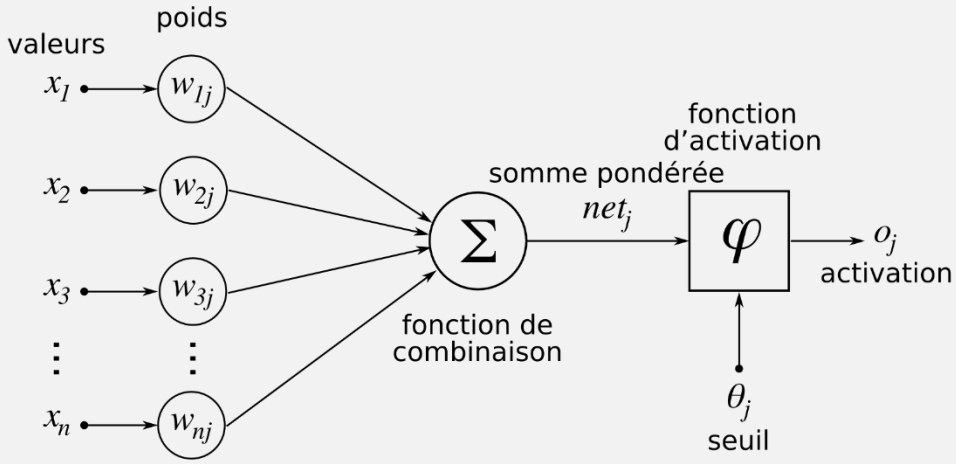
AHAMADA Abdoul-Hakim

N° d'inscription : 6022

MODELISATION A L'AIDE DE RESEAUX DE NEURONES



- Introduction
- Une approche particulière
- Historique des réseaux
- Présentation
- Généralité de la démarche
- Exemple
- Efficacité des réseaux
- Une approche légitime

Introduction

Le cerveau naturel est le plus mystérieux de tous les organes biologiques qui composent un être vivant. Son fonctionnement commence à être bien compris par les scientifiques. Néanmoins la gestion des informations qui lui sont communiquées, leur arrangement, leur interprétation, la capacité et la motivation d'utilisation de ces données, toutes ces actions sont autant de mécanismes difficilement définissables et reproductibles de manière artificielle. Cependant les découvertes effectuées sur les neurones, composants de base d'un cerveau, ont donné lieu à des recherches en intelligence artificielle. L'avènement de l'informatique a contribué au développement de cette science qui semble trouver son intérêt essentiellement dans la modélisation de phénomènes non linéaires mal connus.

On se rend compte, au travers des découvertes réalisées sur le cerveau ou à partir de simples observations sur le comportement humain ou animal, combien l'apprentissage est inhérent à l'intelligence. C'est cette phase d'apprentissage qui permet de « ranger » les informations perçues de telle manière qu'elles soient utilisables pour résoudre des problèmes nouveaux.

Qu'est-ce-que le neurone et les réseaux sont ? Comment fonctionne un réseau de neurones ? Par quelles méthodes apprend-il et s'améliore-t-il ? Dans quelles situations l'utilisation de réseaux de neurones s'avère utile ?

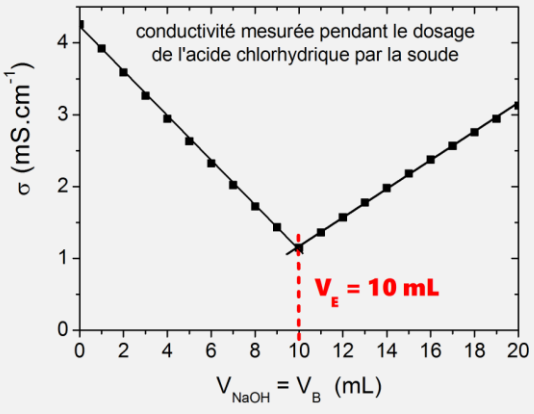
Une approche particulière

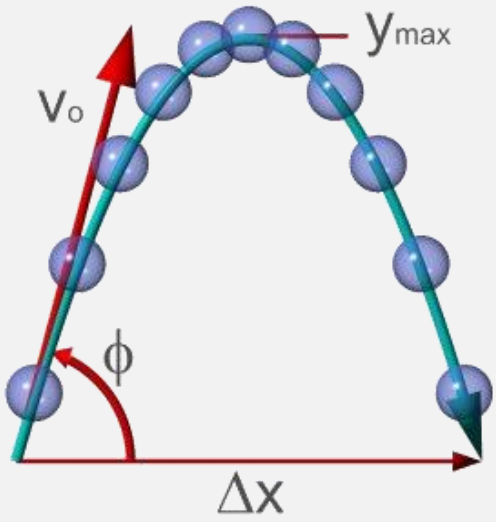
S'intéresser à la prévision, c'est chercher à créer des modèles. Cette recherche n'est pas un procédé réglé, plusieurs démarches se distinguent fondamentalement.

On peut, par exemple, chercher à théoriser le phénomène étudié, en déterminer les causes, et obtenir une description de ce phénomène, ainsi que de tous ceux qui lui sont liés. La physique théorique procède ainsi.

Mais on peut également se détacher de la théorie, et se contenter de créer un outil qui, à partir de la description d'une situation, délivre une nouvelle information sur cette situation. On peut obtenir un tel outil en se contentant de rectifier ses propriétés, de manière à ce que ce qu'il renvoie soit conforme à des observations expérimentales, ou à tout type de savoir acquis au préalable, sans rentrer dans la compréhension profonde du phénomène.

Certaines techniques de régression supposent connue « l'allure » de la solution, et ajustent l'outil après lui avoir conféré cette allure. Par exemple, les chimistes savent que la conductivité d'une solution évolue linéairement par rapport au volume de mélange titrant versé lors de chaque phase de dosage, et peut se contenter d'approcher la courbe expérimentale par une fonction affine par morceaux (l'outil) afin de récupérer des informations sur la solution. Le même procédé peut être utilisé pour étudier la trajectoire parabolique d'un corps en chute libre.





Ces techniques demandent malgré tout une connaissance partielle du problème étudié. Les réseaux de neurones proposent d'aller plus loin, et définissent une structure informatique générale, commune à toutes leurs applications, adaptable de manière tout aussi générale à chaque cas particulier.

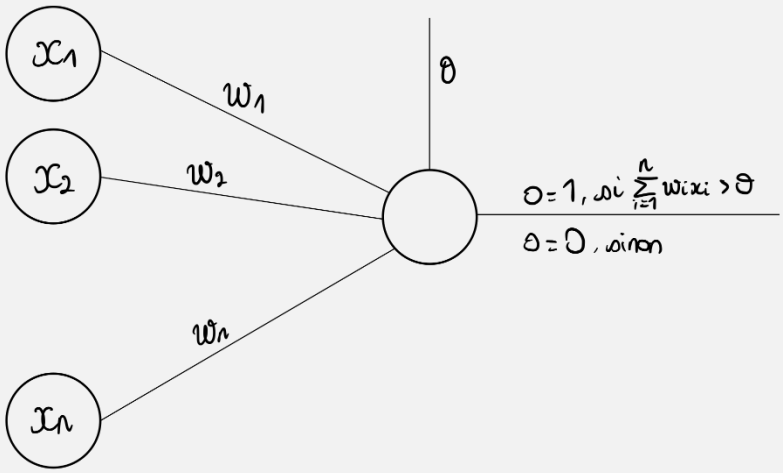
Historique des réseaux

Une brève étude de neurones humains permet de mieux comprendre la forme choisie pour les neurones artificiels, et leur mode de fonctionnement. En particulier, il est important de remarquer que :

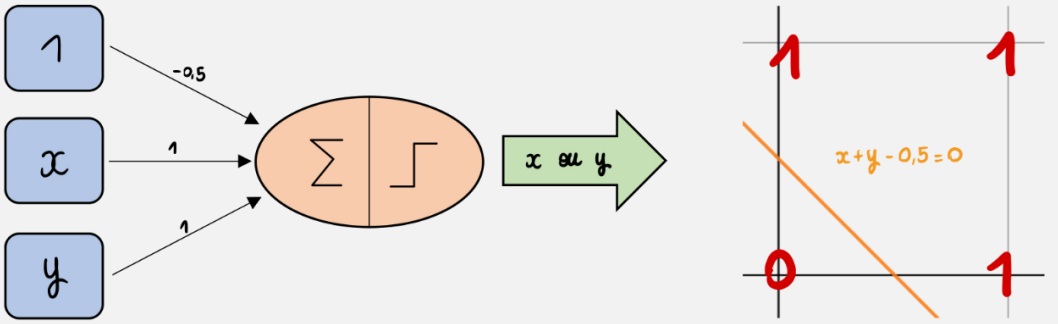
- L'information circule de neurone en neurone le long de synapses, sous forme d'un signal électrique.
- Les neurones humains s'activent s'ils sont stimulés au-delà d'un certain seuil ; ils ne s'activent pas sinon.
- Les connexions entre neurones sont initialisées aléatoirement à la naissance, c'est lors de la croissance et des contacts avec l'environnement que la plastique du cerveau prendra sa forme définitive.
- Les liaisons renforcées seront celles qui lient deux neurones activés simultanément ; les autres disparaissent (on parle de stabilisation effective).

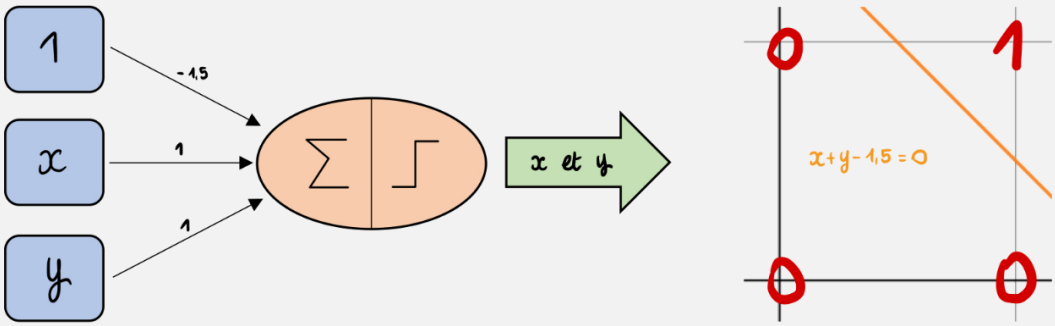
Les origines historiques des réseaux de neurones remontent au début des années 40. L'idée de base est alors de stimuler le fonctionnement d'un neurone biologique par des automates linéaires à seuils interconnectés. En 1943, McCulloch et Pitts réussissent à montrer que leurs modèles, composés d'automates à états finis, sont capables de calculer certaines fonctions logiques, ce qui fait naître l'idée d'ordinateur universel.

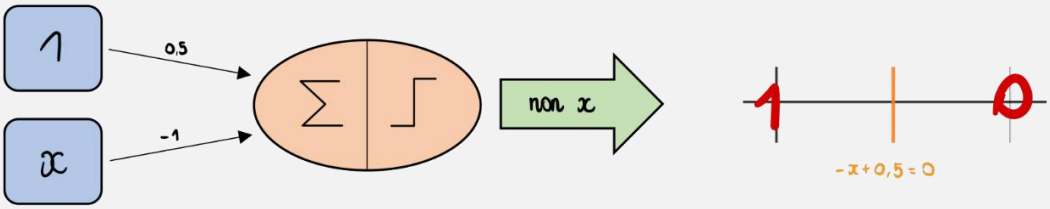
Il faut attendre 1958 pour voir la naissance, grâce à Frank Rosenblatt, d'une méthode analytique rigoureuse d'adaptation des poids au sein d'un modèle multicouches appelé perceptron : des entrées x_n sont combinées linéairement (chacune associée à un poids, dit synaptique, w_n). Le neurone renvoie 1 si la combinaison linéaire ainsi obtenue est strictement supérieure au seuil θ, 0 sinon. Le perceptron est un séparateur linéaire : il permet de séparer $\mathbb{R}^n$ en deux grâce à un hyperplan d'équation $\sum_{i=1}^{n} w_i x_i = \theta$.



Rosenblatt a démontré la convergence d'un algorithme itératif d'adaptation des poids pour les perceptrons monocouche. Héritant des idées précédemment décrites, le perceptron était en mesure d'apprendre n'importe quelle fonction logique à partir d'expériences grâce à la modification des poids synaptiques. Ces résultats n'ont pas manqué de susciter l'enthousiasme et de nourrir de grandes ambitions. La version présentée ci-dessous utilise la fonction de Heaviside (présentée plus loin), de seuil 0, et qui admet comme $n+1^e$ entrée la valeur 1 : le seuil effectif est alors l'opposé du poids associé à cette entrée. Chacun de ces neurones est associé à une division de $\mathbb{R}^2$ par la droite associée aux poids (pour le NON, $\mathbb{R}$ est séparé en deux par une valeur).

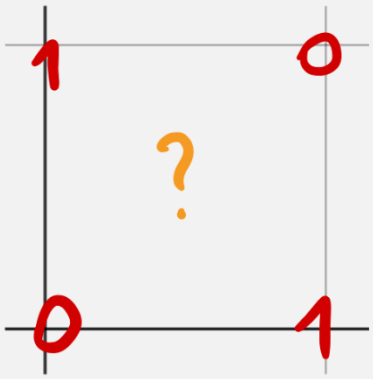




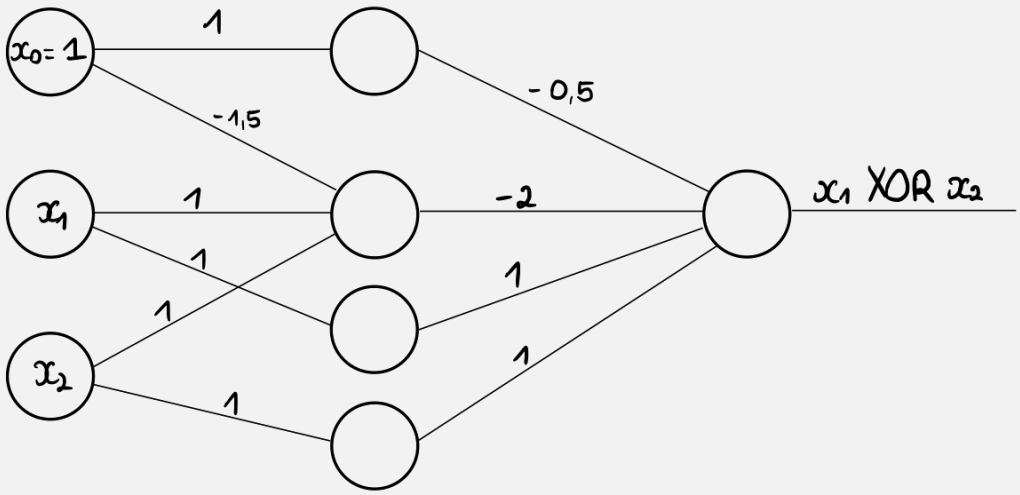


Néanmoins, les difficultés du perceptron à résoudre certains problèmes comme la modélisation du OU exclusif (XOR) furent rapidement mises en évidence. Supposons qu'il existe une droite d'équation $ax + by + c = 0$ permettant de discriminer les points (0,1) et (1,0) des points (0,0) et (1,1), et qu'elle les sépare de manière à ce que (0,1) et (1,0) soient du « côté strictement positif ». Alors

$$\begin{cases} a + c > 0 \\ b + c > 0 \\ a + b + c > 0 \\ c < 0 \end{cases} \Rightarrow \begin{cases} a + b + 2c > 0 \\ -a - b - c > 0 \end{cases} \Rightarrow c > 0$$



En ce sens, les travaux de Minski et Papert, en 1969, constituent une âpre critique des perceptrons, et semblent menacer l'avenir de la recherche en matière de neurones artificiels. Il n'en est rien, les réseaux de neurones avec couches cachées (1986) permettent de contourner le problème. Ces réseaux admettent des entrées, qui sont traitées par une première couche de neurones (dits neurones d'entrée), dont les sorties constituent les entrées des neurones suivants... et ainsi jusqu'à obtenir en dernière couche un problème linéairement séparable, qui sera alors résolu par les neurones de sortie. Le XOR peut être modélisé par un réseau à une couche cachée :

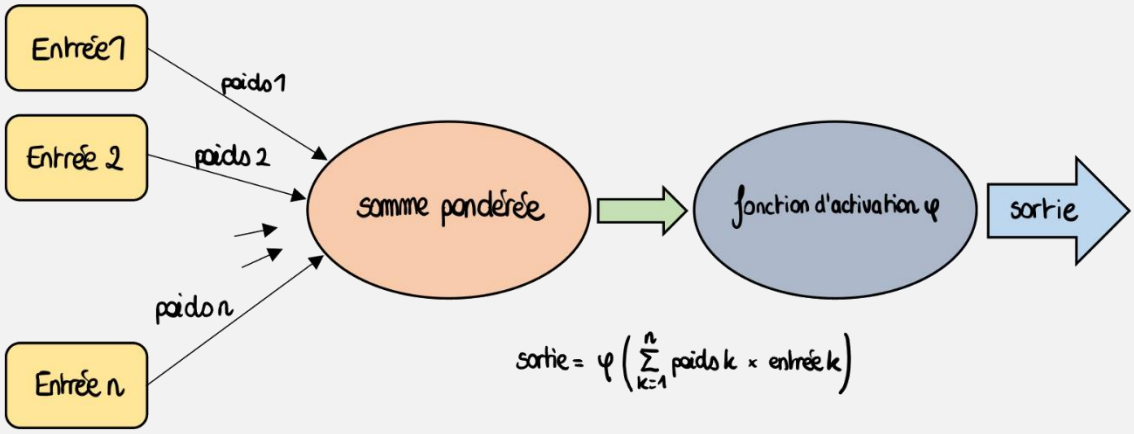


Ces réseaux permettent aux neurones d'être appliqués à de nombreux sujets, et d'être souvent efficaces. Voyons à présent comment rendre ces applications possibles.

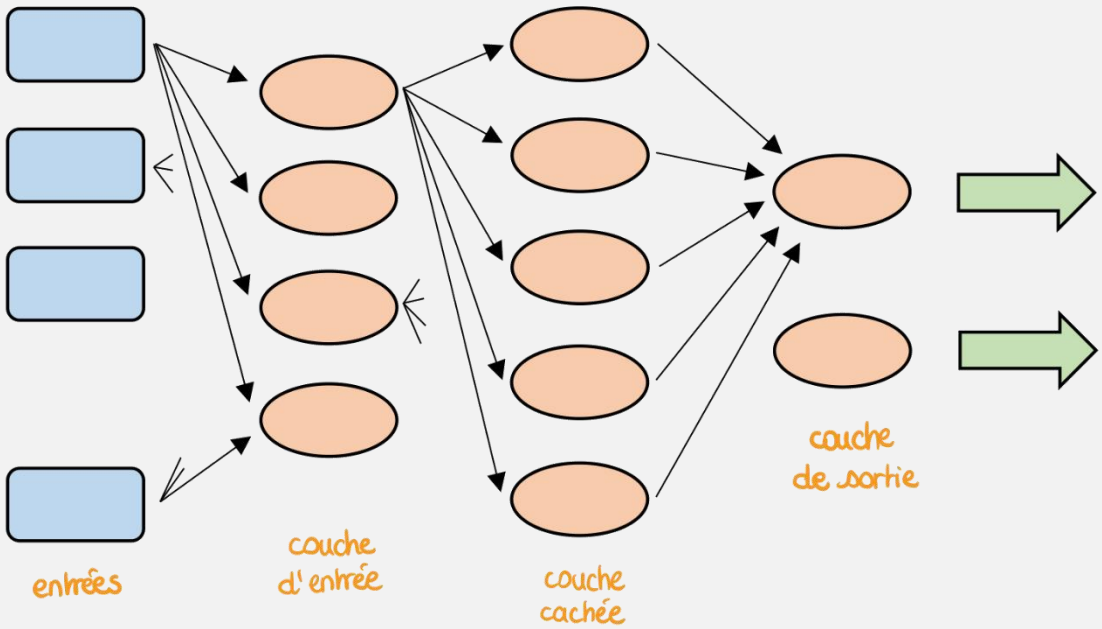
Présentation

❖ Le neurone formel

Un neurone représente un processus de traitement d'entrées numériques qui consiste à combiner linéairement ces entrées, chacune étant associée à un poids (on appelle état, ou activité, d'un neurone la somme ou le produit de ses entrées pondérées), puis à appliquer une fonction φ, dite d'activation, à cette combinaison linéaire. L'image obtenue par cette fonction constitue la sortie du neurone.



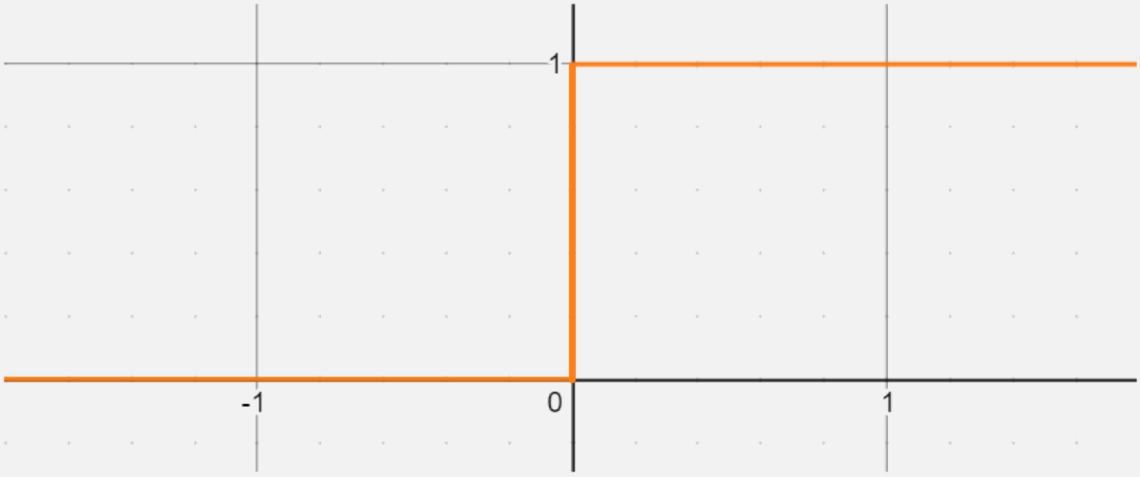
De tels neurones peuvent être agencés en cascade afin de former un réseau.



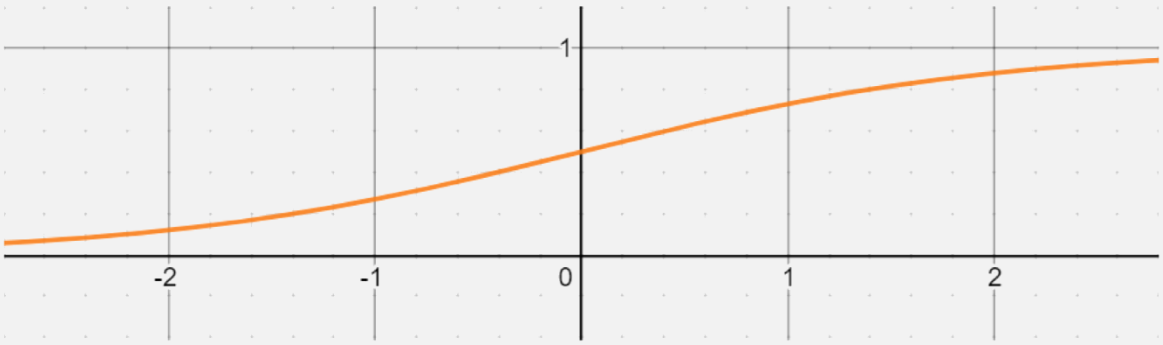
Le traitement effectué par chaque neurone demande l'utilisation d'une fonction d'activation. Celle-ci est de préférence non-linéaire, sinon le neurone ne se distingue pas d'une classique application linéaire (les fonctions d'activations sont généralement croissantes et bornées). Le rôle de cette fonction est donc de s'extraire de l'approximation linéaire, et de s'ouvrir à des horizons plus vastes. Les deux fonctions d'activation les plus utilisées sont :

- La fonction de Heaviside H, ou fonction seuil, correspondant à un neurone dont la sortie est binaire.
- La fonction sigmoïde S (la tangente hyperbolique modifiée : $\boldsymbol{S(x) = \frac{1+th(x)}{2} = \frac{1}{1+e^{-x}}}$), qui est la plus souvent employée du fait de sa caractéristique non linéaire et de sa dérivabilité continue sur le domaine ; c'est en fait la version « continue » (indéfiniment dérivable) de la fonction seuil.

Il existe d'autres fonctions employées, notamment les fonctions à base radiale, qui ne sont autres que des fonctions gaussiennes, mais elles ne seront pas étudiées.



Heaviside
H(x)=0, si x≤0
H(x)=1, si x>0

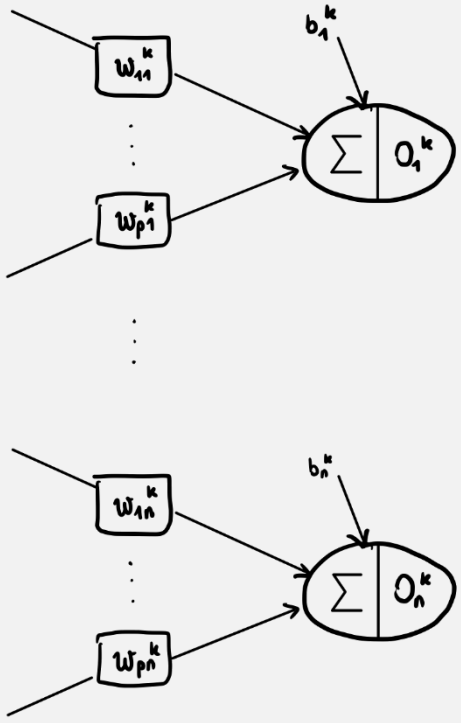


Sigmoïde : pour tout x, $S(x) = \frac{1}{1+e^{-x}}$

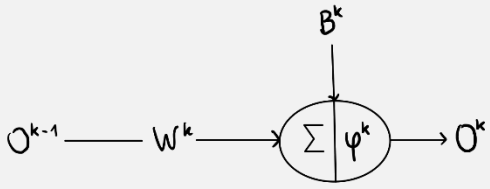
Chaque neurone utilisé dans un réseau se voit connecté à un biais, c'est-à-dire à une valeur constante, pour l'une de ses entrées. Ceci permet au neurone de translater son domaine d'activité et d'ajuster son seuil d'efficacité. L'effet de polarisation obtenu est indispensable si l'on veut exploiter correctement l'aptitude du réseau neuronal à reproduire n'importe quelle non-linéarité sur tout l'espace d'une univers donné.

❖ **Notion de couche**

On appelle couche un ensemble de n neurones possédant chacun p+1 entrées (p entrées + un biais). La figure ci-dessous représente une couche k composée de n neurones. Les équations suivantes expriment l'activité et la sortie d'un neurone j de cette couche : $x_j^k = \sum_{i=1}^{p} i_i w_{ij}^k + b_j^k$; $o_j^k = \varphi_j^k(x)$; avec : w_{ij}^k le poids reliant le neurone i de la couche k-1 au neurone j de la couche k, et b_j^k le biais du neurone j de la couche k.



De la même façon, on peut exprimer matriciellement la transformation des informations au sein d'une couche : $X^k = O^{k-1}W^k + B^k$; $O^k = \varphi^k(X^k)$

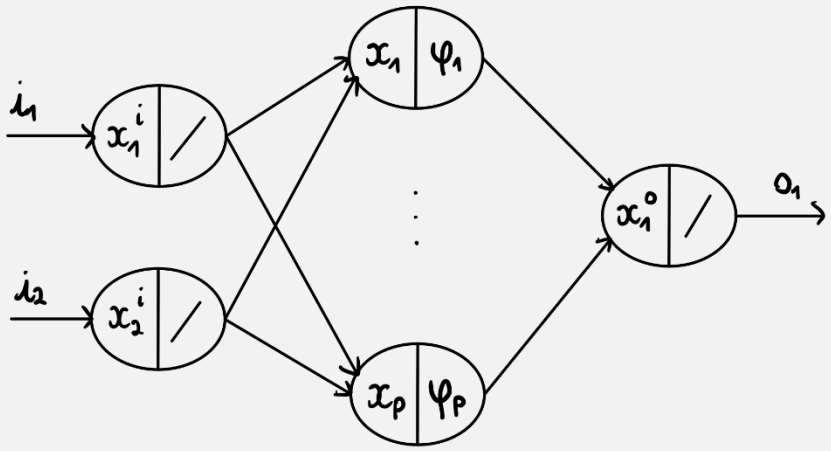


❖ **Réseau monocouche**

Chaque neurone est capable de reproduire sa propre fonction d'activation. La connexion en couches puis en réseau, va permettre de démultiplier leurs capacités en profitant des propriétés de toutes les fonctions d'activation. Ainsi, par ajustement des poids et des biais, qui sont autant de degré de liberté, le réseau sera capable d'approcher n'importe quelle fonction.

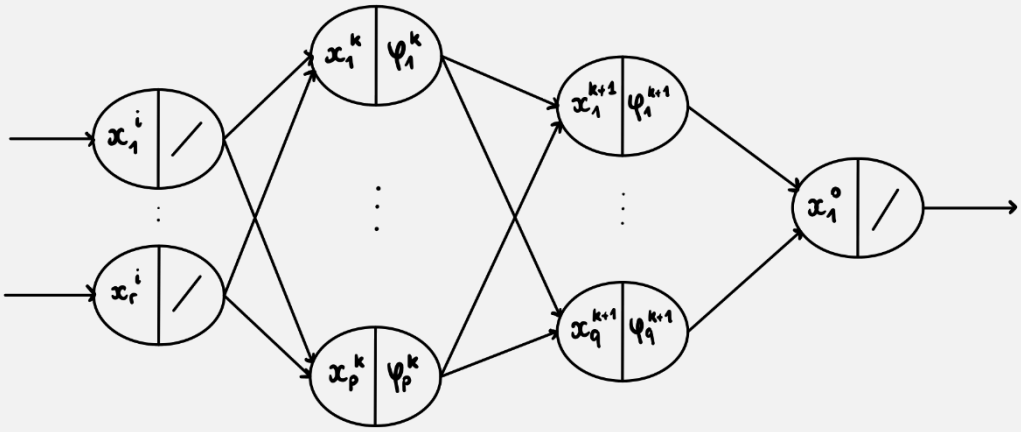
On peut donc considérer le réseau de neurones comme « une boîte noire » capable de synthétiser n'importe quelles sorties fonctions d'entrées. Il paraît alors naturel de créer un réseau dont le flux d'informations transite de l'entrée vers la sortie ; c'est pourquoi on organise une architecture dite à couches. En disposant les neurones en plusieurs couches, nous construisons un réseau de type *feedforward*, dans lequel les connexions n'existent que d'une couche vers la suivante ; le système ainsi défini est donc statique. Au sein de celui-ci, les neurones pourront être différents d'une couche à l'autre, ainsi qu'à l'intérieur d'une même couche. Les entrées et les sorties sont pourvues de neurones à fonctions d'activation linéaires pour respectivement distribuer et récolter les informations. Lorsque l'état d'un neurone est formé par la somme des entrées, on le symbolise par un signe somme. De même, pour les fonctions d'activation, on représente la fonction soit par une lettre, soit par un dessins décrivant la fonction.

Sur la représentation naturelle (figure II.4.2), les poids et les biais n'ont pas été représentés par souci de clarté. Les neurones de la couche d'entrée et de sortie ne servent respectivement qu'à distribuer et à recueillir les informations. C'est pour cette raison que les fonctions d'activation de ces neurones sont toujours linéaires. De plus la couche d'entrée, qui n'a qu'un rôle de distribution, possède des poids unitaire et aucun décalage, les biais sont donc nuls. Par ailleurs, on appelle également couche cachée toute couche n'appartenant ni à la couche d'entrée ni à la couche de sortie.



❖ **Réseau multicouche**

La qualité d'émulation augmente généralement avec le nombre de neurones. Dans le cas des réseaux monocouches, la recherche de l'amélioration conduit à augmenter l'unique couche cachée. Afin d'éviter une hypertrophie de cette couche, on utilise généralement des réseaux possédant plusieurs couches cachées. Il a en effet été montré que la qualité d'approche se renforce avec l'adjonction de couches cachées supplémentaires.



Nous définissons pour chaque neurone i de la couche k l'état x_i^k et la sortie o_i^k. Les couches k et k+1 comportent respectivement p et q neurones, nous avons les équations : $x_i^{k+1} = \sum_{j=1}^{p} o_j^k w_{ji}^{k+1} + b_i^{k+1}$; $o_i^{k+1} = \varphi_i^{k+1}(x_i^{k+1})$; avec : w_{ji}^{k+1} le poids allant du neurone j couche k au neurone i couche k+1, et b_i^{k+1} le poids de l'entrée de polarisation (biais). Il s'en déduit l'expression matricielle du comportement de la couche k+1, en notant X^{k+1} et O^{k+1} les vecteurs des états et des sorties des neurones de la couche k+1 : $X^{k+1} = O^k W^{k+1} + B^{k+1}$; $O^{k+1} = \varphi^{k+1}(X^{k+1})$; avec les matrices suivantes :

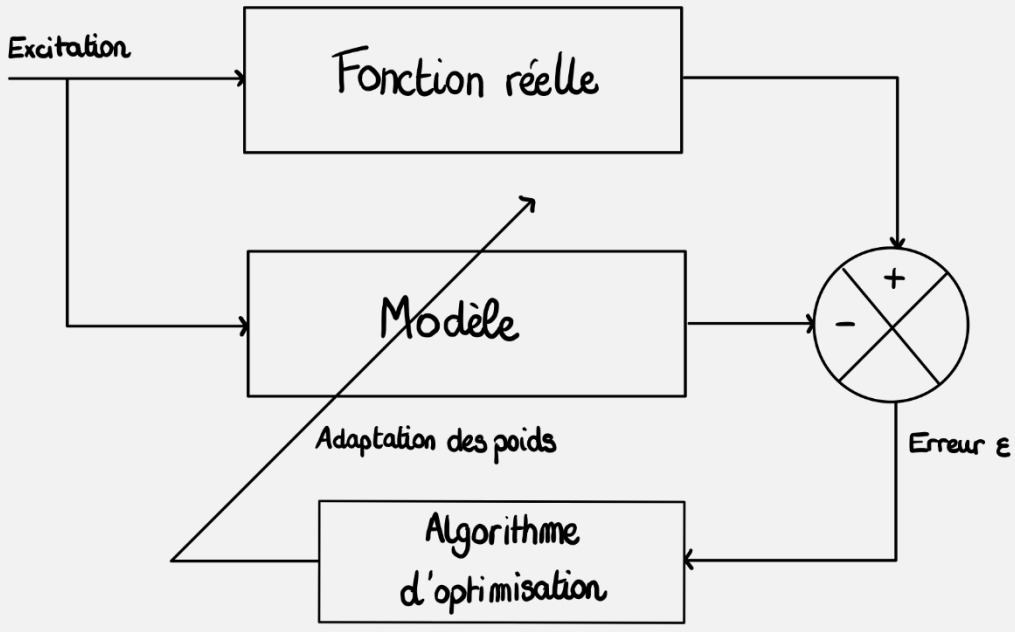
$$W^{k+1} = \begin{bmatrix} w_{1,1}^{k+1} & \cdots & w_{1,q}^{k+1} \\ \vdots & \ddots & \vdots \\ w_{p,1}^{k+1} & \cdots & w_{p,q}^{k+1} \end{bmatrix} ; B^{k+1} = \begin{bmatrix} b_1^{k+1} \\ \vdots \\ b_q^{k+1} \end{bmatrix} ; \varphi^{k+1}(X^{k+1}) = \begin{bmatrix} \varphi_1^{k+1}(x_1^{k+1}) \\ \vdots \\ \varphi_q^{k+1}(x_q^{k+1}) \end{bmatrix}$$

A partir des expressions précédentes, il est facile de décrire le cas général d'un réseau à N couches sous la forme matricielle suivante : $O = \varphi^N(((.....(X^1W^1 + B^1).....)W^{N-1} + B^{N-1})W^N + B^N)$.

Tout problème résolu à l'aide de neurones utilisera ce type d'architecture, et même souvent l'une de ces deux fonctions d'activation. L'exemple des fonctions booléennes ET et OU nous montre qu'une même allure de structure peut coder deux fonctions différentes, seuls les poids synaptiques permettent de discriminer les deux réseaux. Etudions maintenant la manière d'adapter ces réseaux, et ces poids, à chaque problème abordé.

Généralité de la démarche

L'intérêt d'un réseau de neurones réside dans sa capacité à composer avec ses fonctions d'activation pour approcher n'importe quelle fonction, linéaire ou non, à partir d'échantillons expérimentaux. Les poids et les biais sont autant de paramètres à régler, dont une seule combinaison offrira une approche optimale au sens des moindres carrés. En effet, on utilise généralement la somme des écarts quadratiques comme critère d'ajustement, d'une part car ce dernier ne peut pas s'annuler par simple retranchement des erreurs et, d'autre part, la somme des écarts quadratiques possède des propriétés statistiques liées à la définition même de la variance. Afin d'aboutir à cette synthèse, il est nécessaire d'effectuer une phase dite « d'apprentissage », qui consiste à trouver les poids et les biais optimaux pour une architecture de réseau donné. Comme les fonctions d'activation et les fonctions à approcher sont généralement de nature non linéaire, les méthodes employées s'inspirent des méthodes d'optimisation non linéaires. Les plus connues sont le recuit simulé, les algorithmes génétiques, la méthode des directions conjuguées et la descente de gradient. L'apprentissage est réalisé suivant le schéma ci-dessous, de sorte que l'erreur ε entre la fonction réelle et la fonction estimée soit réduite au maximum ; on parle dans ce cas d'apprentissage supervisé, l'algorithme d'optimisation étant alors le superviseur.



Ainsi, on décompose l'ensemble global d'échantillons en deux sous-ensembles : celui des échantillons d'apprentissage, qui sert à l'ajustement des poids ; et celui des échantillons de validité, qui permet de tester les performances de l'apprentissage obtenues sur le même domaine de modélisation.

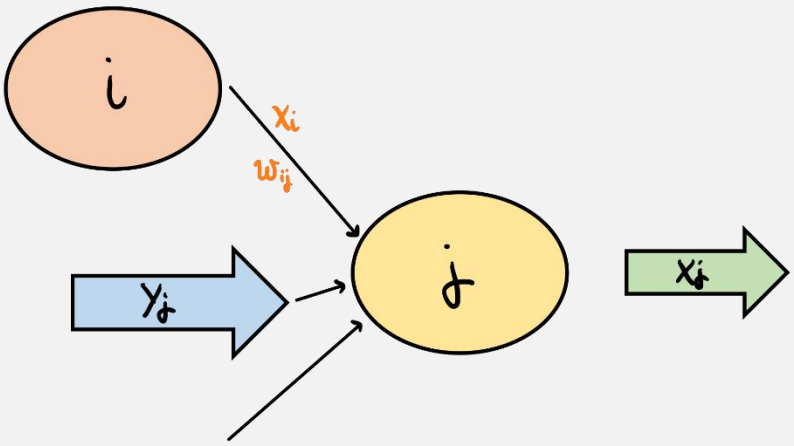
Etudions le principe de rétropropagation du gradient. Considérons un réseau comprenant N entrées et M sorties. Au cours de cette procédure d'apprentissage, un vecteur $I=\{I_0,I_1,...,I_N\}$ est présenté à l'entrée du réseau ; le vecteur de sortie $X=\{X_1,...,X_M\}$ qui en résulte est alors différent de celui désiré, soit $T=\{T_1,...,T_M\}$. Chaque composante I_i, X_i ou T_i est un vecteur composé de n échantillons. Un vecteur erreur E entre les vecteur T et X est introduit et les poids sont ajustés de façon à minimaliser l'erreur globale au sens des moindres carrés : $E = \frac{1}{2}\sum_{i=1}^{M}(X_i - T_i)^2$. La minimisation de cette erreur traduit l'amélioration des performances du réseau relativement à l'échantillon d'exemples.

On a vu que seuls les poids synaptiques déterminent ce que code le réseau. On va donc chercher à modifier chaque poids w pour diminuer l'erreur, et ceci en suivant la dérivée de la fonction d'une variable E(w), dérivable si on choisit des fonctions d'activations dérivables. Le signe de la dérivée indique le sens d'évolution du poids qui entrainera une diminution de l'erreur. C'est la démarche de descente du gradient.

L'erreur est en réalité une fonction de tous les poids, et des entrées, mais les cas d'application des réseaux permettent de supposer une certaine régularité de la fonction erreur, et le suivi des dérivées partielles ne s'oppose pas à la recherche du minimum global (bien que cette démarche ne garantisse pas la convergence vers ce minimum).

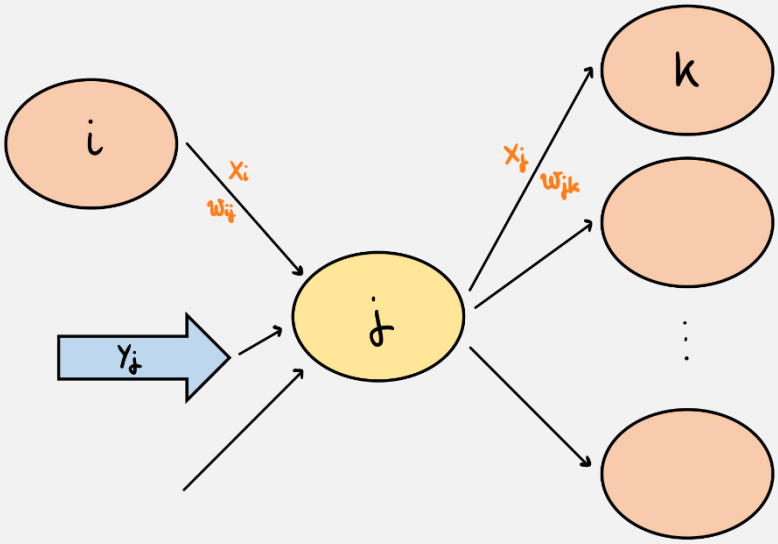
L'algorithme de rétropropagation du gradient définit chaque nouveau poids en fonction de l'ancien : $w_{ij}(new) = w_{ij}(old) - \mu \frac{\partial E}{\partial wij(old)}$; μ étant le coefficient d'apprentissage (plus μ est grand, plus la modification des poids se fera rapidement), w_{ij} étant le poids allant du neurone i, de sortie X_i, au neurone j, de sortie X_j.

- Si le neurone j est un neurone de sortie :



L'erreur est directement reliée à X_j, un des termes de la somme étant $\frac{1}{2}(X_j - T_j)^2$. Le poids w_{ij} est relié à la sortie X_j selon $X_j = \varphi(w_{ij}X_i + \dots)$, notons $Y_j = w_{ij}X_i + \dots$ la combinaison linéaire entrant dans le neurone j. Donc, en utilisant la sigmoïde comme fonction d'activation, sa dérivée étant $S'(x) = S(x)(1 - S(x))$, et comme $X_j = S(Y_j)$, on peut calculer la dérivée de l'erreur en fonction du poids w_{ij} : $\frac{\partial E}{\partial wij} = \frac{\partial E}{\partial Yj}\frac{\partial Yj}{\partial wij} = S'(Y_j)(X_j - T_j)X_i = X_j(1 - X_j)(X_j - T_j)X_i$.

- Si le neurone j n'est pas un neurone de sortie :

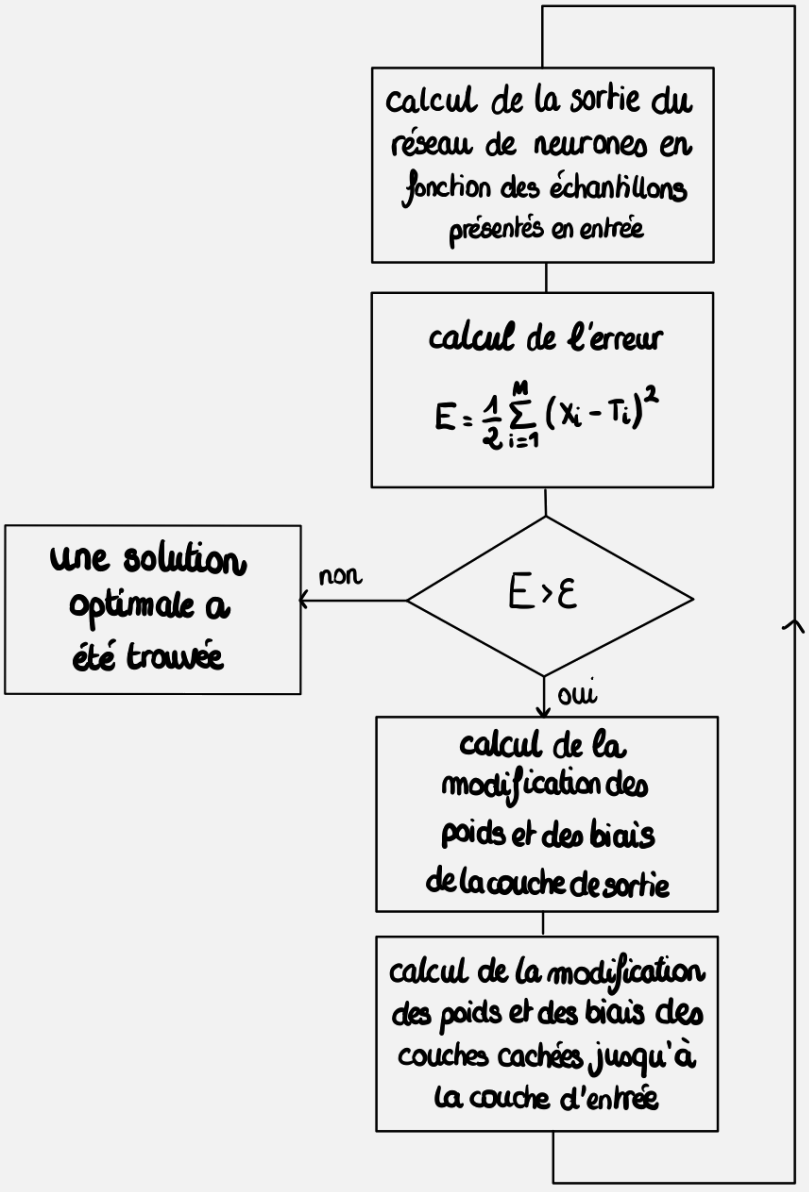


L'impact de Y_j sur l'erreur, $\frac{\partial E}{\partial Yj}$, s'exprime au travers de la récupération de cette sortie en tant qu'entrée des neurones k de la couche suivante, avec $Y_k = w_{jk}X_j + \dots = w_{jk}S(Y_j) + \dots$:

$$\frac{\partial E}{\partial wij} = \frac{\partial E}{\partial Yj}\frac{\partial Yj}{\partial wij} = X_i\sum_{k=1}^{n}\frac{\partial E}{\partial Yk}\frac{\partial Yk}{\partial Yj} = X_i\sum_{k=1}^{n}\frac{\partial E}{\partial Yk}(w_{jk}X_j(1 - X_j))$$

Ainsi, on calcule la variation d'erreur en fonction de chaque poids. On voit que le calcul des $\frac{\partial E}{\partial Yj}$ sera d'abord effectué en couche de sortie, puis les valeurs obtenues se propageront en amont du réseau, pour déterminer les suivantes. C'est pourquoi cet algorithme porte de le nom de rétropropagation du gradient de l'erreur.

Il suffit maintenant de modifier le poids dans le sens de la diminution de l'erreur : $\Delta w_{ij} = -\mu \frac{\partial E}{\partial wij}$. Une fois que tous les poids et les biais ont été modifiés, on représente à nouveau le vecteur I en entrée et on recalcule le vecteur en sortie du réseau. Si l'erreur E est toujours jugée trop grande, on réitère la procédure d'ajustement précédemment décrite, jusqu'à ce que E respecte de critère de précision fixé. Le réseau a alors appris, il est performant par rapport à l'échantillon d'exemples.



Exemple

On reprend la généralité de la démarche :

On voudrait adapter un réseau de neurones de manière à ce qu'il approxime une fonction choisie. Par exemple, en hydrologie, la fonction qui, à la donnée de la description météorologique d'une suite de jours (température, pression, hauteur de précipitations…) associe le débit d'une rivière pour le jour suivant.

Toute la construction du « bon » réseau sera guidée par un principe phare : on s'appuie sur un échantillon d'exemples, dont on connaît les caractéristiques (les variables météorologiques d'entrée du réseau, ainsi que le débit effectif, que le réseau devrait déterminer), puis on modifie les poids synaptiques de manière à ce que les sorties du réseau soient conformes aux valeurs connues (du débit).

On a parlé d'entrées, de variables, en supposant connaître quelles sont les grandeurs susceptibles de caractériser la situation étudiée. Si déterminer les grandeurs pertinentes n'est pas immédiat, on peut utiliser un réseau de neurones pour cela, en se contentant de faire une étude de l'erreur relative à un échantillon d'exemples, sans étudier le problème dans ses particularités.

On réunit tous les candidats-variables (on retient les températures quotidiennes moyennes, maximales et minimales, la pression, la hauteur des précipitations, le mois de l'année et le débit des six jours précédant le jour dont le débit est à déterminer), puis on demande au réseau d'apprendre un échantillon d'exemples, forts de ces variables exhaustives. On applique le réseau ainsi entraîné à un échantillon test, qu'il n'a pas appris, et on relève l'erreur relative à cet échantillon. Puis on réitère ce processus en écartant une variable descriptive des exemples (en conservant toutes les autres), et on récupère la nouvelle erreur. Si elle est moindre que l'erreur initiale, cette variable avait fourni au réseau des informations qui l'avaient empêché de conclure correctement : elle nuit à la compréhension du problème. Si l'erreur est sensiblement inchangée, cette variable n'a pas d'impact sur les performances du réseau. Si l'erreur est fortement augmentée, cela signifie que la variable est indispensable à la modélisation, elle est pertinente.

Pour l'hydrologie, cette analyse mène aux résultats suivants, pour un apprentissage sur un an, et un test sur une autre année (données de la station Bourges-Le Moulon, années 2000-2001) :

Variable retirée	Aucune	T_{min}	T_{moy}	T_{max}	Pression	Hauteur des précipitations	Mois	Débit des jours précédents
Erreur	8,75	9,3	9,3	9,42	9,22	15,9	9,01	12,6

Ainsi, sans être inutiles, les variables températures, pression et mois ne sont pas déterminantes pour le débit, alors que les précipitations et le débit des jours précédents sont indispensables.

Enfin, bien que l'allure de tout réseau soit constante, il est nécessaire de déterminer le nombre de couches, et de neurones qui constitueront le réseau optimal. Encore une fois, la seule analyse de l'erreur suffit à répondre à ces questions. On essaie tour à tour diverses structures, et on sélectionne celle qui conduit à la plus faible erreur.

Dans le cas de l'hydrologie, en se contentant de garder le même nombre de neurones dans chaque couche cachée (mêmes conditions d'apprentissage que pour la sélection des variables) :

Erreur	Neurone	3	4	5	6	7	8	9
Couche								
2		77,01	77,7	99,46	86,23	59,29	204,3	217,5
3		33,62	28,07	25,99	18,19	15,37	14,95	51,64
4		33,83	24,77	21,54	16,26	12,58	11,47	18,09
5		33,53	24,36	21,15	15,89	12,29	11,03	17,68
6		33,41	24,29	21,1	15,85	12,25	11,03	17,68
7		33,41	24,29	21,1	15,85	12,25	11,03	17,68
8							11,03	
9							11,03	

La structure sélectionnée sera donc celle qui présente 5 couches cachées, chacune comportant 8 neurones. Avoir plus de couches n'améliore pas la performance, mais allonge le temps de calcul, et avoir plus de neurones détériore même les capacités du réseau.

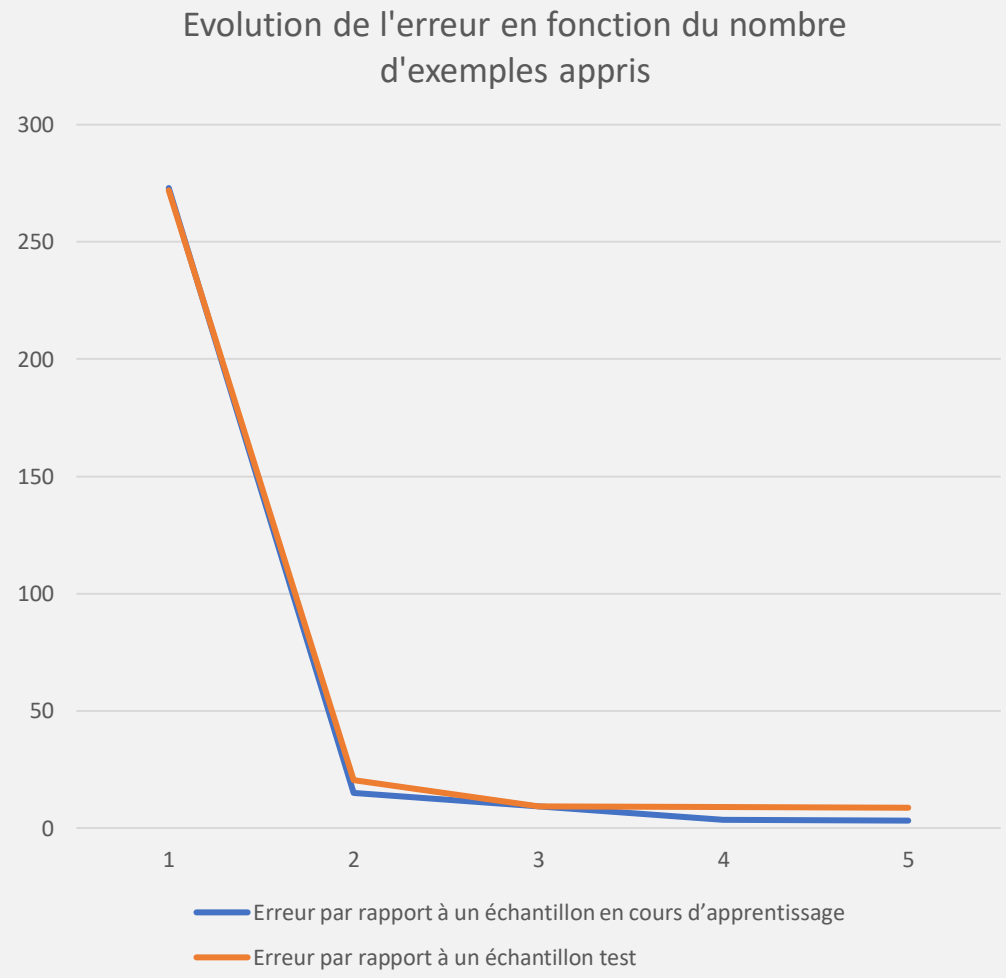
Ainsi, le principe fondateur de la démarche des réseaux de neurones, la diminution de l'erreur locale, relative à un échantillon d'exemples, permet la construction intégrale du réseau. Mais cette construction est-elle vraiment efficace ?

Efficacité des réseaux

Commentons d'abord le pari de parvenir à créer un modèle fonctionnel à partir d'un ensemble limité d'exemples. Les problèmes physiques d'application des neurones présentent une certaine régularité, ce qui permet à un nombre restreint d'exemples d'être représentatifs.

En ce qui concerne l'hydrologie, l'erreur par rapport à un échantillon test suit la même évolution temporelle que l'erreur par rapport à un échantillon en cours d'apprentissage. Les valeurs ont été révélées tous les 10000 exemples appris (l'année 2000 apprise en boucle).

Erreur par rapport à un échantillon en cours d'apprentissage	273	14,84	9,22	3,51	3,13
Erreur par rapport à un échantillon test	272	20,47	9,27	8,97	8,63



Le réseau, en se contentant d'assimiler l'information de la première année, gagne aussi en efficacité par rapport à la deuxième. La limitation de l'apprentissage est justifiée.

Enfin, utilisons de nouveau l'exemple de l'hydrologie pour montrer l'efficacité des réseaux de neurones : après apprentissage de la première moitié de l'année 2000, en se limitant à la variable précipitations, l'évaluation de l'écart relatif moyen entre sortie du réseau et débit enregistré est de 25% sur l'ensemble de l'année 2000, et de 32% sur l'année 2001. Cet écart permet de séparer les débits forts des débits faibles, et rend approximativement compte de ce que sera le débit de la rivière le lendemain.

Une approche légitime

Les réseaux de neurones, engendrés par la biologie, sont un outil d'approximation universelle : le principe qui guide leur construction se veut extrêmement général, adaptable à toute situation. L'implémentation informatique de ces réseaux est d'ailleurs la même quelle que soit l'utilisation envisagée, seule la récupération de l'information présente ses spécificités. L'étude des neurones est donc absolue.

Par ailleurs, la démarche présentée prouve par ses succès la compatibilité entre approche universelle et efficacité pratique. Les applications des neurones sont nombreuses : on peut par exemple coder toute fonction booléenne (sous sa forme normale disjonctive, puisqu'on connaît des réseaux codant le ET, le OU et le NON) avec un perceptron multicouches. Les extensions à tout problème qualitatif (dépistage de maladies, prévention de risques...) sont immédiates.

On peut reprocher aux réseaux de neurones d'être insondables : les poids, les couches ne permettent pas de déterminer comment le réseau parvient à « comprendre » le phénomène qu'il modélise, c'est l'effet boîte noire. Malgré cela, l'utilisation des neurones permet de résoudre des problèmes de manière peu conventionnelle, donc apporte un éclairage intéressant.