Aprentissage des neurones flous

Jelena Godjevac
LAMI IN F EPFL, Ecublens, 1015 Lausanne, Suisse
godjevac@di.epfl.ch

Abstract

Le but du travail présenté dans cet article est de proposer un modèle de neurone flou et de tester les méthodes d'apprentissage possibles. La première partie est consacrée au *neurone artificiel formel* proposé par Mc Cullogh et Pitts, bien étudié dans la littérature sur les réseaux de neurones. Une expérience simple de séparation de deux classes d'objets est présentée. Dans la deuxième partie, on propose une architecture pour un *neurone flou* ainsi que la méthode d'apprentissage basée sur l'algorithme de descente du gradient. La même application est présentée. On montre que les performances de l'apprentissage du neurone flou sont satisfaisants. Le neurone classique est plus efficace parce que ce problème de séparation des classes est linéaire.

1. Neurone artificiel classique

1. 1 Le modèle du neurone

Il y a autant de modèles proposés pour un neurone artificiel que d'architectures pour un réseau de neurones. Le modèle le plus simple est celui de McCulloch et Pitts, présenté sur la Figure 1. Dans ce neurone, $x_1,...,x_n$ sont les *stimulis d'entrée*, $w_1,...,w_n$ sont les *poids synaptiques* et θ est la *valeur du seuil.*

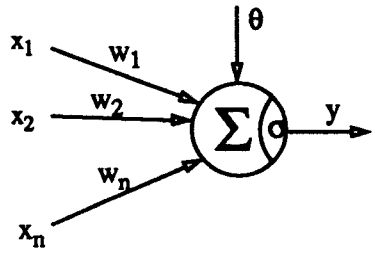


Figure 1. Modèle d'un neurone artificiel

Le *potentiel* de ce neurone est:

$$p = \sum_{i=1}^{n} w_i x_i \qquad 1.1.1$$

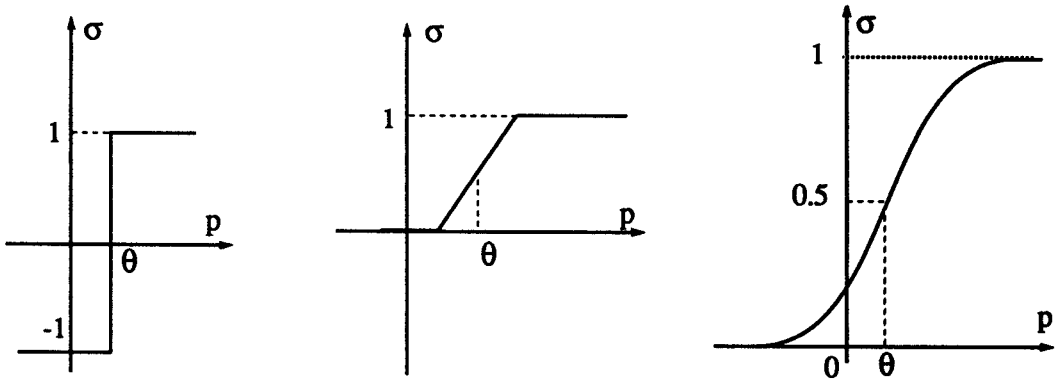
On peut remarquer que le neurone effectue une somme pondérée des entrées et fait passer le résultat par une nonlinearité (une *fonction d'activation*) $\sigma(p - \theta)$. Les caractéristiques de ce neurone sont le seuil et le type de nonlinearité. Les fonctions d'activation les plus utilisées sont montrées à la Figure 2. Dans la plupart des applications, on prend la fonction sigmoïde, parce qu'elle est considérée comme la meilleure approximation du neurone biologique [Bla92]. Elle est présentée à la Figure 3 est sa forme analytique est:

$$\sigma(x) = \frac{1}{1 + e^{\frac{-(x-\theta)}{T}}} \qquad 1.1.2$$

où θ est le seuil et 1/T est la pente à l'origine de la courbe. Si T→ 0, la fonction sigmoïde devient la fonction hard limiter présentée sur la Figure 2. Alors, la sortie du neurone est:

$$y = \sigma(p-\theta) = \sigma\left(\sum_{i=1}^{n} w_i x_i - \theta\right) \qquad 1.1.3$$

Les réseaux de neurones sont caractérisés par leur topologie (la matrice de connections), par les caractéristiques des neurones et par leur règle d'apprentissage. Ces règles spécifient un ensemble initial des poids et indiquent comment ces poids doivent être adaptés pour améliorer la performance du réseau.



Hard limiter | Fonction treshold | Fonction sigmoïde

Figure 2. Nonlinearités dans le modèle de neurone artificiel

1. 2 Méthode d'apprentissage

Le but de l'expérience qui suit est de séparer deux classes d'objets dans un plan (l'espace des stimulis) en deux régions par une droite d'équation:

$$w_1 x_1 + w_2 x_2 - \theta = 0 \qquad 1.2.1$$

On peut le faire avec un neurone qui a deux entrées x_1 et x_2, deux poids synaptiques w_1 et w_2 et le seuil θ. La sortie du neurone est dénotée par y.

Supposons qu'il faut séparer les données montrées sur la Figure 4. cela veut dire que le neurone doit apprendre à approximer la fonction nonlinéaire presentée à la Figure 3.

L'algorithme d'apprentissage est une sorte d'apprentissage supervisé [Tou92]. Cela veut dire qu'il existe une base d'apprentissage avec les entrées et les sorties désirées du neurone (ou du réseau). Le superviseur présente au neurone chaque vecteur d'entrée et si la sortie désirée y_d est différente de la sortie du neurone, on modifie les poids. C'est un algorithme itératif qu'on peut diviser en 5 étapes:

1. Initialisations des poids w_1 et w_2 et du seuil θ à des valeurs aléatoires
2. Présentation d'une entré de la base d'apprentissage: x_1, x_2 et de la sortie désirée y_d
3. Calcul de la sortie y du neurone (équation 1.1.3)
4. Si la sortie du neurone est différente de la sortie désirée, pour cette entré, modification des poids synaptiques d'après la relation (μ est une constante positive entre 0 et 1):

$$w_i(t+1) = w_i(t) + \mu x_i (y - y_d) \qquad 1.2.2$$

5. Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement, retour a l'étape 2.

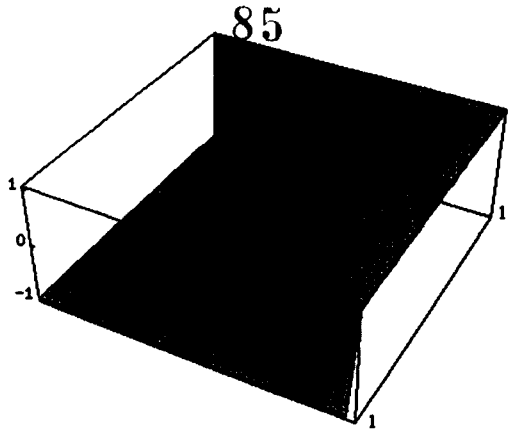


Figure 3. Modèle de la fonction nonlinéaire que le neurone doit apprendre

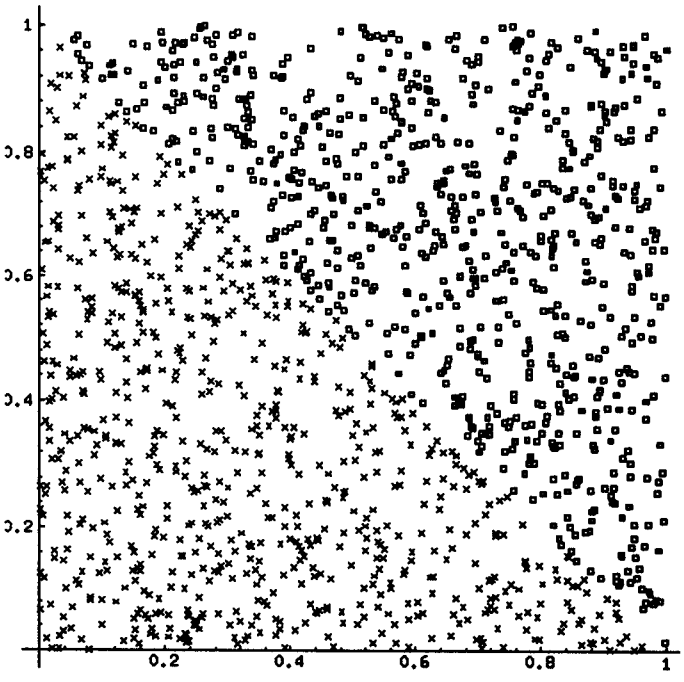


Figure 4. Deux classes separables dans le plan (x_1,x_2)

Pour appliquer l'équation 1.2.2 sur la valeur de seuil, il est nécessaire de présenter la droite 1.2.1 comme:

$$w_1x_1 + w_2x_2 + w_3x_3 = 0 \qquad 1.2.3$$

La valeur de seuil est:

$$\theta = -w_3x_3 = (-1)\,w_3 \qquad 1.2.4$$

L'entrée x_3 a une valeur fixe -1 et l'adaptation de poid w_3 est égale à l'adaptation de la valeur de seuil. On applique l'algorithme présenté ci-dessus pour les trois valeurs de poid de la même manière. Les entrées x_1 et x_2 prennent les valeurs de la base d'apprentissage, mais l'entrée x_3 a la valeur fixe -1 pendant l'apprentissage.

1. 3 Résultats de l'expérience

Dans cette expérience, la fonction d'activation utilisée est le hard limiter de la Figure 2b. Le coefficient μ est fixé à 0.02.

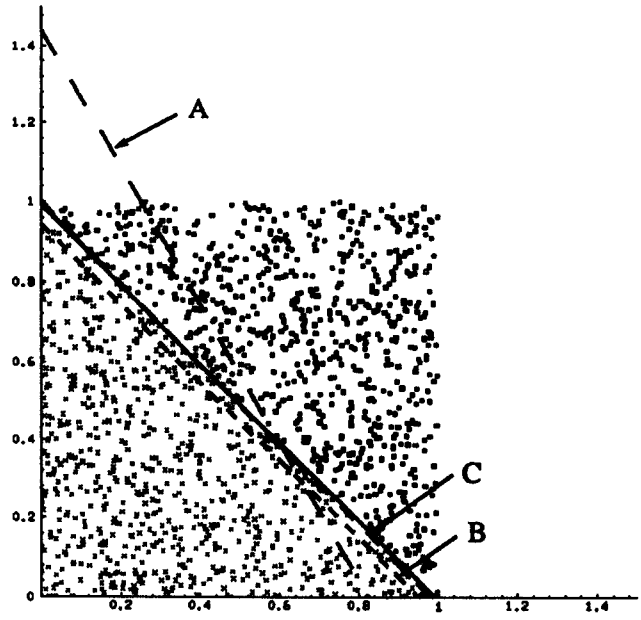


Figure 5. Résultat d'apprentissage après A. 100, B. 500, C. 1000 itérations

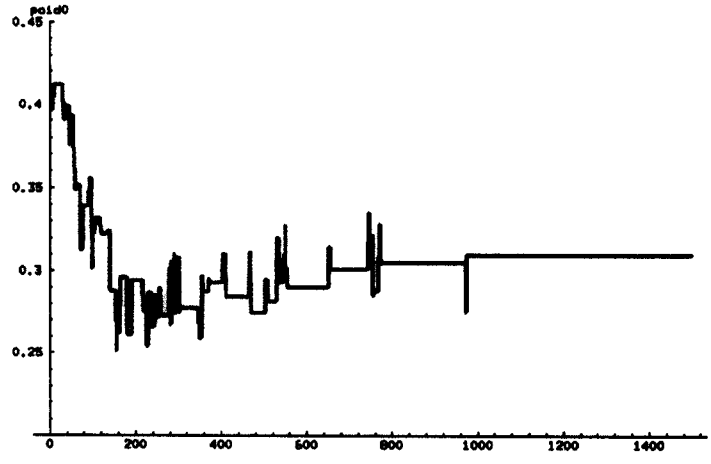


Figure 6. Changement du poid w_1 pendant l'apprentissage

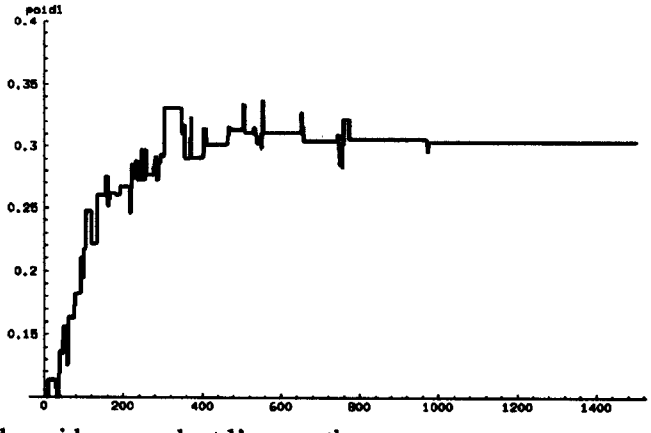


Figure 7. Changement du poid w_2 pendant l'apprentissage

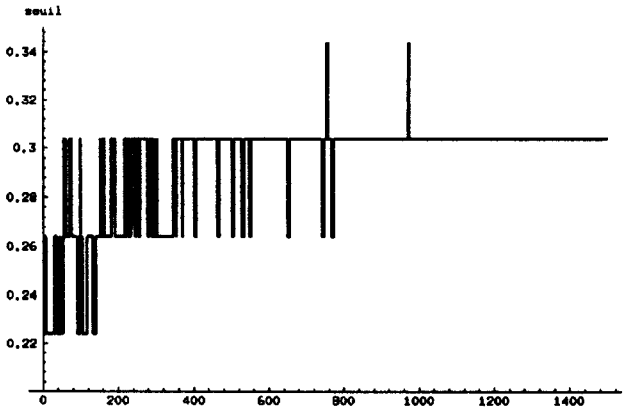


Figure 8. Evolution de la valeur de seuil pendant l'apprentissage

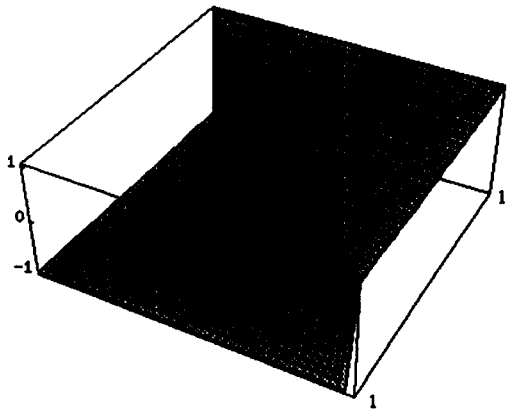


Figure 9. Fonction nonlinéaire apprise par le neurone classique

Le résultat de l'apprentissage après 100, 500 et 1000 itérations est montré à la Figure 5. On peut voir que les classes sont complètement séparées et qu'après 1000 exemples de la base d'apprentissage, l'adaptation des poids est terminée.Sur les Figure 6 et Figure 7, on peut voir l'évolution des valeurs des poids et sur la Figure 8, l'évolution de la valeur du seuil pendant l'apprentissage. Le résultat est aussi satisfaisant si la fonction d'activation est la sigmoïde (Figure 3), mais on ne va pas montrer ici le résultat. Si on représente en 3D la fonction que le neurone a appris (ou approximé), on a la fonction montrée sur la Figure 9.

2. Neurones flous

Gupta et al. [Gup91] proposent trois modèles de neurones flous, ainsi que les idées principales sur les mécanismes d'apprentissage et d'adaptation, mais sans donner de détails d'implantation. Ils affirment que les neurones flous sont un outil très puisant pour la modélisation des phénomènes associés avec la réflexion humaine et la perception.

2. 1 Structure de base d'un neurone flou

La structure générale d'un neurone flou est très similaire au modèle du neurone à la Figure 1. On l'a présenté à la Figure 10. Les entrées sont les sous ensembles flous x_1,..., x_n. Ces sous ensembles peuvent avoir les valeurs linguistiques comme par exemple: chaud, large, haut etc. La première opération effectuée est la pondération, mais elle n'est pas effectuée comme dans le cas d'un neurone classique.

La seconde est l'opération d'agrégation qui est en général, la fonction d'implication floue et dans la plupart de cas, on prend une T-norme ou une T-conorme.

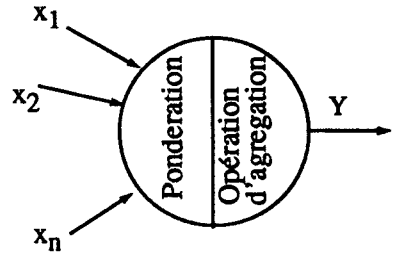


Figure 10. Le modèle général d'un neurone flou

2. 2 Modèle du neurone flou

Il y a plusieurs modèles de neurones flous décrits dans la littérature [Gup91]. Seul un de ces modèles sera considéré par la suite. Ce modèle est une extension du neurone classique. Comme montré à la Figure 11, il a *n* entrées floues ou non-floues et les opérations de pondération sont remplacées par des fonctions d'appartenance. Cela veut dire qu'à chaque entrée on attribue une valeur d'appartenance à un sous-ensemble flou.

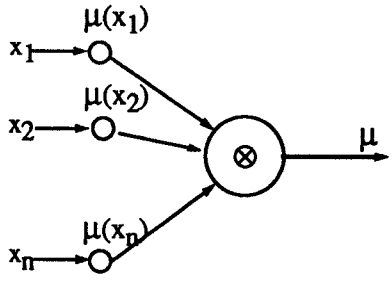


Figure 11. Modèle du neurone flou

Les opérations de pondération sont dites synaptiques. Les résultats sont les valeurs d'appartenance. Ces valeurs sont transformées par des opérations floues, appelées somatiques. La sortie du neurone (qui est considérée comme le "niveau de confiance" ou le "niveau de vérité") est une valeur précise dans un intervalle [0,1].

L'opération floue appliquée dans ce neurone peut être une des opérations comme min, max ou une des T-normes ou T-conormes [Gup91]. Si on désigne cette opération comme ⊗, on peut représenter mathematiquement le neurone par la relation suivante:

$$\mu(x_1,x_2,...,x_n) = \mu_1(x_1) \otimes \mu_2(x_2) \otimes ... \otimes \mu_n(x_n)$$

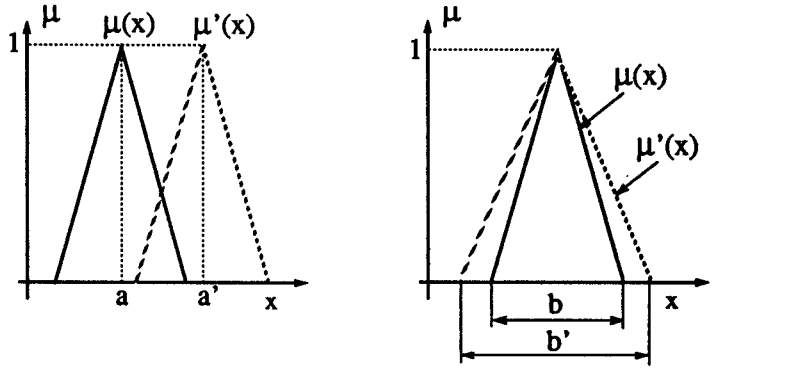
ou $x_1,...,\ x_n$ sont des entrées dans le neurone, $\mu_1,...,\ \mu_n$ sont des fonctions d'appartenance et μ est la sortie du neurone.

2. 3 Méthodes d'apprentissage

Le but de la conception d'un réseau neuro-flou est d'effectuer certaines tâches. Les mécanismes d'adaptation proposés ici sont synaptiques et somatiques. Ils sont basés sur l'apprentissage supervisé dans les réseaux non-flous.

L'adaptation synaptique consiste à la pondération des entrées du neurone. Dans le cas d'un neurone classique, les entrées sont multipliées avec les poids synaptiques et l'apprentissage consiste aux changements des poids synaptiques. Dans le cas d'un neurone flou, cette opération est plus complexe. Si

les fonctions d'appartenance ou les entrées sont triangulaires les modifications proposées sont montrées à la Figure 12. Les lignes pointillées représentent les entrées floues avant la modification.



a - centre avant l'apprentissage
a'- centre après l'apprentissage
b - largeur avant l'apprentissage
b'- largeur après l'apprentissage

Figure 12. Changement des fonctions d'appartenance pendant l'apprentissage

Les adaptations somatiques sont les modifications de la structure de neurone flou. Elle peuvent être:

- changements de la base des règles
- changements des fonctions d'appartenance
- changements des opérations d'agrégation

3. Expérience d'apprentissage

Le but de cette expérience est de concevoir et de tester un algorithme pour un neurone flou de la même manière comme pour un neurone classique. Elle était inspirée par l'article de M. Gupta et al. [Gup91] qui propose les différentes architectures des neurones flous. Aucune méthode d'apprentissage n'étant proposée, nous allons en développer une dans la suite de cet article.

3. 1 Modèle du neurone simulé

Les premiers essaies étaient de répéter l'expérience du paragraphe 1. 3, mais avec le neurone flou. La fonction qu'il faut apprendre est montrée à la Figure 3.

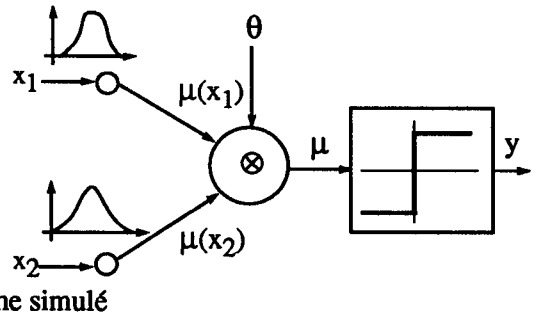


Figure 13. Modèle du neurone simulé

Le neurone simulé (Figure 13) a deux entrées et une sortie. L'opération de pondération est la fuzzification. Cela veut dire qu'il faut d'abord définir la fonction d'appartenance de chaque entrée. Mais comment quantifier l'espace des entrées? Dans un système flou, il est nécessaire de définir plusieurs fonctions d'appartenance pour chacune des entrées. Au lieu de compliquer le modèle du neurone élémentaire en définissant plusieurs fonctions d'appartenance, on utilisera seulement une et le pas suivant sera de ressembler plusieurs neurones élémentaires dans un réseau.

3. 2 L'apprentissage d'un neurone flou

Les fonctions d'appartenance du neurone simulé, sont définies comme à la Figure 14.

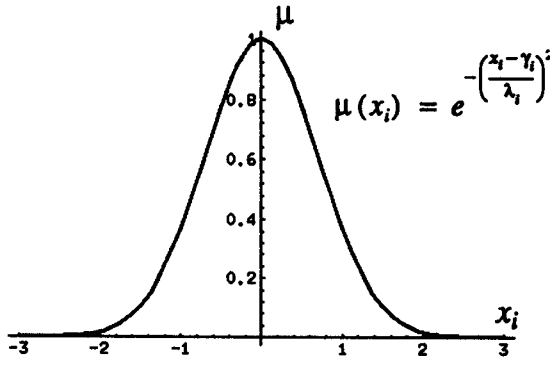


Figure 14. Fonction d'appartenance en forme de courbe en cloche, $\gamma_i = 0$ et $\lambda_i = 1$

L'opération ⊗ dans le neurone est l'adition. La sortie du neurone est:

$$y = \text{sgn}[\mu(x_1) + \mu(x_2) - \theta] = \sigma[\sum \mu(x_i) - \theta] \qquad 3.2.1$$

L'erreur entre la sortie désirée y_d est la sortie du neurone y qu'on veut minimiser est donnée par:

$$E(t) = \frac{1}{2}(y(t) - y_d)^2 \qquad 3.2.2$$

L'adaptation des paramètres est basée sur l'apprentissage supervisé et sur la méthode de descente de gradient. Cela veut dire qu'il faut trouver les dérivés de E par tous ses paramètres (équations 3.2.4, 3.2.5 et 3.2.6). Les changements des paramètres a_i correspondent aux modifications présentées sur la Figure 12a et les changements des b_i aux modifications présentées sur la Figure 12b. La règle générale pour le changement des paramètres est donne par:

$$z(t+1) = z(t) + K_z \frac{\partial E(t)}{\partial z} \qquad 3.2.3$$

On peut remarquer que l'opération ⊗ n'est pas une des opérations "classique" de la logique floue comme min ou max. La raison d'éviter l'utilisation de ces opérateurs est qu'ils ne sont pas derivable. Les dérivés partielles de l'erreur sont:

$$\frac{\partial E}{\partial \gamma_i}(t) = (y(t) - y_d)\, e^{-\left(\frac{x_i(t)-\gamma_i(t)}{\lambda_i(t)}\right)^2} \frac{2}{\lambda_i^2(t)}(x_i(t) - \gamma_i(t)) \qquad 3.2.4$$

$$\frac{\partial E}{\partial \lambda_i}(t) = (y(t) - y_d)\, e^{-\left(\frac{x_i(t)-\gamma_i(t)}{\lambda_i(t)}\right)^2} \frac{2}{\lambda_i^3(t)}(x_i(t) - \gamma_i(t))^2 \qquad 3.2.5$$

$$\frac{\partial E}{\partial \theta}(t) = -(y(t) - y_d) \qquad 3.2.6$$

La modification des paramètres est donnée par:

$$\gamma_i(t+1) = \gamma_i(t) + K_\gamma (y(t) - y_d)\, e^{-\left(\frac{x_i(t)-\gamma_i(t)}{\lambda_i(t)}\right)^2} \frac{2}{\lambda_i^2(t)}(x_i(t) - \gamma_i(t)) \qquad 3.2.7$$

$$\lambda_i(t+1) = \lambda_i(t) + K_\lambda(y(t) - y_d)\, e^{-\left(\frac{x_i(t)-\gamma_i(t)}{\lambda_i(t)}\right)^2} \frac{2}{\lambda_i^3(t)} (x_i(t) - \gamma_i(t))^2 \qquad 3.2.8$$

$$\theta_i(t+1) = \theta_i(t) - K_\theta(y(t) - y_d) \qquad 3.2.9$$

L'algorithme d'adaptation est le même comme celui pour un neurone classique. La seule différence est que la modification de paramètres se fait séparément pour chaque sorte de paramètres.

3. 3 Résultats de l'expérience

La fonction que le neurone doit apprendre est montrée sur la Figure 3. Cette méthode d'apprentissage marche parfaitement bien. Le neurone a appris la fonction présentée après 7000 itérations.

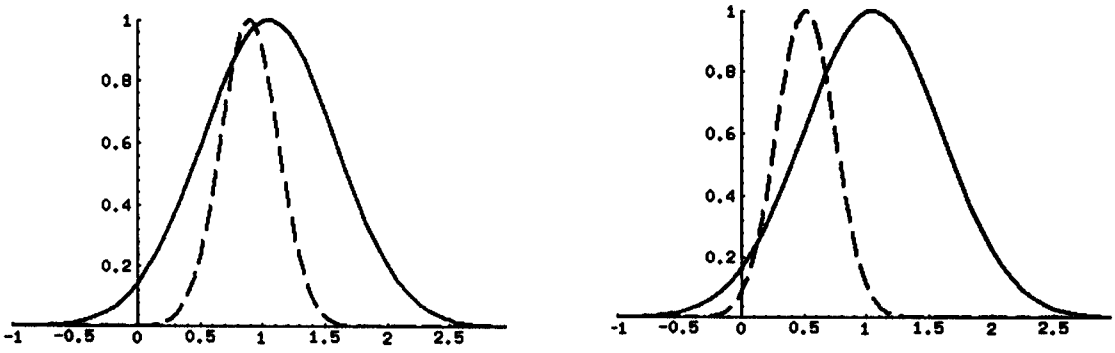


Figure 15. Evolution des fonctions d'appartenance

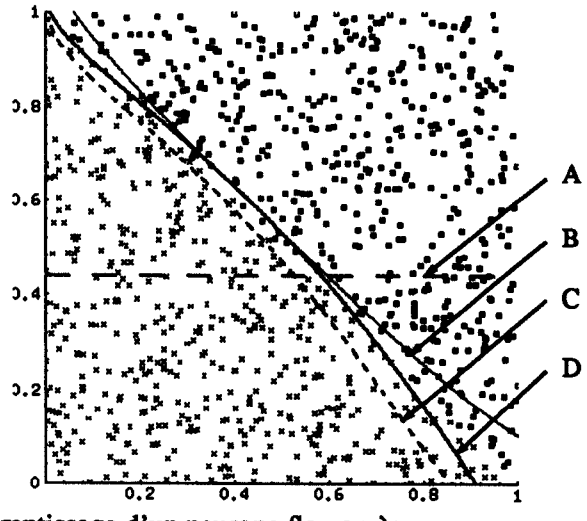


Figure 16. Le résultat d'apprentissage d'un neurone flou après
A. 100, B. 500, C. 1000, D. 10 000 itérations

L'evolution de la fonction apprise est montrée sur la Figure 16. Au lieu de présenter les courbes de changement de paramètres des fonctions d'appartenance, on trouve que c'est plus intéressant de montrer l'évolution des fonctions d'appartenance (Figure 15). Au début d'apprentissage, leurs paramètres sont des valeurs aléatoires entre 0 et 1 et à la fin se stabilisent.

Il est clair qu'un neurone isolé ne peut pas apprendre de fonctions plus compliquées. Comme on le sait, s'est aussi le cas avec le neurone classique.

3. 4 Comparaison entre un neurone classique et un neurone flou

Les performances d'un neurone classique et d'un neurone flou dans le cas d'une séparations linéaire sont montrées à la Table 1. Le neurone classique est plus efficace car sa structure est linéaire et donc mieux adaptée au problème.

No. d'itérations	Neurone classique	Neurone flou
100	9%	26%
500	4.2%	4.4%
1000	4.1%	2%
10 000	0.05%	1.26%

TABLE 1. **Performances d'un neurone classique et d'un neurone flou**

3. 5 Future directions

Ce qui nous intéresse c'est d'incorporer un neurone flou dans un réseau de neurones et d'essayer de représenter un système basé sur la logique floue avec un tel réseau et d'effectuer l'apprentissage de fonctions compliquées. Le problème qui se pose dans une telle conception est de définir le nombre de neurones et leur nature. Si on suppose que le neurone présenté ci-dessus exprime bien le processus de fuzzification on doit concevoir les neurones qui peuvent exprimer une inférence floue.

Il faut concevoir un neurone dont la structure se formalise en une règle linguistique du type:

Si x_1 est X_1 et x_2 est X_2 alors y est Y_1

Chaque règle a deux parties:

- la partie antécédente, exprimée par ***si*** ... et
- la partie conséquente exprimée par ***alors*** ...

Le neurone décrit donne le résultat de la partie antécédente de la règle. La sortie de ce neurone est la valeur avec laquelle il faut prendre en compte la partie conséquente. Le noyau de ce neurone est en faite, l'expression d'une T-norme et l'opération de pondération est l'attribution des valeurs d'appartenance à une valeur concrète (fuzzification).

4. Références

[Bla92] F. Blayo, Réseaux neuronaux, multicopié no. 0604.01 EPFL, Lausanne 1992

[Gup91] M.M.Gupta, J.Qi, On Fuzzy Neuron Models, *Int. Joint Conf. on Neural Networks, IEEE+INNS*, Seattle, July 91, p. II 431 - II 435

[Tou92] C. Touzet, Les réseaux de neurones artificiels - Introduction au connexionnisme, *Collection de l'EERIE*, Nîmes 1992