


Figure 3.9. L'Architecture du perceptron

L'apprentissage pour le perceptron en modifiant l'intensité de ses synapses. Les cellules de sortie évoluent l'intensité de la simulation en provenance des cellules de la rétine en effectuant la somme des intensités des cellules actives, avec la formule suivante :

$$aj = \sum_i^1 x_i w_{ij} \quad 3.16$$

Avec :

a_j : activation de la $j^{\text{ème}}$ cellule de la couche de sortie.

x_j : valeur de sortie (0 ou 1) de la $i^{\text{ème}}$ cellule de la rétine

w_{ij} : intensité de la connexion entre $i^{\text{ème}}$ cellule de la rétine et la $j^{\text{ème}}$ cellule de la couche de sortie.

Les cellules de sortie deviennent actives si leur degré d'activation dépassé un seuil noté θ_j généralement fixé à 0, mais sans que cela soit nécessaire.

10.1.2. Règle d'apprentissage

Les paramètres du perceptron, c'est à dire les points synaptiques des neurones, peuvent être déterminés grâce à un entraînement supervisé, effectué sur un ensemble de formes pré classifiées. La règle d'apprentissage du perceptron prend le nom de « règle de window hoff ». Pour apprendre, le perceptron doit savoir qu'il a commis une erreur, et il doit connaître la réponse qu'il aurait du donner. De ce fait, on parle d'apprentissage supervisé.

L'apprentissage se produit jusqu'à ce que le perceptron soit capable de donner toutes les réponses correctes .

La règle d'apprentissage de Window Hoff s'écrit de la manière suivante :

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \eta(t_j - o_j)x_i = w_{ij}^t + \Delta w_{ij} \quad 3.17$$

Avec:

Δw_{ij} : changement à effectuer pour la valeur w_{ij} ,

x_i : valeur de sortie (0 ou 1) de la $i^{\text{ème}}$ cellule de la rétine,

o_j : réponse de la $j^{\text{ème}}$ cellule de sortie (0 ou 1),

t_j : réponse technique (ou désirée) de la $j^{\text{ème}}$ cellule de la couche de sortie (0 ou 1),

$w_{ij}^{(t)}$: intensité de la connexion entre la cellule de la rétine et la $j^{\text{ème}}$ cellule de sortie au temps t

η : une constante positive généralement comprise entre 0 et 1.

10.2. L'ADALINE

Conçu par B. Window [16] dans les années 1960, l'ADALINE (Adaptative Linear Element) est un perceptron sans couche cachée, donc, à un seul neurone qui reçoit le stimulus arrivant de la couche d'entrée et donne la réponse correspondante.

10.2.1. Architecture du réseaux

L'ADALINE est un neurone qui possède des valeurs d'activation continues et une fonction d'activation linéaire.

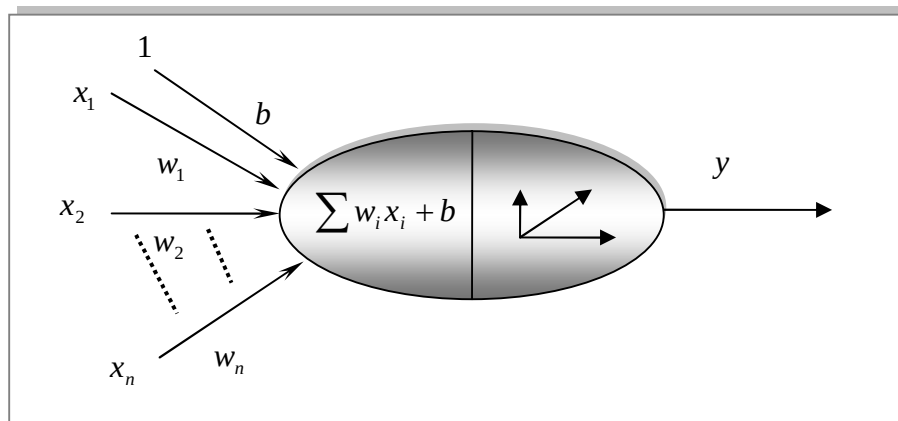


Figure III.11 Architecture de l'ADALINE

10.2.2 Règle d'apprentissage

La règle d'apprentissage de l'adaline, appelée règle de Window Hoff ou LMS (règle de Moindre carrées) utilise la descente de gradient de la fonction de la somme des erreurs quadratiques, donnée par :

$$E(w) = \frac{1}{2} \sum_{k=1}^m (t_k - y_k)^2 \quad \text{III.18}$$

avec

$$y_k = w^t x_k \quad \text{III.19}$$

La descente du gradient donne la direction qui minimise $E(w)$:

$$w(t+1) = w(t) - \eta \nabla E(w) \quad \text{III.20}$$

10.3. Le réseau de Hopfield

Le modèle de Hopfield a été proposé en 1982. C'est un réseau récurrent dont les cellules sont complètement interconnectées, il n'y a plus la notion de couche comme dans le perceptron. Les neurones sont à sorties binaires (à deux états -1 et 1 ou 0 et 1), dont la loi d'apprentissage est la règle de Hebb.

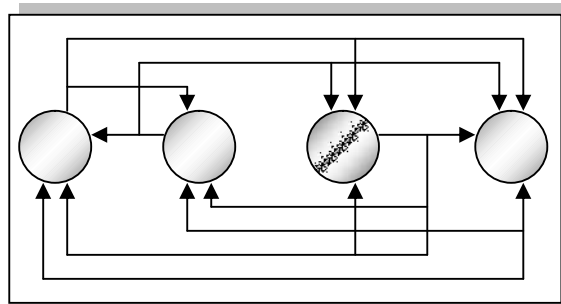


Figure III.12. Le réseau de Hopfield

10.4. La carte auto organisatrice de Kohonen

Le modèle a été proposé dès les années 70 par Von der Marsberg, puis par Kohonen. Ce modèle a pour objectif de représenter des données complexes. Il est réalisé à partir d'un réseau de K neurone à n entré et donc k sorti.

Les entrées sont des vecteurs à n composantes toutes connectées totalement aux k neurones du réseau par nk connexions modifiables. Les neurones du réseau sont placés en général dans un espace de 1 ou 2 dimensions.

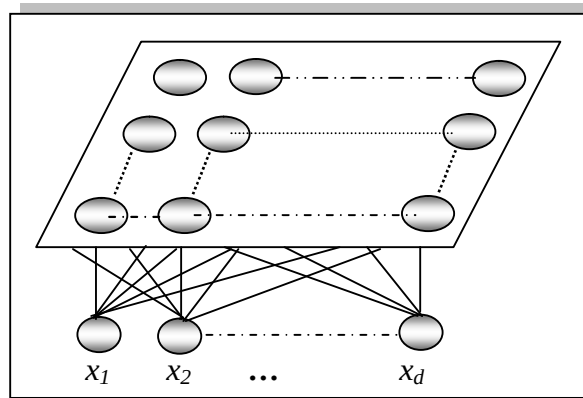


Figure III.13. Réseaux de Kohonen.

10.5. Le Perceptron multicouche (PMC)

Ils sont une amélioration du perceptron comprenant une ou plusieurs couches intermédiaires dites cachées. Ils utilisent, pour modifier leurs poids, un algorithme de rétro propagation du gradient, qui est une généralisation de la règle de Window Hoff. Il s'agit toujours de minimiser l'erreur quadratique, ce qui est assez simple quand on utilise une fonction f dérivable (la sigmoïde par exemple) et consiste à propager la modification des poids de la couche de sortie jusqu'à la couche d'entrée.

10.5.1. Architecture de réseaux

La mise en cascade du perceptron conduit à ce qu'on appelle le perceptron multicouche (Fig.III.14).

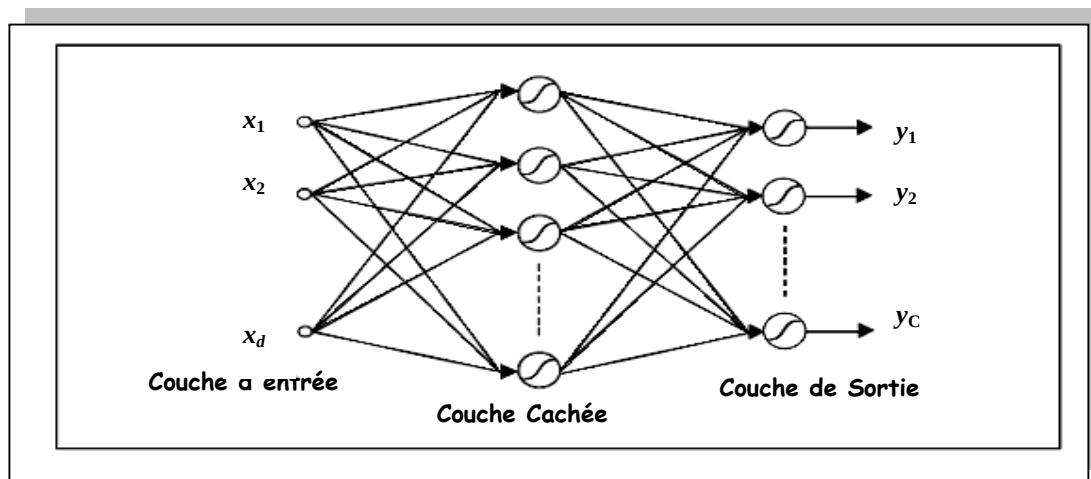


Figure III.14. Architecture du perceptron multicouches

Lorsque le vecteur de caractéristiques d'un objet est présenté à l'entrée du réseau, il est communiqué à tous les neurones de la première couche. Les sorties des neurones de cette couche sont alors communiquées aux neurones de la couche suivante et ainsi de suite. La dernière couche du réseau est appelée couche de sortie, les autres étant désignées sous le terme de couches cachées car les valeurs de sortie de leurs neurones ne sont pas accessibles de l'extérieur [10].

La sortie du réseau aura une expression de la forme suivante :

$$y = f\left(\sum Z\left(f\left(\sum wx + w_0\right)\right) + Z_0\right) \quad 3.25$$

La fonction d'activation des neurones doit absolument être non linéaire, car sans elle, le perceptron multicouche ne ferait qu'implanter une série de transformations linéaires consécutives, qui pourraient dès lors se réduire à une seule. L'algorithme d'apprentissage des perceptrons multicouches, comme sous le nom d'algorithme de rétro propagation, nécessite toute fois que les fonctions d'activations des neurones soient continues et dérivables.

10.5.2. La règle d'apprentissage

L'apprentissage du perceptron multicouches est supervisé et consiste à adopter les poids des neurones de manière à ce que le réseau soit capable de réaliser une transformation donnée, présentée par un ensemble d'exemples constitué d'une suite de N vecteurs d'entrées

$$X_k = [x_{k1} \ x_{k2} \ \dots \ x_{kd}]^T \text{ associés à une autre suite de vecteurs de sorties désirées } T^k = [t_1^k \ t_2^k \ \dots \ t_{hl}^k]^T.$$

A chaque présentation d'un stimulus n° k on réalise les étapes successives suivantes :

- 1- passage du stimulus de l'entrée vers la sortie du réseau ;
- 2- la rétro propagation de l'erreur dans le réseau pour la mise à jour des matrices des poids et de biais afin de réduire l'erreur entre la sortie obtenue et la cible recherchée.

1-Passage de stimulus à travers le réseau :

Chaque cellule de la couche cachée somme toutes ces entrées et fait passer le signal obtenu à travers sa fonction de transfert.

De même pour les cellules de la couche de sortie sous forme matricielle, le vecteur de sortie n° k à pour expression

$$O_k = g\left[Z\left(f\left(wx_k + w_0\right)\right) + Z_0\right] \quad 3.26$$

Où f et g sont les fonctions de transfert (activation), respectivement des couches cachées et de sortie.

Le vecteur de sortie est comparé à celui que l'on désire obtenir, le vecteur t_k en déduit

L'erreur :

$$e_k = t_k - o_k \quad 3.27$$

2-Rétro propagation de l'erreur :

L'apprentissage du réseau consiste à modifier, à chaque pas d'apprentissage, les poids et les biais afin de minimiser la somme des carrées des erreurs en sortie. La méthode de rétro propagation est basée sur la technique du gradient :

$$E_k = e_k^t e_k = \frac{1}{2} (t_k^t t_k + o_k^t o_k - 2 o_k^t t_k) \quad 3.28$$

$$\nabla E_k / Z = \frac{1}{2} [\nabla o_k^t o_k - 2 o_k^t t_k] / Z \quad 3.29$$

Si l'on considère les fonctions de transferts identiques pour les deux couches ($f \equiv g$), la sortie du réseau est donnée par :

$$o_k = f[Z h_k + Z_0] \quad 3.30$$

Le gradient de E_k par rapport à la matrice Z peut se calculer comme suit :

$$\nabla E_k / Z = \frac{\partial E_k}{\partial Z} = \frac{\partial E_k}{\partial o_k} \frac{\partial o_k}{\partial (Z h_k + Z_0)} \frac{\partial (Z h_k + Z_0)}{\partial Z} \quad 3.31$$

D'après l'expression de E_k , la première dérivée partielle est :

$$\frac{\partial E_k}{\partial o_k} = o_k - t_k \quad 3.32$$

La deuxième dérivée partielle dépend du type de la fonction de transfert utilisée. Dans le cas de sigmoïde unipolaire, la deuxième dérivée partielle a pour expression :

$$\frac{\partial o_k}{\partial (Z h_k + Z_0)} = f(Z h_k + Z_0) \cdot [1 - f(Z h_k + Z_0)] = o_k \cdot (1 - o_k) \quad 3.33$$

où 1 représente un vecteur unitaire de même taille que le vecteur de sortie o_k et l'opérateur « $\cdot$ » définit le produit terme à terme ou produit de hadamard.

La troisième dérivée partielle est simple à calculer :

$$\frac{\partial (Z h_k + Z_0)}{\partial Z} = h_k^T \quad 3.34$$

La mise à jour se faisant dans le sens inverse du gradient, la matrice de poids Z et l'étape future ($k+1$) est :

$$Z(k+1) = Z(k) - \eta \nabla E_k / Z = Z(k) - \eta (t_k - o_k) \cdot * o_k \cdot * (1 - o_k) h_k^T = Z(k) + \eta \partial_s h_k^T \quad 3.35$$

Avec :

$$\partial_s = (t_k - o_k) \cdot * o_k \cdot * (1 - o_k) \quad 3.36$$

η : le gain d'apprentissage.

La même méthode sera utilisée pour la mise à jour du vecteur des biais Z_0 . En considérant que le vecteur d'entrée passant par les biais est unitaire. On obtient facilement :

$$Z_0(k+1) = Z_0(k) - \eta \nabla E_k / Z_0 = Z_0(k) - \eta (t_k - o_k) \cdot * o_k \cdot * (1 - o_k) = Z_0(k) + \eta \partial_s \quad 3.37$$

L'erreur de sortie ∂_s est rétro propagée à la sortie de la couche cachée à travers la transposée de la matrice Z .

Par analogie avec l'expression précédente la remise à jour de la matrice Z , où h_k est le vecteur de sortie de la couche cachée, on obtient :

$$w(k+1) = w(k) - \eta \nabla E_k / w = w(k) + \eta Z(k)^T \partial_s \cdot * h_k \cdot * (1 - h_k) x_k^T = w(k) + \eta \partial_h x_k^T \quad 3.38$$

$$w_0(k+1) = w_0(k) - \eta \nabla E_k / w_0 = w_0(k) + \eta Z(k)^T \partial_s \cdot * h_k \cdot * (1 - h_k) = w_0(k) + \partial_h \quad 3.39$$

Où

$$\partial_h = \eta Z(k)^T \partial_s \cdot * h_k \cdot * (1 - h_k) \quad 3.40$$

Formalisation :

Nous donnons ici une formalisation assez sommaire de cet algorithme. Notons :

x : matrice de stimulus de dimension (1, N) où N représente le nombre de stimulus à présenter au réseau.

x_k : stimulus n° k (colonne n° k de la matrice x)

T : matrice cible ou réponse théorique de dimensions (1, w)

t_k : cible n° k (colonne n° k de la matrice T)

h_k : sortie de la couche cachée de dimension (L, i).

o_k : réponse du réseau, vecteur colonne à j composantes.

w : matrice de poids des connexions liant la couche d'entrée à la couche cachée, de dimension (L, i).

w_0 : vecteur de biais des cellules de la couche cachée de dimension (L, 1)

Z : matrice de poids de connexions liant la couche cachée à la couche de sortie de dimension (j, L).

Z_0 : vecteur de biais des cellules de la couche de sortie de dimension (j, 1).

11. Exemple d'apprentissage (Back propagation):

La théorie se rattachant à l'apprentissage des RNA peut paraître complexe à première vue. Bien que certains algorithmes soient réellement complexes, l'apprentissage supervisé d'un réseau à propagation avant avec un des algorithmes les plus populaire, soit « Back propagation », est plutôt simple. Voici les différentes étapes à suivre lors de l'apprentissage d'un réseau de neurones à propagation avant avec l'algorithme « Back propagation ».

1er étape : Initialiser les poids des liens entre les neurones. Souvent une valeur entre 0 et 1, déterminée aléatoirement, est assignée à chacun des poids.

2e étape : Application d'un vecteur entrées-sorties à apprendre.

3e étape : Calcul des sorties du RNA à partir des entrées qui lui sont appliquées et calcul de l'erreur entre ces sorties et les sorties idéales à apprendre.

4e étape : Correction des poids des liens entre les neurones de la couche de sortie et de la première couche cachée selon l'erreur présente en sortie.

5e étape : Propagation de l'erreur sur la couche précédente et correction des poids des liens entre les neurones de la couche cachée et ceux en entrées.

6e étape : Boucler à la 2e étape avec un nouveau vecteur d'entrées-sorties tant que les performances du RNA (erreur sur les sorties) ne sont pas satisfaisantes.

Les figures ci-dessous présentent un RNA dont les poids des liens entre les neurones et celui des biais ont été déterminés aléatoirement. Dans cet exemple, le vecteur à apprendre comporte les entrées $x_1 = 0.8$ et $x_2 = 0.2$, pour une sortie souhaitée $d = 0.4$. La fonction d'activation est la sigmoïde (équation (1)) et sa dérivée est exprimée par l'équation (2). L'équation (3) illustre comment calculer la sortie d'un neurone et l'équation (4) comment modifier les poids entre les neurones lors de l'apprentissage (taux d'apprentissage $\mu = 1$). Remarquez que les poids sont toujours incrémentés selon: « $\mu \cdot \text{erreur en sortie} \cdot f'(\text{somme des entrées}) \cdot \text{entrée du lien}$ ».

12. Analyse de la problématique et Choix du type de RNA

D'une part, les perceptron multicouches ont montré leur efficacité pour la classification des formes [5, 6, 20], et d'autre part, dans notre cas d'application, le système de

détection est basé sur la classification des signaux. Par conséquent, l'utilisation d'un réseau de neurones à couches est préférable pour essayer de résoudre le problème posé [6, 9].

Par ailleurs, les réseaux que nous allons simuler sont tous des réseaux à trois couches et les étapes de construction du réseau peuvent être subdivisées de la manière suivante :

- Choix des entrées du réseau, c'est à dire choix du pas d'échantillonnage pour le signal qui va être traité (courant statorique, courant rotorique et vitesse) et détermination du nombre d'entrées du réseau (nombre de neurones de la couche cachée);
- Choix des sorties, c'est à dire détermination du nombre de sorties et leur nature ; pour faciliter l'interprétation des résultats de la sortie du réseau par le système expert, notre choix a été orienté sur les nombres binaires (0,1);
- Comme les sorties sont binaires et les entrées réelles, la fonction de sortie sera une fonction linéaire affine et la fonction d'activation une fonction sigmoïde;
- Détermination du nombre de neurones cachés et du nombre des couches cachées: ils seront déterminés par tâtonnement à partir d'un algorithme d'apprentissage.

13. Choix du type d'algorithme d'apprentissage

Concernant l'apprentissage, le RNA est entraîné par le modèle de rétropropagation, pour les raisons suivantes [5, 6, 9, 20] :

- Le système d'apprentissage est supervisé ;
- Les entrées sont continues ;
- Il a été utilisé avec succès dans plusieurs domaines d'applications.

14. Conclusion

Le présent chapitre a été consacré à la présentation de la théorie des réseaux de neurones, en partant du neurone biologique, jusqu'à l'étude des différents types de réseaux, parmi lesquels on a présenté deux types : les réseaux bouclés et les réseaux non bouclés.

A la lumière de l'étude effectuée, il a été constaté que les réseaux non bouclés correspondent bien à notre cas d'application. En effet, les réseaux de neurones multicouches entraînés par l'algorithme de rétropropagation du gradient sont aujourd'hui les modèles les

plus utilisés en diagnostic en raison de leur capacité à résoudre les problèmes de classification.

Par ailleurs, pour obtenir a priori des informations (sous forme de données) sur le comportement de la machine asynchrone (qui est principalement la tâche du SACI), celles-ci seront substituées par des données obtenues par simulation numérique des différents états de fonctionnement de MASDS.