LE DATA MINING

La fouille de données

Dr. Nedioui Med Abdelhamid
Université Echahid Hama Lakhder Eloued - Algérie
nedioui3904@gmail.com

Chapitre 6

Classification par réseaux de neurones

1. Introduction

Les réseaux de neurones, fabriquées de structures cellulaires artificielles, constituent une approche permettant d'aborder sous des angles nouveaux les problèmes de perception, de mémoire, d'apprentissage et de raisonnement (en d'autres termes 'intelligence artificielle "I.A."). Ils s'avèrent aussi des alternatives très prometteuses pour contourner certaines des limitations des méthodes numériques classiques. Grâce à leur traitement parallèle de l'information et à leurs mécanismes inspirés des cellules nerveuses (neurones), ils infèrent des propriétés émergentes permettant de solutionner des problèmes jadis qualifiés de complexes.

2. Réseaux de neurones :

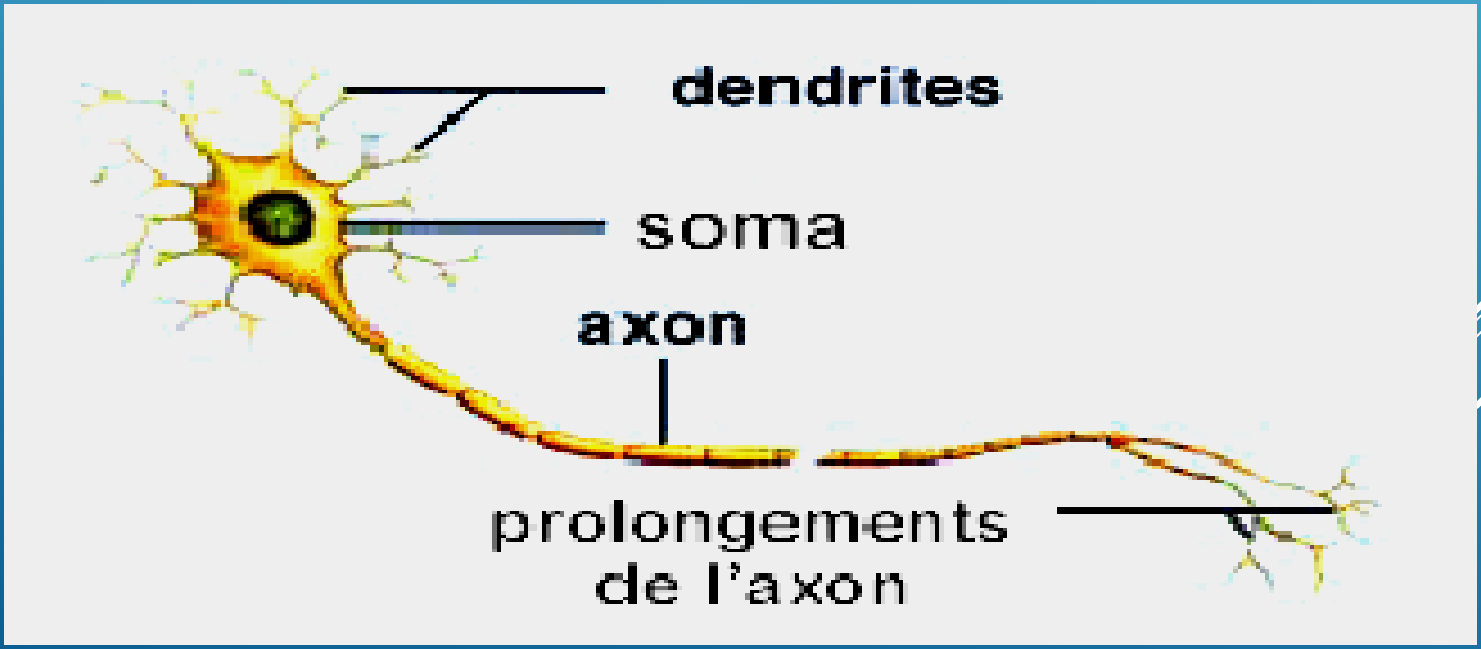
Les réseaux de neurones artificiels fondés sur des modèles qui tentent de simuler les cellules du cerveau humain et leurs interconnexions. Le but, est d'exécuter des calculs complexes et de trouver, par apprentissage, une relation non linéaire entre des données numériques et des paramètres.

2.1. Neurone biologique

Le cerveau humain contient 1000 milliards de cellules, dont 100 milliards sont des neurones constitués en réseaux.

2. Réseaux de neurones :

Le neurone est une cellule nerveuse. Elle se compose d'un corps cellulaire appelé ***Soma*** qui contient le noyau (où se déroule les activités cellulaires vitales) de prolongement appelé ***Neurite***. Ces dernières sont de deux types, les ***Dentrites*** qui servent de canaux d'entrées et l'axone, unique qui est le canal de sortie .



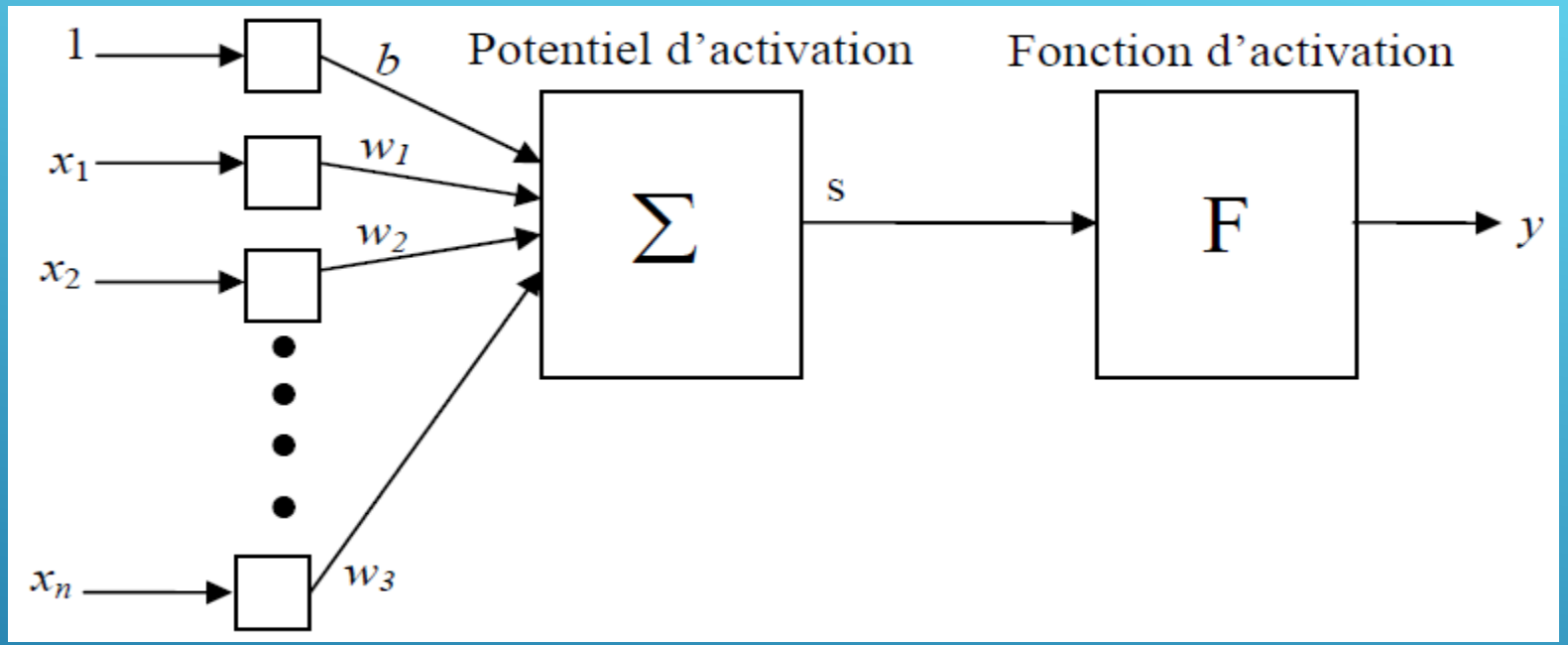
2. Réseaux de neurones :

2.2 Neurone formel

Le neurone formel est une modélisation mathématique qui reprend les principes du fonctionnement du neurone biologique. Sachant qu'au niveau biologique, les connexions entre les neurones n'ont pas toutes la même valeur, les chercheurs ont donc créé un algorithme qui pondère la somme de ses entrées par des poids synaptiques (coefficients de pondération). En général, un neurone formel est un élément de traitement possédant ***n*** entrées $x_1, x_2, .., x_n$ (qui sont les entrées externes ou les sorties des autres neurones) et une ou plusieurs sorties.

2. Réseaux de neurones :

2.2 Neurone formel



Avec :

- Les *xi* sont les entrées du réseau.
- S est le potentiel d'activation.
- Les *wi* représentent les poids synaptiques.
- *yi* la sortie du réseau tels que : $y = f(s)$; $s = \sum wi\ xi + b$

2. Réseaux de neurones :

2.3 Fonction d'activation

La fonction d'activation (ou fonction de seuillage, ou fonction de transfert) sert à introduire une fonction non linéarité dans le fonctionnement du neurone. Les fonctions de seuillage présentent généralement trois intervalles :

- en dessous du seuil, le neurone est non actif (souvent dans ce cas, sa sortie vaut 0 ou - 1).
- aux alentours du seuil, une phase de transition.
- au dessus du seuil, le neurone est actif (souvent dans ce cas, sa sortie vaut 1) .

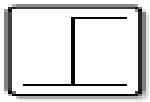
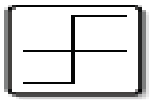
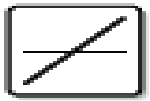
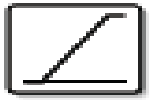
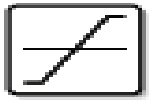
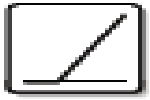
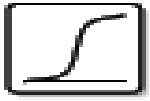
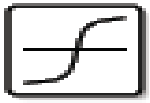
2. Réseaux de neurones :

2.3 Fonction d’activation

Dans sa première version, le neurone formel était implémenté avec une fonction à seuil, mais de nombreuses versions existent. Ainsi, le neurone de McCulloch et Pitts a été généralisé de différentes manières, en choisissant d’autres fonctions d’activations, comme les fonctions énumérées dans le tableau suivant. Les troie fonctions les plus utilisées sont les fonctions « seuil », « linéaire », « sigmoïdes ».

2. Réseaux de neurones :

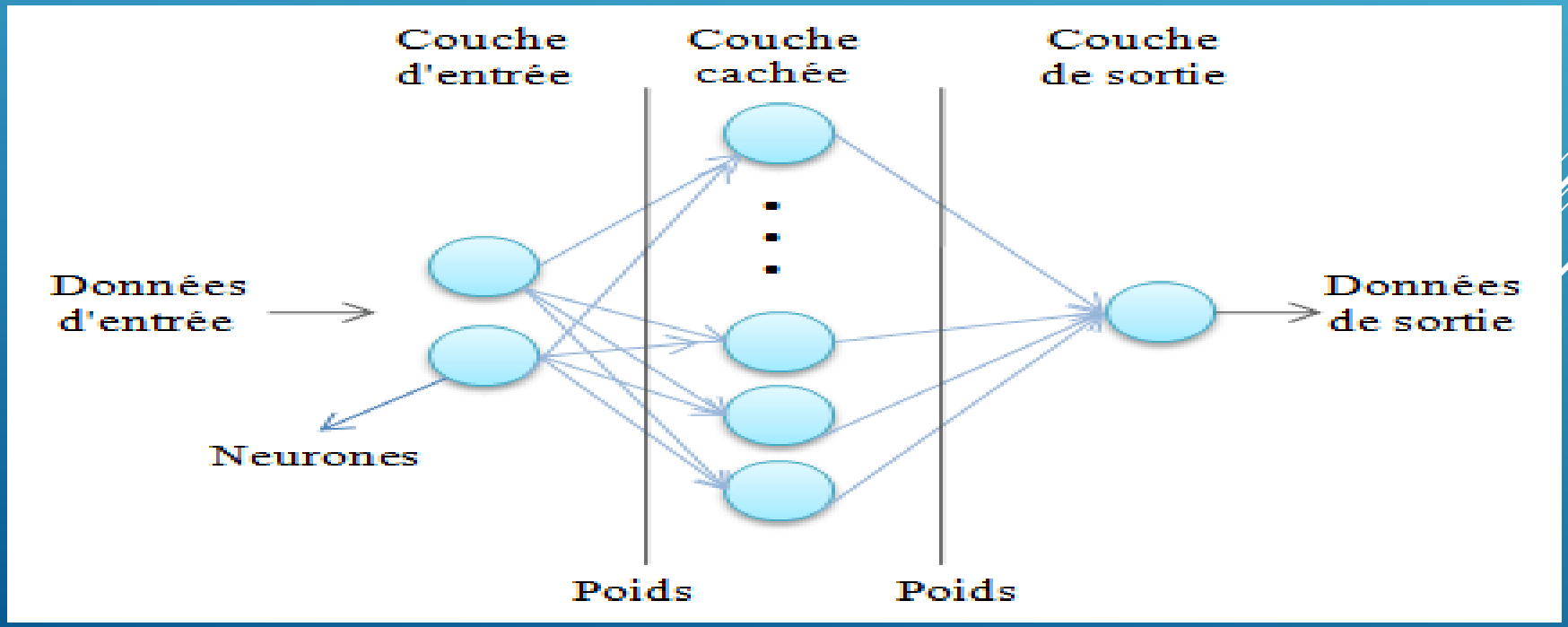
2.3 Fonction d'activation

Nom de la fonction	Relation d'entrée/sortie	Icône
seuil	$a = 0 \quad \text{si } n < 0$ $a = 1 \quad \text{si } n \geq 0$	
seuil symétrique	$a = -1 \quad \text{si } n < 0$ $a = 1 \quad \text{si } n \geq 0$	
linéaire	$a = n$	
linéaire saturée	$a = 0 \quad \text{si } n < 0$ $a = n \quad \text{si } 0 \leq n \leq 1$ $a = 1 \quad \text{si } n > 1$	
linéaire saturée symétrique	$a = -1 \quad \text{si } n < -1$ $a = n \quad \text{si } -1 \leq n \leq 1$ $a = 1 \quad \text{si } n > 1$	
linéaire positive	$a = 0 \quad \text{si } n < 0$ $a = n \quad \text{si } n \geq 0$	
sigmoïde	$a = \frac{1}{1+\exp^{-n}}$	
tangente hyperbolique	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$	
compétitive	$a = 1 \quad \text{si } n \text{ maximum}$ $a = 0 \quad \text{autrement}$	

2. Réseaux de neurones :

2.4 Architecture de réseau

Un réseau de neurones est un maillage de plusieurs neurones, généralement organisés en couches. Pour construire une couche de *S* neurones, il s'agit simplement des les assembler comme à la figure :



2. Réseaux de neurones :

2.4 Architecture de réseau

Un réseau de neurones est une structure de réseau constituée d'un nombre de nœuds interconnectés par des liaisons directionnelles. Chaque nœud représente une unité de traitement et les liaisons représentent les relations causales entre les nœuds.

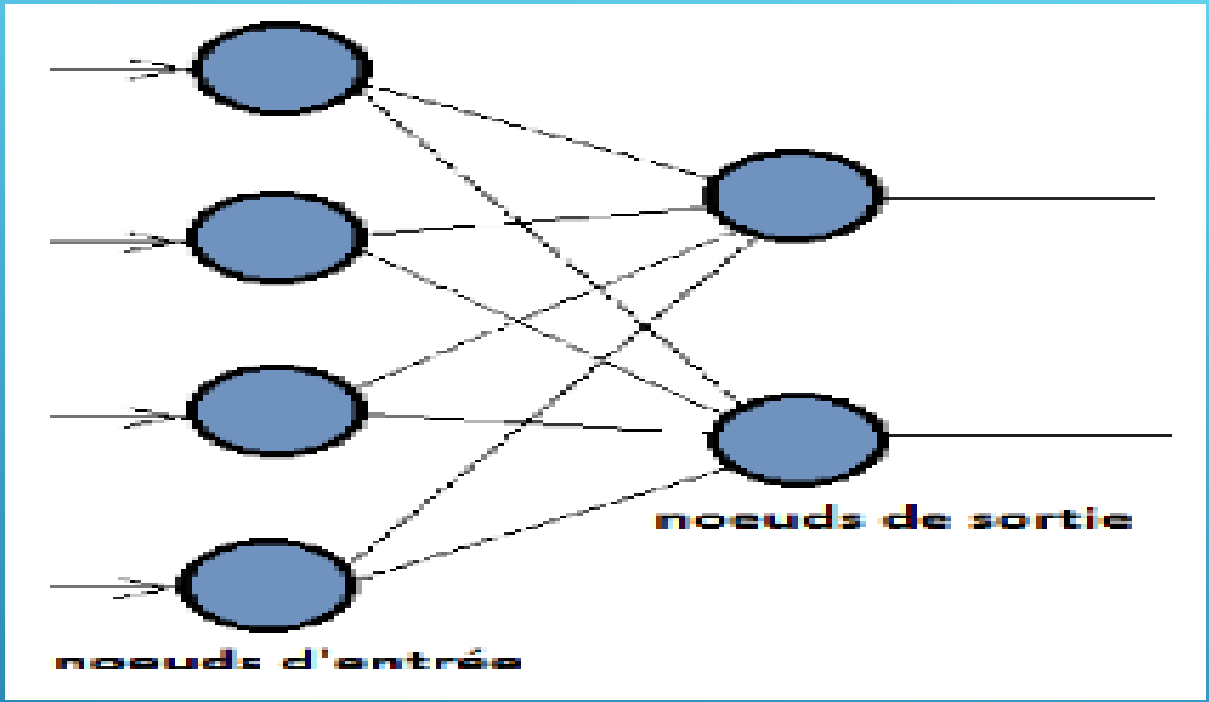
L'architecture d'un réseau de neurones artificiel est définie par la structure de ses neurones et leur connectivité. Elle est spécifiée par le nombre d'entrées, de sorties, de nœuds et la façon selon laquelle sont interconnectés et organisés les nœuds.

2. Réseaux de neurones :

2.4.1 Réseau de neurone monocouche (Perceptron)

Le perceptron simple est dit simple parce qu'il ne dispose que de deux couches ; la couche en entrée et la couche en sortie. Le réseau est déclenché par la réception d'une information en entrée. Le traitement de la donnée dans ce réseau se fait entre la couche d'entrée et la couche de sortie qui sont toutes reliées entre elles. Le réseau intégral ne dispose ainsi que d'une seule matrice de poids, ce que limite le perceptron simple à un classificateur linéaire permettant de diviser l'ensemble d'informations obtenues en deux catégories distingues.

2. Réseaux de neurones :



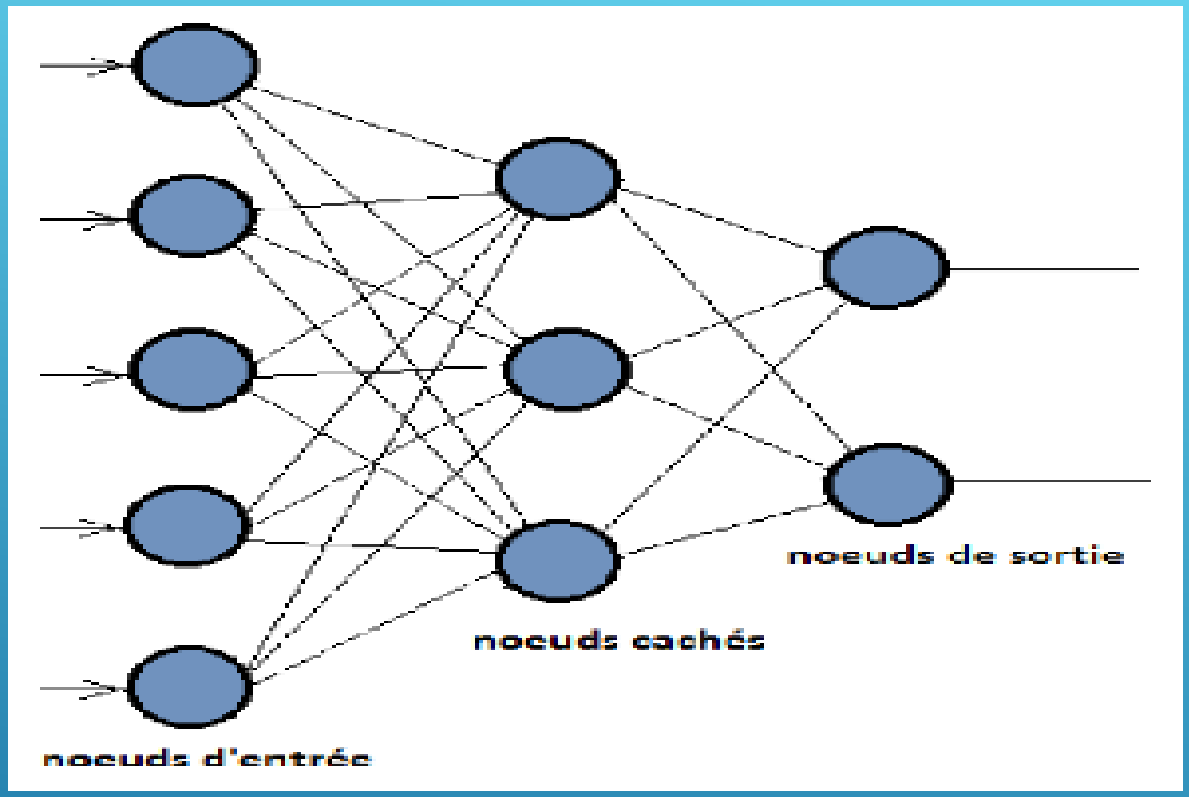
Le réseau de neurones possède ainsi N informations en entrée et P sorties, chaque neurone renvoyant sa sortie. Une utilisation courante est que chaque neurone de la couche sortie représente une classe. Pour un exemple X donné, on obtient la classe de cet exemple en prenant la plus grande des p sorties.

2. Réseaux de neurones :

2.4.2 Réseau de neurone multicouche

Le perceptron multicouche se structure de la même façon. L'information entre par une couche d'entrée et sort par une couche de sortie. À la différence du perceptron simple, le perceptron multicouche dispose entre la couche en entrée et la couche en sortie une ou plusieurs couches dites « cachées ». Le nombre de couches correspond aux nombres de matrices de poids dont disposent le réseau. Un perceptron multicouche est donc mieux adapté pour traiter les types de fonctions non-linéaires.

2. Réseaux de neurones :



Dans ce réseau, les neurones de la première couche reçoivent toutes les informations entrées, ceux de la deuxième reçoivent toutes les sorties des neurones de la première couche, et ainsi de suite jusqu'au neurone de sortie qui reçoit celles de la dernière couche.

2. Réseaux de neurones :

2.5 Exemple

Prenons un exemple concret et cherchons à construire un modèle qui exprime le temps qu'il a fait dans une journée (beau temps ou mauvais temps).

#2	Entrées (X)	Sortie (Y)	Exemples
	Des données météorologiques	Si il fait beau ou mauvais temps	
Algorithme attendu	le modèle qui donne le temps qu'il fait en fonction des données météo.		

Température	Pression	Hygrometrie	Vent	Résultat
20°C	1013hPa	80%	10km/h	Beau
10°C	970hPa	100%	60km/h	Mauvais

Pour cela, il faut lui fournir un certain nombre de paramètres en entrée : La température, L'hygrométrie, La pression atmosphérique et La vitesse du vent. Et lui donner des exemples pour s'entraîner.

2. Réseaux de neurones :

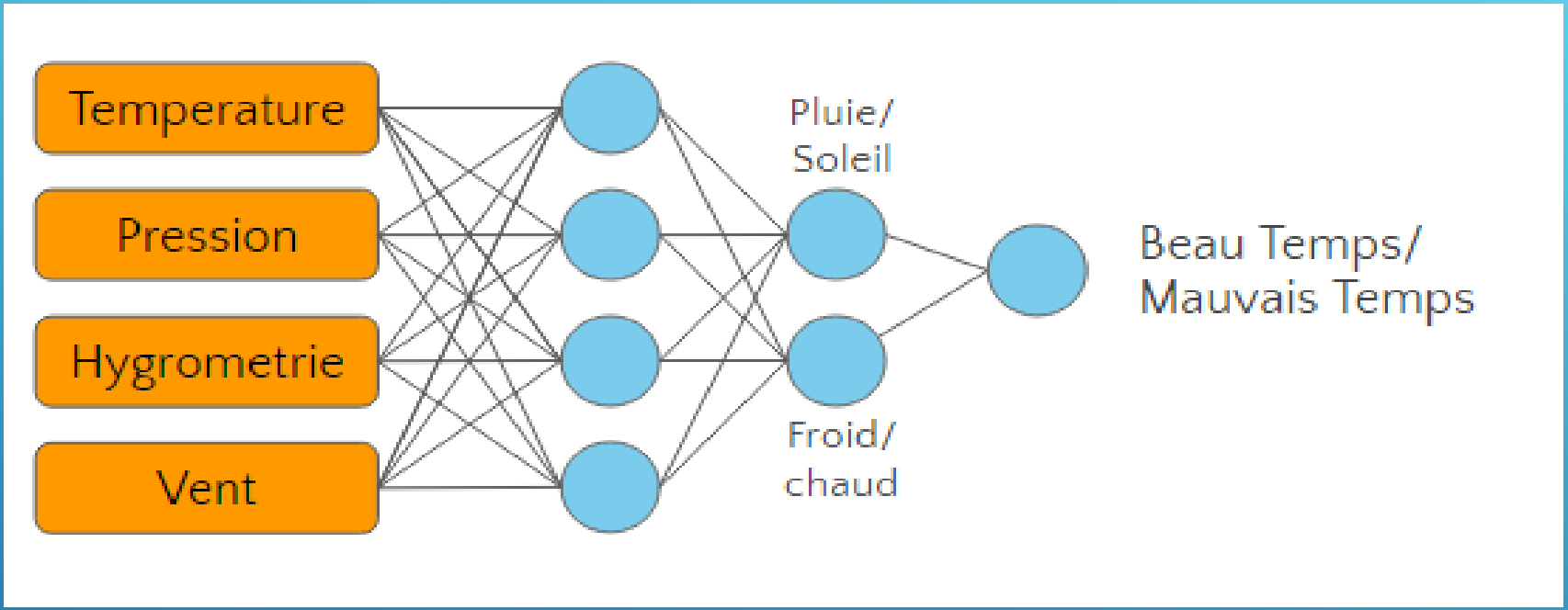
2.5 Exemple

Evidemment plus le nombre d'exemples sera grand et varié, plus l'entraînement permettra d'arriver à un modèle précis et pertinent. Ici il faudrait un grand nombre de cas exposés pour être capable de généraliser la subjectivité liée au beau temps ou au mauvais temps. Pour la simplicité de l'exemple, nous nous en tiendrons à ces quelques valeurs.

Afin modéliser et appréhender cette relation complexe nous avons besoin d'un réseau de neurones. Ce réseau de neurones aura deux couches (et sera donc extrêmement simple) :

2. Réseaux de neurones :

2.5 Exemple

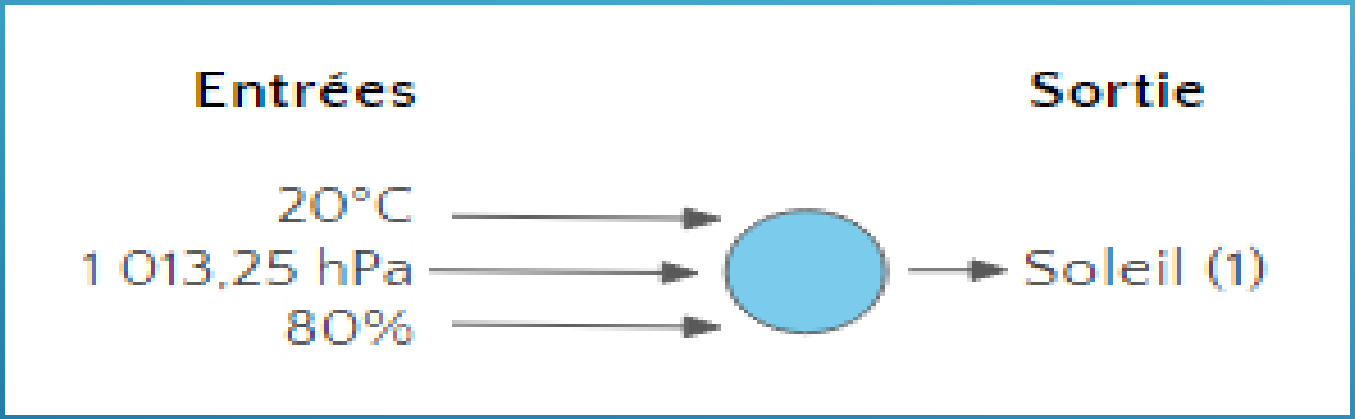


L'intérêt du réseau de neurones est de modéliser l'impact des différents facteurs et leur relation entre eux. Lorsque la complexité des facteurs est grande, plutôt que de les traiter tous ensemble, on décompose l'analyse en étapes, les plus petites possibles.

2. Réseaux de neurones :

2.5 Exemple

Chaque étape est représentée par un neurone. Un neurone reçoit un certain nombre d'informations, chacune pondérée (p), et renvoi une réponse binaire 0 ou 1.

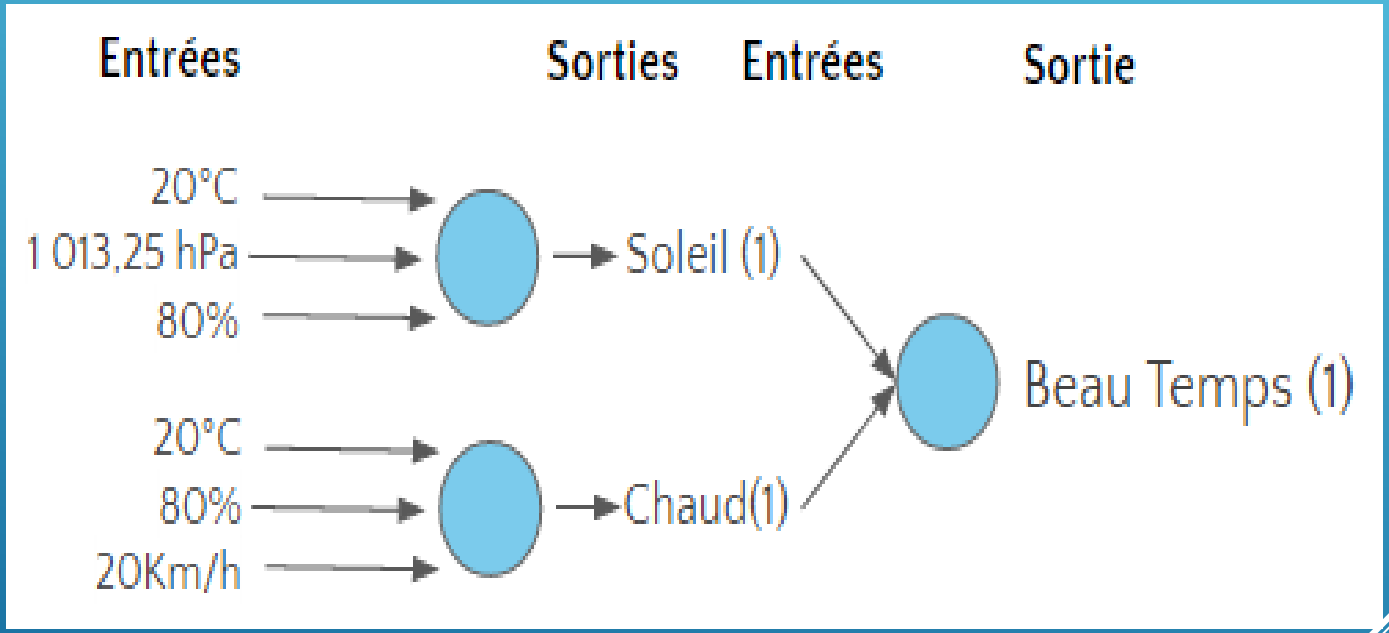


Ensuite cette réponse va venir alimenter le neurone suivant dans le réseau, qui lui-même produira une réponse binaire, et ainsi de suite jusqu'au dernier neurone du réseau.

2. Réseaux de neurones :

2.5 Exemple

La sortie du dernier neurone du réseau représente la réponse que l'on cherche à obtenir.



Pour affiner ce modèle, on joue sur la pondération de chaque entrée et le seuil qui déclenche la sortie 0 ou la sortie 1.

3. Apprentissage des Réseaux de neurones

La tâche principale des réseaux de neurones artificiels est l'apprentissage pour la classification, qui est réalisée par un processus itératif d'adaptation des poids wi pour arriver à la meilleure fonction permettant d'avoir :

$$f(x_i) = y_i, \forall\ i = 1 \ldots N.$$

Les valeurs des wi Sont initialisées aléatoirement, et corrigées selon les erreurs entre les yi obtenus et attendus.

Dans un réseau de neurones multicouches, la correction se fait dans le sens inverse du sens de propagation des données ce qui est appelé la **backpropagation**. À chaque présentation d'un exemple d'apprentissage au réseau, on passe par deux étapes :

3. Apprentissage des Réseaux de neurones

1. Dans l'étape de propagation, les valeurs du vecteur d'entrée (l'exemple) sont reçues dans la couche d'entrée et propagées d'une couche à l'autre jusqu'à la sortie où un vecteur de sortie (les y_i) est obtenu.

2. Dans la phase de backpropagation, les w_i sont ajustés de la dernière couche jusqu'à la première de manière à rapprocher les y_i obtenus de ceux attendus.

Ces deux étapes sont répétées avec chaque exemple d'apprentissage pour obtenir à la fin un réseau de neurones artificiel entrainé. L'utilisation d'un RNA entrainé, se fait par l'injection des valeurs du vecteur de l'exemple à classifier, dans l'entrée et recevoir sa classe à la sortie par parpropagation.

3. Apprentissage des Réseaux de neurones

3.1 Apprentissage par descente de gradient

C'est une méthodes d'apprentissage simple dans un perceptron monocouche qui utilisant l'erreur quadratique. La méthode suit l'algorithme suivant :

Algorithme ApprretissageRNA par décente de gradient

1: Créer n variables dw_i, pour $1 \leq i \leq n$, égales à 0

2: Prendre un exemple e_k, pour $1 \leq k \leq N$

3: Calculer la sortie obtenue avec les poids actuels, notée s_k

4: Rajouter à dw_i, pour tout $1 \leq i \leq n$, le nombre $\alpha(y_k - s_k)x_i$

5: Réitérer (3) et (4) sur chacun des exemples

6: pour $1 \leq i \leq n$, remplacer w_i par $w_i + dw_i$

3. Apprentissage des Réseaux de neurones

α est un nombre réel appelé taux d'apprentissage qui doit être choisi précisément. Une valeur trop grande risque de faire osciller le réseau autour d'un minimum local et une valeur trop petite augmente le nombre d'itérations. En pratique, on diminue graduellement α au fur et à mesure des itérations.

Afin d'obtenir de bons résultats, il faudra passer plusieurs fois les exemples à chaque neurone, de sorte que les poids convergent vers des poids "idéaux".

Le problème avec cette méthode est que l'on corrige sur la globalité des exemples, ce qui fait que le réseau ne s'adaptera aux exemples qu'après un certain moment.

3. Apprentissage des Réseaux de neurones

3.2 Apprentissage par Widrow-Hoff

L'algorithme de Widrow-Hoff, ou encore, n'est en fait qu'une variante de l'algorithme descente de gradient. Cette méthode élaborée par Widrow et Hoff consiste à modifier les poids après **chaque exemple**, et non pas après que tous les exemples aient défilé. Ceci va donc minimiser l'erreur de manière précise, et ce sur chaque exemple. Instinctivement, on constate bien que le réseau de neurones va s'améliorer nettement mieux et va tendre bien plus rapidement à classifier parfaitement (ou presque) chacun des exemples.

3. Apprentissage des Réseaux de neurones

Voici donc l'algorithme de Widrow-Hoff.

Algorithme AprretissageRNA par Widrow-Hoff

Entrée :

- n poids reliant les n informations à notre neurone ayant des valeurs quelconques
- N exemples (X_k, y_k) où X_k est un vecteur à n composantes x_i, chacune représentant une information de cet exemple
- Le taux d'apprentissage α

Sortie : les n poids modifiés

pour Tout exemple $= (X_k, y_k)$ **faire**

Calculer la sortie s_k du neurone

pour $1 \leq i \leq n$ **faire**

$w_i = w_i + \alpha(y_k - s_k)x_i$

fin pour

fin pour

3. Apprentissage des Réseaux de neurones

3.3 Exemple d'apprentissage du OU logique

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

- **x** est le vecteur associé à l 'entrée (x_0, x_1, ..., x_n)
- **y** la sortie correspondante souhaitée
- **s** la sortie calculer de réseau

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

On ajoute une entrée supplémentaire x_0 (**le biais**), avec le coefficient synaptique suivant : $w_0 = 1$

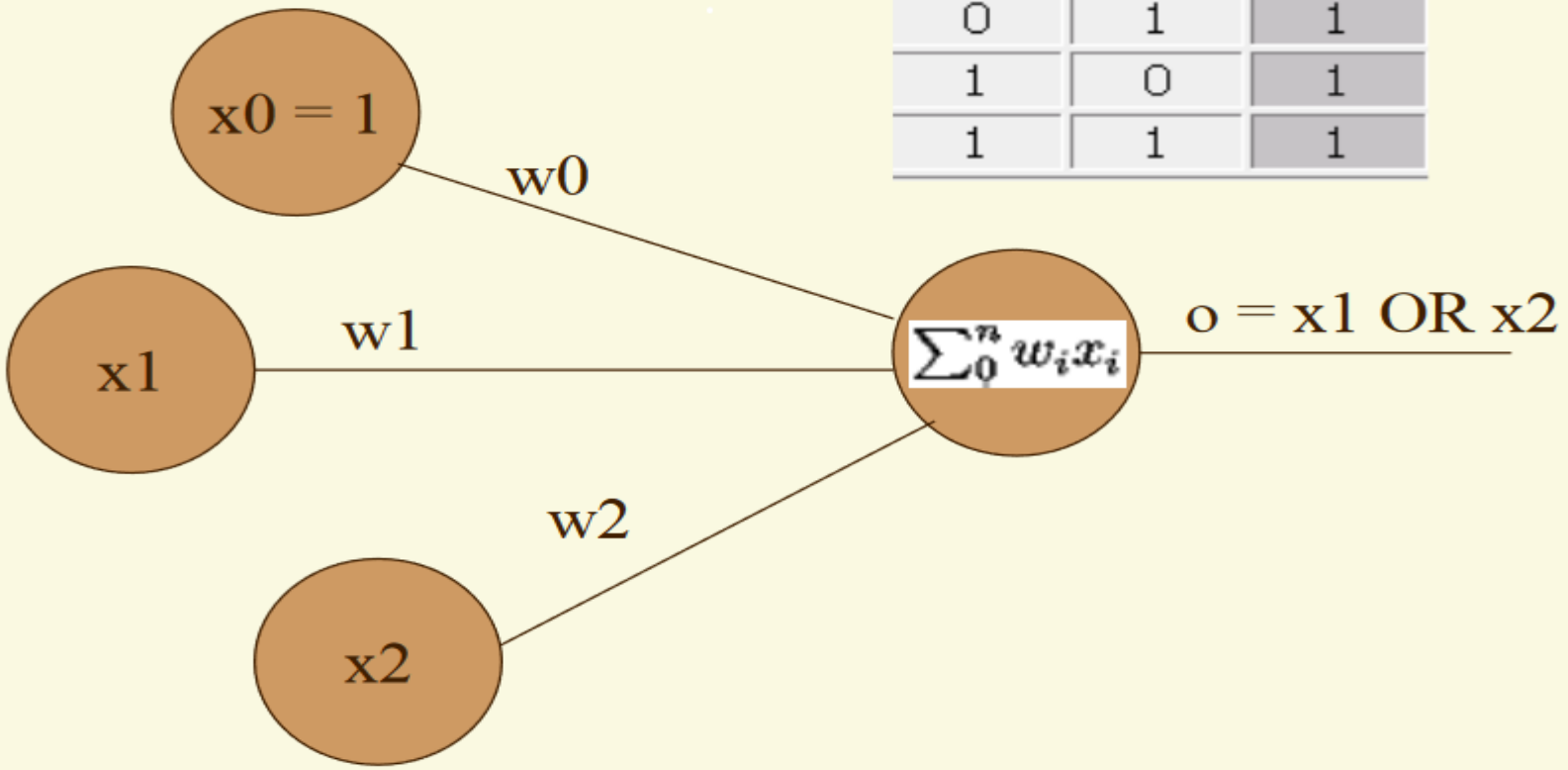
On associe comme fonction de transfert la fonction de **Heavyside** : $f(x) = 1$ si $x > 0$ et $f(x) = 0$ sinon

3. Apprentissage des Réseaux de neurones

3.3 Exemple d'apprentissage du OU logique

Apprentissage du OU logique

a	b	OR
0	0	0
0	1	1
1	0	1
1	1	1



Initialisation : $w_0 = 0$; $w_1 = 1$; $w_2 = -1$; $x_0 = 1$; $\alpha = 1$

3. Apprentissage des Réseaux de neurones

3.3 Exemple d'apprentissage du OU logique

Étape	w_0	w_1	w_2	Entrée	$\sum_0^2 w_i x_i$	S	y	w_0	w_1	w_2
init								0	1	-1
1	0	1	-1	100	0	0	0	0+0x1	1+0x0	-1+0x0
2	0	1	-1	101	-1	0	1	0+1x1	1+1x 0	-1+1x1
3	1	1	0	110	2	1	1	1	1	0
4	1	1	0	111	2	1	1	1	1	0
5	1	1	0	100	1	1	0	1+(-1)x1	1+(-1)x0	0+(-1)x0
6	0	1	0	101	0	0	1	0+1x1	1+1x0	0+1x1
7	1	1	1	110	2	1	1	1	1	1
8	1	1	1	111	3	1	1	1	1	1
9	1	1	1	100	1	1	0	1+(-1)x1	1+(-1)x0	1 +(-1)x0
10	0	1	1	101	1	1	1	0	1	1

Donc : $w_0 = 0$; $w_1 = 1$; $w_2 = 1$

4. Avantages et inconvénients

4.1 Avantage

- L’intérêt essentiel des réseaux de neurones artificiels réside dans le parallélisme de leur structure, leur capacité d’adaptation ainsi que dans leur mémoire distribuée.
- Un réseau de neurones possède également une grande résistance au bruit ou au manque de fiabilité des données.
- L'idée d'apprentissage est plus simple à comprendre que les complexités de s statistiques multi-variables. Elle est intuitive.

4. Avantages et inconvénients

4.2 Inconvénients

- Un RNA représente une boîte noire, et il est très difficile voire impossible d'analyser et comprendre son fonctionnement en face d'un problème donné, ce qui empêche de choisir la structure (type, nombre de nœuds, organisation, connexions,...etc) la mieux adaptée à ce problème.
- L'ordre de présentation des exemples d'entrainement au réseau influe directement sur les résultats obtenus.

4. Avantages et inconvénients

4.2 Inconvénients

Pour surmonter ce problème, il est nécessaire de répéter au moins la phase d'entrainement avec des ordres différents des exemples ce qui augmente considérablement le temps d'apprentissage.

- Dans le cas des bases de données, les RNA ne permettent pas de traiter des exemples avec des attributs symboliques (catégoriels) qu'après un encodage adapté, à l'inverse de plusieurs autres techniques d'apprentissages tel que les SVMs et les arbres de décision.