Intelligence Naturelle et Intelligence Artificielle

Natural Intelligence and Artificial intelligence

Daniel M. Dubois

Ingénieur Physicien, Docteur en Sciences Appliquées, Dr Honoris Causa,
Professeur d'Informatique Appliquée et Intelligence Artificielle,
HEC®-Ecole de Gestion de l'Université de Liège N1, 14 rue Louvrex, B-4000 Liège, Belgique
Directeur de l'asbl CHAOS : Centre for Hyperincursion and Anticipation in Ordered Systems,
Institut de Mathématique B37, Université de Liège, Grande Traverse 12, B-4000 Liège 1,
Belgique
http://www.ulg.ac.be/mathgen/CHAOS, Daniel.Dubois@ulg.ac.be
GSM: +32495510419, SKYPE: d.m.dubois

Résumé *: Cet article présente une approche systémique du concept d'intelligence naturelle en ayant pour objectif de créer une intelligence artificielle. Ainsi, l'intelligence naturelle, humaine et animale non-humaine, est une fonction composée de facultés permettant de connaître et de comprendre. De plus, l'intelligence naturelle reste indissociable de la structure, à savoir les organes du cerveau et du corps. La tentation est grande de doter les systèmes informatiques d'une intelligence artificielle qui devrait avoir une fonction composée des mêmes facultés que celles englobées dans l'intelligence humaine. L'intelligence artificielle doit aussi rester indissociable de la structure, à savoir les organes d'un cerveau artificiel et d'un corps artificiel. En conséquence de quoi, le robot artificiel est la structure idéale pour voir l'émergence d'une intelligence artificielle forte. Notez qu'une intelligence artificielle faible se développe sur des systèmes informatiques par un programme de simulation de l'intelligence. Dans cette communication, des mémoires neuronales programmables seront présentées et simulées.*

Mots-clés : intelligence naturelle, intelligence artificielle, robot, mémoire neuronale programmable

Abstract : *This article presents a systemic approach to the concept of natural intelligence with the aim of creating an artificial intelligence. Thus, the natural intelligence, human and nonhuman animals, is a function composed of faculties to know and understand. In addition, natural intelligence is inseparable from the structure, namely the organs of the brain and body. The temptation is to equip computer systems with artificial intelligence that should have a function composed of the same faculties as those encompassed in human intelligence. The artificial intelligence must also remain inseparable from the structure, namely the organs of an artificial brain and an artificial body. As a result, the artificial robot is the ideal structure to see the emergence of a strong artificial intelligence. Note that a weak artificial intelligence is developed on computer systems by a simulation program of intelligence. In this paper, neural programmable memories will be presented and simulated.*

Key words : natural intelligence, artificial intelligence, robot, neural programmable memory

1. Introduction

L'intelligence revêt diverses acceptions selon le fait qu'on l'envisage d'un point de vue purement général, étymologique, informatique ou psychologique, voire philosophique. Dans un premier temps, intéressons-nous à l'étymologie du terme : du latin « intellegentia (dér. de intellegere « comprendre ») », il s'agit essentiellement de la « faculté de comprendre » L'intelligence est une « fonction mentale d'organisation du réel en pensées chez l'être humain, en actes chez l'être humain et l'animal ». L'intelligence est un « ensemble des fonctions psychiques et psycho-physiologiques concourant à la connaissance, à la compréhension de la nature des choses et de la signification des

faits; faculté de connaître et de comprendre », et donc lie le corps à cette fonction mentale qui permet de connaître et de comprendre. Ainsi, l'intelligence naturelle, humaine et animale non-humaine, est une fonction composée de facultés à la fois mentale, psychique et psychophysiologique permettant de connaître et de comprendre. De plus, l'intelligence naturelle reste indissociable de la structure, à savoir les organes du cerveau et du corps.

En logique informatique, l'intelligence « artificielle » est désignée comme une « recherche de moyens susceptibles de doter les systèmes informatiques de capacités intellectuelles comparables à celles des êtres humains ». Dans le cadre des systèmes informatiques à la robotique, l'intelligence artificielle prétend être une fonction composée des mêmes facultés que celles englobées dans l'intelligence humaine. L'intelligence artificielle doit donc aussi rester indissociable de la structure, à savoir les organes d'un cerveau artificiel et d'un corps artificiel. En conséquence de quoi, le robot artificiel est la structure idéale pour voir l'émergence d'une intelligence artificielle forte. Notez qu'une intelligence artificielle faible se développe sur des systèmes informatiques par un programme de simulation de l'intelligence.

Que faudrait-il mettre en place pour avoir un « robot intelligent» ? La question est éminemment compliquée dans le contexte actuel des avancées en matière d'intelligence artificielle. Elles sont certes encourageantes, mais encore malheureusement insuffisantes pour pouvoir prétendre intégrer cette propriété à un robot. Comme je le précise dans mon article (Daniel M. Dubois, 2010) : « Lorsqu'une machine sera capable de comprendre notre langage et de dialoguer avec nous, alors une intelligence consciente artificielle émergera naturellement de la machine. » En effet, l'un des concepts clés de l'intelligence est la notion de langage. En « milieu naturel », c'est ce qui nous permet de raisonner, d'imaginer et donc de nous projeter dans l'avenir, autrement dit, d'anticiper. En « milieu artificiel », c'est ce qui permet d'exécuter une action spécifique via un programme codé. Qu'elle différence et ressemblance y-a-il entre ce langage naturel et le langage artificiel de programmation en informatique, et plus particulièrement en Intelligence Artificielle ? Cette question pose la question fondamentale suivante : Les chercheurs en linguistique et les chercheurs en informatique ont-ils compris ce qu'était réellement un langage ? Si c'était le cas, nous pourrions parler à notre ordinateur dans notre langue maternelle pour lui demander des tâches à accomplir. Au lieu de cela, l'intelligence naturelle a conçu un langage intermédiaire pour dialoguer avec les machines, des ordinateurs aux robots. Il y a encore beaucoup de recherche à faire pour comprendre ce qu'est un langage naturel et comment un langage artificiel pourrait servir de véritable dialogue entre robots et entre les robots et l'homme. C'est un des challenges pour le XX1e siècle.

Comment l'homme fait-il pour raisonner, parler, calculer, apprendre ? Deux types d'approches ont été essentiellement explorés.

D'une part, l'analyse logique des tâches relevant de la cognition humaine est reconstituée par programme. C'est cette approche qui a été privilégiée par la psychologie cognitive classique, et est étiquetée sous le nom de **cognitivisme**.

D'autre part, puisque la pensée est produite par le cerveau, on a étudié son fonctionnement. La physiologie du cerveau montre que celui-ci est constitué de cellules (les *neurones*) interconnectées. Le cerveau contient environ 100 milliards de neurones. On ne dénombre que quelques dizaines de catégories distinctes de neurones et aucune catégorie de neurones n'est propre à l'homme. Les neurones reçoivent les signaux (impulsions électriques) par des extensions très ramifiées de leur corps cellulaire (les dendrites) et envoient l'information par de longs prolongements (les axones). Les impulsions électriques sont régénérées pendant le parcours le long de l'axone. La durée de chaque impulsion est de l'ordre d'une milli-seconde et son amplitude d'environ cent milli-volts. La vitesse de propagation des influx nerveux est de l'ordre de 100 mètre/seconde, c'est-à-dire bien inférieure à la vitesse de transmission de l'information dans un circuit électronique. Les contacts entre deux neurones, de l'axone à une dendrite, se font par l'intermédiaire des synapses. Lorsqu'un potentiel d'action atteint la terminaison d'un axone, des neuromédiateurs sont libérés et se lient à des récepteurs post-synaptiques présents sur les dendrites. L'effet peut être excitateur ou inhibiteur. On compte quelques centaines à plusieurs dizaines de milliers de synapses par neurone. Chaque neurone intègre en permanence jusqu'à un millier de signaux synaptiques. Ces signaux n'opèrent pas

de manière linéaire (effet de seuil). Le nombre total de connexions est ainsi estimé à environ 10^{15}. La connectique du cerveau ne semble pas être codée par l'ADN et la structure du cerveau provient donc en partie des informations de l'environnement par apprentissage. Il semble que l'apprentissage se fasse par un double mécanisme : des connections sont établies de manière redondantes et aléatoires puis seules les connexions entre des neurones simultanément actifs sont conservés (phase de sélection) tandis que les autres sont éliminés. C'est cette approche qui a conduit à l'étude de réseaux de neurones formels. On désigne par **connexionisme** la démarche consistant à vouloir rendre compte de la cognition humaine par des réseaux de neurones. Cette seconde approche a donc mené à la définition et l'étude de réseaux de neurones formels qui sont des réseaux complexes d'unités de calcul élémentaire interconnectées.

Il existe deux courants de recherche sur les réseaux de neurones.

D'une part, l'étude et la modélisation des phénomènes naturels d'apprentissage à l'aide de réseaux de neurones où la pertinence biologique est importante.

D'autre part, l'obtention d'algorithmes efficaces ne se préoccupant pas de la pertinence biologique. Bien que les réseaux de neurones formels aient été définis à partir de considérations biologiques, pour la plupart d'entre eux, de nombreuses caractéristiques biologiques (le temps, la mémoire) ne sont pas prises en compte.

Toutefois, nous allons donner un bref aperçu de quelques propriétés élémentaires de neurophysiologie qui permettent de relier les neurones formels aux neurones réels. Nous donnons ensuite un bref historique des réseaux de neurones. Enfin, nous donnons une classification des différents types de réseau et les principales applications.

2. Bref historique des réseaux de neurones artificiels

Depuis 1940, deux tendances se sont dégagées pour réaliser des machines artificielles à l'image de l'intelligence humaine. D'une part, les ordinateurs digitaux exécutant des programmes enregistrés à partir d'algorithmes. Un algorithme est une suite ordonnée d'instructions que l'ordinateur exécute dans l'ordre. D'autre part, les réseaux de neurones artificiels pour l'apprentissage d'opérations de manière non-algorithmique et sans programmation d'instructions. A cela, les réseaux de neurones artificiels imitent le comportement des neurones du cerveau.

La première définition d'un neurone formel a été donnée par McCulloch et Pitts en 1943. Les percepts ou concepts sont physiquement représentés dans le cerveau par l'entrée en activité (simultanée) d'une assemblée de neurones (Donald Hebb, 1949) : deux neurones entrant en activité simultanément vont être associés (c'est-à-dire que leur contacts synaptiques vont être renforcés). On parle de loi de Hebb et d'associationnisme. L'hypothèse concurrente est la spécialisation de certains neurones dans des tâches cognitives complexes, par le neurone « grand-mère ». Le perceptron de Frank Rosenblatt (1958) est le premier modèle pour lequel un processus d'apprentissage a pu être défini. De cette période, date également les travaux de Widrow et Hoff. Le livre de Minski et Papert "Perceptrons" (1969) contient une étude critique très complète des perceptrons, notamment l'impossibilité de représenter la règle booléenne du OU exclusif (XOR) à l'aide d'un seul neurone formel. On lui reproche d'avoir sonné le glas des recherches sur les réseaux neuronaux dans les années 70. L'algorithme de rétro-propagation du gradient dans les réseaux multi-couches a été inventé au début des années 80 par Rumelhart et McClelland, Parker, Hinton et Le Cun. Le modèle de Hopfield (1982) utilise des réseaux totalement connectés basés sur la règle de Hebb qui ont permis de définir la notion d'attracteurs et de mémoire associative. Les poids synaptiques sont déterminés par des ensembles de données qui définissent des puits de stabilité du réseau, chaque puits correspondant à une classification. Après avoir introduit les données d'entrée à reconnaître, les données de la couche de sortie sont ré-injectées dans les neurones de la couche d'entrée jusqu'à ce que le réseau se stabilise dans un puits qui correspond à sa reconnaissance. Les cartes de Kohonen (1982) sont basées sur un algorithme non supervisé basé sur l'auto-organisation. Ce réseau permet de classer topologiquement des données d'entrée, les données ayant les propriétés les plus semblables se répartissent sur des neurones proches l'un de l'autre. La machine de Boltzman (1985) est un autre type de réseaux à attracteurs avec une dynamique de Monte-Carlo.

Le progrès de la technologie a permis le développement des ordinateurs digitaux qui envahirent notre société avec l'arrivée des microprocesseurs et des micro-ordinateurs et le développement des réseaux de neurones artificiels a débuté son réel essor dans les années 1990 quand les ordinateurs digitaux ont été suffisamment puissants pour permettre l'exécution d'algorithmes d'apprentissage. A noter que des ordinateurs neuronaux ont également été développés.

3. Classification et applications des réseaux de neurones

Un réseau de neurones formels est constitué d'un grand nombre de cellules de base interconnectées. De nombreuses variantes sont définies selon le choix de la cellule élémentaire, de l'architecture du réseau et de la dynamique du réseau. Une cellule élémentaire peut manipuler des valeurs binaires ou réelles. Les valeurs binaires sont représentées par {0,1} ou {-1,+1}. Différentes fonctions peuvent être utilisées pour le calcul de la sortie. Le calcul de la sortie peut être déterministe ou probabiliste. L'architecture du réseau peut être sans rétroaction, c'est-à-dire que la sortie d'une cellule ne peut influencer son entrée. Elle peut être avec rétroaction totale ou partielle. La dynamique du réseau peut être synchrone : toutes les cellules calculent leurs sorties respectives simultanément. La dynamique peut être asynchrone : on peut avoir une dynamique asynchrone séquentielle : les cellules calculent leurs sorties chacune à son tour en séquence ou avoir une dynamique asynchrone aléatoire. Ainsi, par exemple, le modèle du Perceptron Multi-Couches (PMC) considère des neurones déterministes à sortie réelle calculée à l'aide de la fonction sigmoïde, une architecture sans rétroaction en couches successives avec une couche d'entrées et une couche de sorties, une dynamique asynchrone séquentielle. Un autre exemple est le modèle de Hopfield qui considère des neurones à sortie stochastique {-1,+1} calculée par une fonction à seuil basée sur la fonction sigmoïde, une interconnexion complète et une dynamique synchrone, constituant une mémoire associative.

Les principales applications des réseaux de neurones sont l'optimisation et l'apprentissage. En apprentissage, les réseaux de neurones sont essentiellement utilisés pour l'apprentissage supervisé, l'apprentissage non supervisé, et l'apprentissage par renforcement.

4. La logique à seuil linéaire du neurone formel de McCulloch et Pitts

Un réseau neuronal est réalisé à partir du modèle de neurone formel de McCulloch et Pitts [1943], définit comme suit :

$$y = \Gamma(w_1.x_1 + w_2.x_2 + w_3.x_3 + ... - \theta)$$

où les x_1, x_2, x_3, … $\in\{0,1\}$, sont les valeurs d'entrée du neurone formel, y $\in\{0,1\}$, la valeur de sortie du neurone formel, les w_1, w_2, w_3, …, sont les poids synaptiques, θ est le seuil, et Γ est la fonction d'activation du neurone représentée par la fonction de Heaviside définie par

$$\Gamma(x) = 0 \text{ si } x \leq 0 \text{ et } \Gamma(x) = 1 \text{ si } x > 0.$$

Le neurone fait la somme linéaire de ses entrées pondérées et la sortie du neurone est activée et égale à la valeur, 1, si cette somme est supérieure au seuil du neurone, autrement la sortie est égale à la valeur, 0. On appelle cette logique, une logique à seuil linéaire.

5. Le Perceptron

Le perceptron est un modèle de réseau de neurones avec algorithme d'apprentissage créé par Frank Rosenblatt en 1958. Un perceptron linéaire à seuil prend en entrée n valeurs $\mathbf{x} = (x_1,..., x_n)$ et calcule une sortie y. Un perceptron est défini par la donnée de n+1 constantes :
les coefficients synaptiques $\mathbf{w} = (w_1, ..., w_n)$ et le seuil (ou le biais) θ.
La sortie y est calculée comme suit :

$y = 1$ si $\mathbf{w.x} = \Sigma_i\, w_i.x_i > \theta$

$y = 0$ si $\mathbf{w.x} = \Sigma_i\, w_i.x_i \leq \theta$

et que l'on représente par la fonction de Heaviside Γ sous le forme suivante:

$y = \Gamma(-\theta + \mathbf{w.x})$

qui vaut 1 si $\mathbf{w.x} > \theta$ et 0 si $\mathbf{w.x} \leq \theta$

Les entrées $x_1,..., x_n$ peuvent être à valeurs dans {0,1} ou réelles, les poids peuvent être entiers ou réels. Une variante très utilisée de ce modèle est de considérer une fonction de sortie prenant ses valeurs dans {-1,1} plutôt que dans {0,1}. Il existe également des modèles pour lesquels le calcul de la sortie est probabiliste. Dans la suite de cette partie sur le perceptron, nous considérerons toujours le modèle déterministe avec une sortie calculée dans {0,1}. Pour simplifier les notations, on peut remplacer le seuil par une entrée supplémentaire x_0 qui prend toujours la valeur $x_0 = -1$, associée un coefficient synaptique w_0, d'où le coefficient w_0 représente le seuil θ. On peut décomposer le calcul de la sortie y en un premier calcul de la quantité $\Sigma_i w_i x_i$ appelée potentiel post-synaptique ou l'entrée totale suivi d'une application d'une fonction d'activation sur cette entrée totale. La fonction d'activation est la fonction de Heaviside définie par :

$$y = \Gamma(\mathbf{w}.\mathbf{x}) = \Gamma(w_0x_0 + w_1x_1 + w_2x_2) = \Gamma(-\theta + w_1x_1 + w_2x_2)$$

6. PERCEPTRON : apprentissage par correction d'erreur

L'algorithme d'apprentissage peut être décrit succinctement de la manière suivante. On initialise les poids du perceptron à des valeurs quelconques. A chaque fois que l'on présente un nouvel exemple, on ajuste les poids selon que le perceptron l'a correctement classé ou non. L'algorithme s'arrête lorsque tous les exemples ont été présentés sans modification d'aucun poids. L'algorithme d'apprentissage par correction d'erreur du perceptron linéaire à seuil est :

Algorithme par correction d'erreur:
Entrée : un échantillon S de $R^p \times \{0,1\}$ ou $\{0,1\}^p \times \{0,1\}$
Initialisation aléatoire des poids w_i pour i entre 0 et p
Répéter
Prendre un exemple $(\boldsymbol{x},d)$ dans S
Calculer la sortie y du perceptron pour l'entrée $\boldsymbol{x}$
- - Mise à jour des poids - -
Pour i de 0 à p
$w_i \leftarrow w_i + (d-y)x_i$
finpour
finRépéter
Sortie : Un perceptron P défini par $(w_0, w_1, ..., w_p)$

7. Apprentissage par descente de gradient

Plutôt que d'obtenir un perceptron qui classifie correctement tous les exemples, il s'agira maintenant de calculer une erreur et d'essayer de minimiser cette erreur. Pour introduire cette notion d'erreur, on utilise des poids réels et on élimine la notion de seuil (ou d'entrée supplémentaire), ce qui signifie que la sortie sera égale au potentiel post-synaptique et sera donc réelle. Soit le couple
$(\underline{x}(n), \underline{d}(n)), \quad n \in S$
désignant la nième donnée d'entraînement du réseau (dans un ensemble S) où
$\mathbf{x}(n) = (x_1(n), ..., x_p(n))$ sont les p entrées
et
$\mathbf{d}(n) = (d_1(n), ..., d_q(n))$ sont les q sorties
L'algorithme consiste à mesurer l'erreur entre les sorties désirées $\mathbf{d}(n)$ et les sorties observées $\mathbf{y}(n)$:

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{y}(n)$$

Soit $E(\mathbf{w})$ la somme des erreurs quadratiques observées sur le neurone de sortie:

$$E(w) = (1/2) \Sigma_{n\in S} \mathbf{e}^2(n)$$

La sortie y(n) du neurone est défini par

$$\mathbf{y}(n) = \mathbf{w}(n).\mathbf{x}$$

Pour corriger l'erreur, il s'agit de modifier le poids $w_i(n)$ dans le sens opposé du gradient
$\partial E(\mathbf{w})/\partial w_i(n)$
de l'erreur

$$\partial E(\mathbf{w})/\partial w_i = - \Sigma_{n\in S} e(n)\, x_i$$

$$\Delta w_i = - \eta \partial E(\mathbf{w})/\partial w_i = \eta \Sigma_{n\in S} e(n)\, x_i$$

avec $0 \leq \eta \leq 1$ représentant le taux d'apprentissage ou gain de l'algorithme.
L'algorithme d'apprentissage par descente de gradient du perceptron linéaire peut être défini :

Algorithme par descente de gradient :
Entrée : un échantillon S de $R^p \times \{0,1\}$; η
Initialisation aléatoire des poids w_i pour i entre 1 et p
Répéter
Pour tout i $\Delta w_i \leftarrow 0$ **finPour**
Pour tout exemple ($\boldsymbol{x}(n)$,$\boldsymbol{d}(n)$) de S
calculer la sortie $\boldsymbol{y}(n)$
Pour tout i $\Delta w_i \leftarrow \Delta w_i + \eta \Sigma_{n \in S} (d - y) x_i$ **finPour**
finPour
Pour tout i $w_i \leftarrow w_i + \Delta w_i$ **finPour**
finRépéter
Sortie : Un perceptron P défini par (w_1, ..., w_p)

8. Algorithme d'apprentissage de Widrow-Hoff

La méthode des moindres carrés, bien connues des statisticiens a été appliquée aux réseaux neuronaux par Bernard Widrow et Marcian Hoff (1960). Cet algorithme est une variante très utilisée de l'algorithme précédent. Au lieu de calculer les variations des poids en sommant sur tous les exemples de S, l'idée est de modifier les poids à chaque présentation d'exemple. La règle de modification des poids donnée devient :

$$\Delta w_i = \eta \, e(n) \, x_i$$

Cette règle est appelée règle delta, ou règle ADALINE, qui est un acronyme pour ADAptive LINear Element (ou ADAptive LInear NEuron).

L'algorithme s'écrit alors :

Algorithme de Widrow-Hoff :
Entrée : un échantillon S de $R^p \times \{0,1\}$; η
Initialisation aléatoire des poids w_i pour i entre 1 et p
Répéter
Prendre un exemple ($\boldsymbol{x}$,d) dans S
Calculer la sortie y du perceptron pour l'entrée $\boldsymbol{x}$
- - Mise à jour des poids - -
Pour i de 1 à p
$w_i \leftarrow w_i + \eta \, (d-y)x_i$
finpour
finRépéter
Sortie : Un perceptron P défini par (w_0, w_1, ..., w_p)

9. La logique à seuil non-linéaire des machines neuronales de Dubois et Resconi

Un des problèmes qui avait été soulevé pour le neurone formel était qu'un seul neurone ne pouvait pas réaliser certaines tables booléennes, comme le célèbre exemple de la table booléenne du OU exclusif, donnée à la Table 1 : y = 0, si $x_1 = x_2$, autrement y = 1, où $x_1, x_2 \in \{0, 1\}$ et $y \in \{0, 1\}$

Table 1 : OU exclusif

x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

En 1993, j'ai résolu ce problème en proposant, avec Germano Resconi, le nouveau paradigme de logique à seuil non-linéaire pour les machines neuronales [Dubois et Resconi, 1993].

Le neurone formel non-linéaire pour le OU exclusif est donné par

$$y = \Gamma(1.x_1 + 1.x_2 - 2.x_1.x_2) = 1.x_1 + 1.x_2 - 2.x_1.x_2$$

qui est une fonction fixe de Heaviside (la fonction est égale à son argument). Remarquons que l'on peut facilement convertir un réseau de neurones non-linéaires en un réseau de neurones linéaires. Toutes les tables booléennes, quel que soit le nombre d'entrée et de sortie, peuvent être réalisées par un réseau de neurones formels linéaires ou non-linéaires.

Considérons maintenant l'algorithme d'apprentissage du OU exclusif (XOR), à l'aide d'un seul neurone, représenté de la manière suivante : $y = -\theta + w_1x_1 + w_2x_2 + w_{12}x_1x_2$

En introduisant le neurone $x_0 = -1$ et le poids $w_0 = \theta$, on obtient :

$y = w_0x_0 + w_1x_1 + w_2x_2 + w_{12}x_1x_2$

L'algorithme de Widrow-Hoff, adapté au cas non-linéaire, par Daniel Dubois [2005]:

Algorithme non-linéaire:

Entrée : un échantillon S de $R^p \times \{0,1\}$; η

Initialisation aléatoire des poids w_i pour i entre 1 et p

Répéter

Prendre un exemple $(\boldsymbol{x},d)$ dans S

Calculer la sortie y du perceptron pour l'entrée $\boldsymbol{x}$

- - Mise à jour des poids - -

Pour i de 1 à p

$w_i \leftarrow w_i + \eta\ (d-y)x_i$

Pour j de i+1 à p

$w_{ij} \leftarrow w_{ij} + \eta\ (d-y)x_i\ x_j$

finpour

finpour

finRépéter

Sortie : Un perceptron P défini par $(w_0, w_1, ..., w_p)$

Remarque : pour des neurones à 3 entrées, on a

$y = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_{12}x_1x_2 + w_{13}x_1x_3 + w_{23}x_2x_3 + w_{123}x_1x_2x_3$

et ainsi de suite pour les neurones à n entrées.

10. Simulation des algorithmes d'apprentissage

Une simulation des algorithmes des types d'apprentissage décrits, écrits en javascript, par Daniel Dubois (2005), sont donnés ci-après, aux Figures 1 à 3.

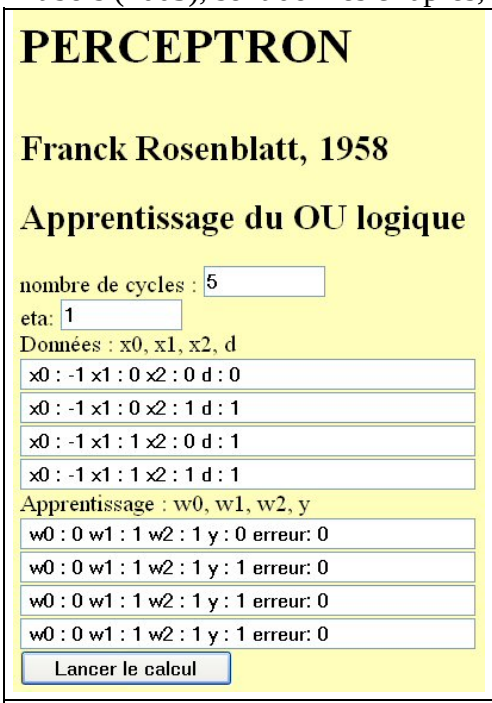

Figure 1 : Apprentissage du OU avec le Perceptron.

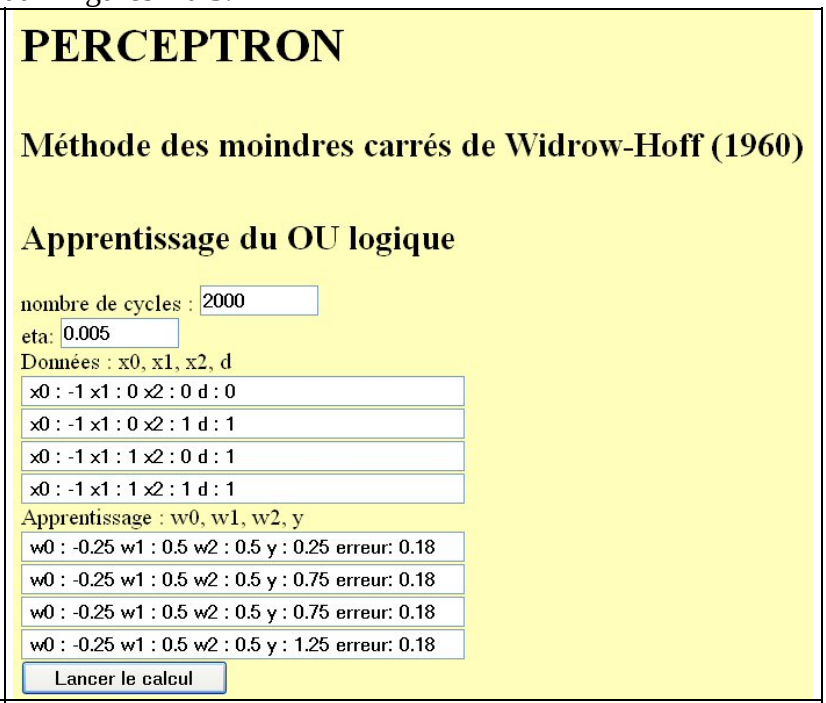

Figure 2 : Apprentissage du OU avec le Perceptron de Widrow-Hoff.

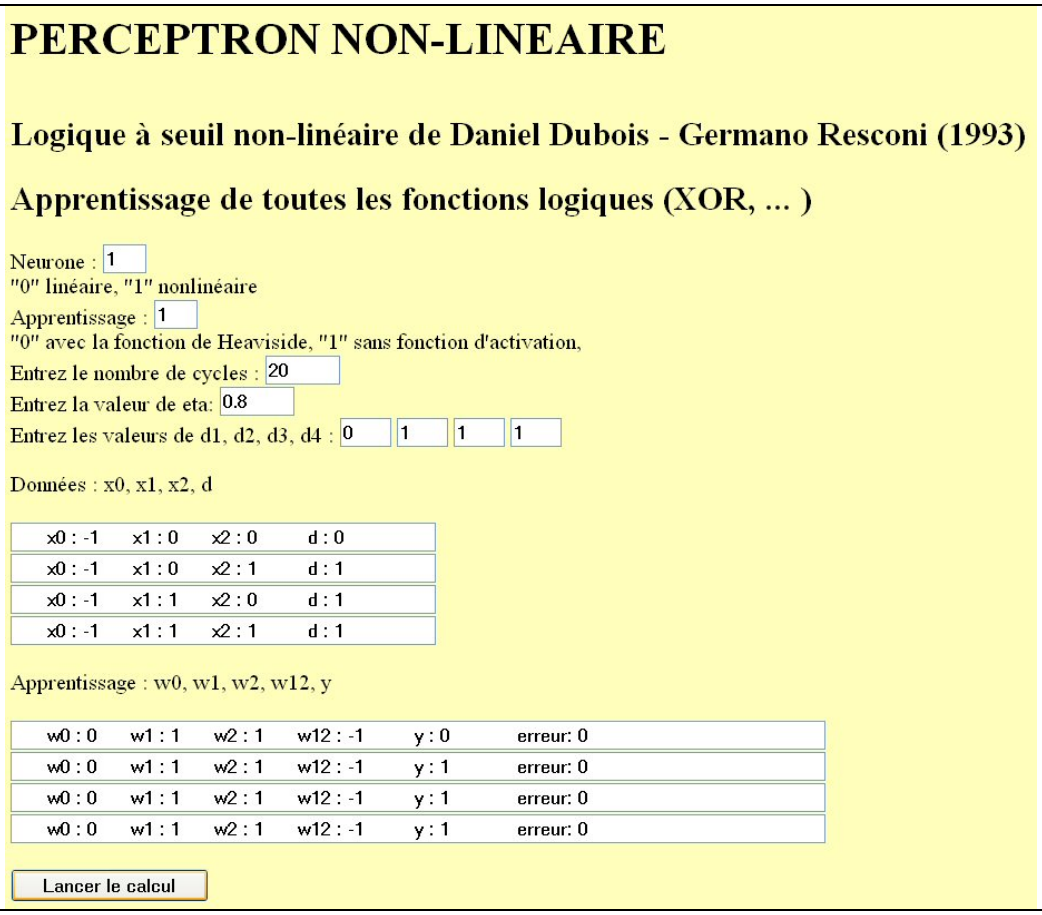

Figure 3a : Apprentissage des fonctions logiques avec le Perceptron non-linéaire : OU.

11. Systèmes neuronaux de mémoires dynamiques

Comme le cerveau, une des composantes essentielles des ordinateurs est la mémoire.

Les ordinateurs sont capables d'émuler des réseaux de neurones artificiels par la programmation. A l'inverse, peut-on émuler un ordinateur sur un réseau de neurones artificiels par l'apprentissage ? La réponse est oui, car, comme nous l'avons montré dans la section précédente, nous pouvons réaliser toutes les tables booléennes par un réseau de neurones.

Nous présentons dans cette section des systèmes neuronaux de mémoires dynamiques.

Voici une mémoire élémentaire, de 1 bit, sous forme d'une table logique, Table 2.

Table 2 : Mémoire à bascule

set	**reset**	**m**	
x_1	**x_2**	**y_1**	**y_2**
0	0	**1**	**1**
0	1	**1**	**0**
1	0	**0**	**1**
1	1	**0**	**1**
1	1	**1**	**0**

Dans cette Table 2, la première entrée, x_1 permet d'initialiser (set) la mémoire, et la seconde entrée, x_2, permet de réinitialiser (reset) la mémoire à la valeur, 0 ou 1. La première sortie, y_1, stocke la valeur à mémoriser, m = 0 ou m = 1, la seconde sortie, y_2, étant le seconde sortie.

Les opérations successives suivantes expliquent le mode de fonctionnement de cette mémoire :

Si on met $x_1 = 0$ et on réinitialise $x_2 = 1$, on a mis la mémoire m = 1, ensuite on met $x_1 = x_2 = 1$, et on obtient la même valeur de sortie m = 1.

Si on met $x_1 = 1$ et on réinitialise $x_2 = 0$, on a mis la mémoire m = 0, ensuite on met $x_1 = x_2 = 1$, et on obtient la même valeur de sortie m = 0.

NB : la première ligne de cette table n'est pas utilisée dans ce type de mémoire.

Les neurones formels de cette mémoire à bascule sont donnés par [Dubois, 1998] :

$$y_1 = \Gamma\ (-1.x_1 - 1.y_2 + 2\ ),\ y_2 = \Gamma\ (-1.x_2 - 1.y_1 + 2\ )$$

avec les poids $w_1 = w_2 = -1$ et le seuil $\theta = -1$. Le schéma du système neuronal de cette mémoire à bascule [Dubois, 1999], montre la nécessité de deux neurones avec une rétroaction de chacun des neurones sur l'autre, pour mémoriser un seul bit. Cette mémoire neuronale a également été proposée avec un seul neurone [Dubois, 1999] :

Table 3. Mémoire hyperincursive sans seuil

x_1	x_2	y
0	0	y
0	1	1
1	0	0
1	1	y

Le neurone formel est [Dubois, 1998] :

$$y = \Gamma\ (\ x_2 - x_1 + y\ )$$

où $y = y_1$, et le cas $x_1 = x_2 = 0$ est identique au cas $x_1 = x_2 = 1$.

Ce type de mémoire neuronale est une mémoire dynamique, puisque son contenu à mémoriser peut changer aussi souvent que possible, sans que les poids synaptiques et le seuil ne soient modifier.

L'apprentissage de mémoires neuronales de ce type ne doit se faire qu'une fois, puisque l'apprentissage consiste à fixer les poids synaptiques et le seuil. La plasticité synaptique n'est utile que pour cette seule phase d'apprentissage.

Des mémoires neuronales élémentaires de types Données / Horloge, Mémoire / Rappel et Mémoire / Reconnaissance [Dubois, 1999] sont présentées ci-après.

Mémoire de type Données / Horloge

Au lieu d'utiliser un ensemble de réinitialisation à bascule pour créer une mémoire d'un bit, la bascule de type-D est une mémoire d'un bit pour lequel les entrées sont une entrée de données d et une entrée d'horloge c. Lorsque l'entrée d'horloge est c = 1, la sortie est m = d, la sortie est égale à la donnée entrée. Lorsque l'entrée d'horloge est c = 0, la sortie est la dernière donnée entrée lorsque c = 1. Une table booléenne de la mémoire d'un bit du type données / horloge a été construite comme suit

Table 4. Mémoire hyperincursive d'un bit de type données / horloge

Donnée d	Horloge c	Set s	Reset r	Mémoire m
0	0	1	1	m
0	1	1	0	0
1	0	1	1	m
1	1	0	1	1

Les fonctions de Heaviside des neurones formels sont les suivantes :

$$s = \Gamma\ [\ -c - d + 2\ ],\ r = \Gamma\ [\ d - c + 1\ ],\ m = \Gamma\ [\ r - s + m\ ]$$

où l'équation de la mémoire m est identique à la condition pour laquelle $x_1 = s$, $x_2 = r$ et $y = m$.

Mémoire / Rappel

Tables 5. Table booléenne de la mémoire neuronale de type Mémoire/Rappel

Mémoire m	Récupération re	Rappel y
0	0	0
0	1	0
1	0	0
1	1	1

Quand la récupération est re = 0, le rappel est toujours y = 0, tandis que quand la récupération est re = 1, on obtient le rappel de la donnée mémorisée, y = m, avec la fonction de Heaviside du neurone formel, y :

$$y = \Gamma [m + re - 1]$$

Mémoire / Reconnaissance

Tables 6. Table booléenne de la mémoire neuronale de type Mémoire/ Reconnaissance

Mémoire m	Donnée à reconnaître drec	Reconnaissance y
0	0	0
0	1	1
1	0	1
1	1	0

La seconde table est un OU exclusif, c'est-à-dire que quand la donnée à reconnaître drec est identique à la donnée mémorisée m, la reconnaissance est y = 0.

Le réseau neuronal du OU exclusif est le suivant :

$$y_2 = \Gamma(- m + drec),\ y_3 = \Gamma(m - drec),\ y = \Gamma(y_2 + y_3)$$

Références

1. W.S. McCulloch et W. Pitts, 1943. A logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of mathematical Biophysics, vol. 5 (1943) 115-133.
2. D.O. Hebb, 1949. The Organization of Behaviour, Wiley, New York 1949.
3. R. Rosenblatt, 1958. Principles of Neurodynamics, Spartan Books, New York 1962.
4. B. Widrow and M. E. Hoff, 1960. Adaptive switching circuits. In 1960 IRE WESCON Convention Record, pages 96-104, New York, 1960. IRE.
5. M. Minsky et S. Papert, 1969. Perceptrons , the MIT Press, Cambridge 1969.
6. J.J. Hopfield, 1982. Neural Networks and Physical Systems with Emergent Collective Computational Abilities , Proceedings of the National Academy of Sciences, USA, 1982, pp. 2554-2558.
7. T. Kohonen, 1982. Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59-69, 1982.
8. David H. Ackley, Geoffrey E. Hinton, Terrence J. Sejnowski, 1985. A Learning Algorithm for Boltzmann Machines. Cognitive Science 9(1): 147-169 (1985)
9. D.E. Rumelhart et al, 1986. Learning representations by back-propagating errors , Nature, vol. 323 (1986) 323. Voir aussi Parallel Distributed Processing, the MIT Press, vol. 1, Cambridge 1986.
10. Daniel M. Dubois et Germano Resconi, 1993. Mathematical Foundation of a Non-linear Threshold Logic: a new Paradigm for the Technology of Neural Machines, ACADEMIE ROYALE DE BELGIQUE, Bulletin de la Classe des Sciences, 6ème série, Tome IV, 1-6, 23 pages, 1993.
11. D. M. Dubois (1998), « Boolean Soft Computing by Non-linear Neural Networks with Hyperincursive Stack Memory ». In Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, Edited by O. Kaynak, L. A. Zadeh, B. Türksen, I. J. Rudas, NATO ASI Series, Series F: Computer and Systems Sciences, volume. 162, Springer-Verlag.
12. Daniel M. Dubois (1999), « Hyperincursive McCulloch and Pitts Neurons for Designing a Computing Flip-Flop Memory ». Computing Anticipatory Systems: CASYS'98 - Second International Conference. Edited by Daniel M. Dubois, Published by The American Institute of Physics, AIP Conference Proceedings 465, pp. 3-21.
13. D. M. Dubois (2008). New Trends in Computing Anticipatory Systems :Emergence of Artificial Conscious Intelligence with Machine Learning Natural Language. COMPUTING ANTICIPATORY SYSTEMS: CASYS'07-Eighth International Conference. AIP Conference Proceedings, vol. 1051, pp. 25-32.
14. Daniel M. Dubois (2005, 19 mars), Introduction aux Réseaux de Neurones Artificiels en Intelligence Artificielle, Séminaire METI, HEC, © Daniel M. Dubois, 2005
15. D. M. Dubois (2010). Breakthrough in the Human Decision Making Based on an Unconscious Origin of Free Will. ACTA SYSTEMICA, Vol. X, pp. 13-18.
16. D. M. Dubois (2010). Natural and Artificial Intelligence, Language, Consciousness, Emotion, and Anticipation. COMPUTING ANTICIPATORY SYSTEMS: CASYS'09, Ninth International Conference. AIP Conference Proceedings, vol. 1303, pp. 236-245.
17. Daniel Dubois (1990). LE LABYRINTHE DE L'INTELLIGENCE – De l'intelligence naturelle à l'intelligence fractale. Paris: InterEditions.