Chapitre III

Commande par les Réseaux de Neurone Floue

III.1 Réseaux neuro-flou

Jusqu'aux années 90, ces deux techniques (neuronal et flou) apparaissaient comme des approches bien distinctes, ayant chacune leurs avantages et leurs inconvénients et leurs domaines spécifiques.

Dans le domaine du contrôle des processus, la commande floue se prêtait bien à un interfaçage avec un opérateur humain en raison de sa capacité à traiter des termes linguistiques. Par contre, il subsistait toujours une certaine part d'arbitraire dans le choix des fonctions d'appartenance et d'imprécision dans l'écriture des règles qui conduisait à une solution qui peut être assez loin de l'optimum.

D'un autre côté, les réseaux neuromimétiques (réseaux multicouches) étaient tout indiqués dans les cas où l'on ne dispose que de données numériques pour élaborer la commande. Ces réseaux apprenaient par l'exemple en minimisant une fonction coût ce qui leur conférait de bonnes qualités de généralisation. Par contre, il était très difficile d'y introduire de la connaissance *a priori* car le contenu du réseau n'était pas interprétable (boîte noire).

Elles présentent également des points communs :

- Les deux méthodes, appliquées au contrôle, présentent de bonnes qualités de robustesse et sont capables d'extrapolation et de généralisation.

- Aucune des deux méthodes ne nécessite de modèle mathématique du système à contrôler et elles peuvent donc toutes deux s'appliquer à des systèmes non linéaires sans complications particulières.

L'idée est donc apparue tout naturellement au début des années 90 de créer un système d'inférence floue optimisé, cherchant à utiliser les méthodes d'apprentissage supervisé, comme par exemple la rétropropagation du gradient, pour optimiser automatiquement certains paramètres d'un système d'inférence floue.

Dans ce qui suit, nous allons présenter trois architectures neuro-flou qui permettent d'optimiser un système d'inférence flou : l'architecture Nomura, l'architecture LSC et l'architecture ANFIS.

III.2 L'architecture Nomura

L'architecture Nomura est une méthode d'autoparamétrage d'un système d'inférence flou dans laquelle est abordé le concept " neuro-flou". Elle fut présentée par les scientifiques japonais Hiroyoshi Nomura [Nomura 91], Isao Hayashi et Noburu Wakami.

Contrairement au perceptron multicouche standard, le réseau n'est pas entièrement connecté et chaque couche du réseau a une fonction particulière, à laquelle correspond une fonction d'activation spécifique.

Ce réseau présente une complète analogie structurelle avec un système d'inférence floue de type "Sugeno" d'ordre zéro (i.e. dans lequel les conclusions des règles sont nettes). Les fonctions d'appartenance sont triangulaires ce qui a conduit au développement d'un algorithme spécifique.

Nous raisonnerons sur un exemple (qui peut facilement se généraliser) de système d'inférence floue à deux entrées e et Δe dont la connaissance s'exprime sous la forme de règles floues de type :

Règle i : Si e est Ai1 et Δe est Ai2 alors u est Wi

Dans lesquelles les conclusions sont des valeurs exactes (non floues), notées Wi. La prise de décision est faite à l'issue des étapes suivantes:

Pour une entrée donnée $(e, \Delta e)$, on calcule les différents degrés d'appartenance aux sous-ensembles flous Ai1 ou Ai2 apparaissant dans la partie prémisse des règles. C'est l'étape de fuzzyfication.

Pour chaque règle, on calcule la valeur de vérité α_i telle que:

α_i =ET((x1 est A1),(x2 est A2))

Enfin, il faut agréger les règles et tirer la conclusion. Les valeurs Wi des parties conclusion des règles sont pondérées par les valeurs de vérité des prémisses:

$$y = \frac{\sum_i \alpha_i \times W_i}{\sum_i \alpha_i} \qquad \text{(III.1)}$$

Chacune de ces étapes sera réalisée par les neurones d'une des couches du système d'inférence floue (figure II.1).

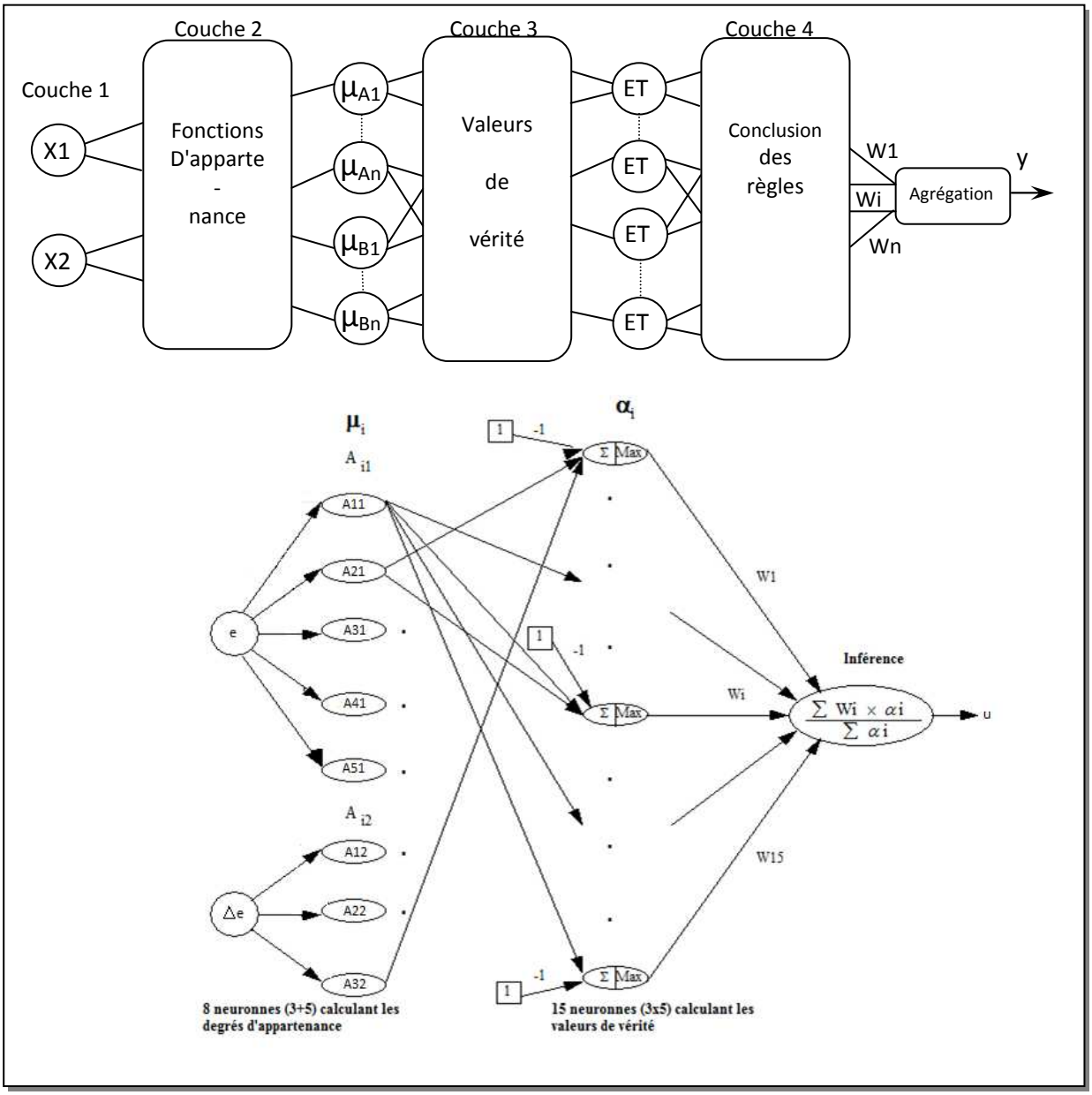

Figure III.1 Représentation du système d'inférence floue

III.2.1 Fonctions des différentes couches

- La première reçoit les entrées du réseau.
- La deuxième couche calcule les degrés d'appartenance des entrées aux diverses fonctions d'appartenance.
- La troisième couche calcule les valeurs de vérité à l'aide des poids entre les deux couches cachées (ils définissent l'opérateur ET choisi).
- La quatrième couche est la couche de sortie. Elle fait l'agrégation des différentes règles et calcule la valeur de sortie. Les poids W_i du réseau entre la troisième et la dernière couche constituent la partie conclusion des règles.

III.2.2 Réalisation des fonctions d'appartenance

Les fonctions d'appartenance de la prémisse proposées par Nomura sont de type triangulaire caractérisées par leur centre a_{ij} pour positionner la fonction sur l'univers du discours, et la largeur b_{ij} pour caractériser son étendue sur cet univers (figure III.6).

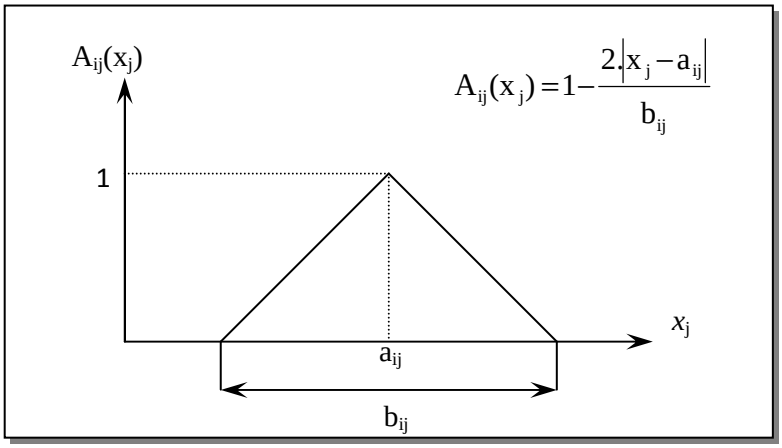

Figure III.2 Fonction d'appartenance utilisée

III.2.3 Calcul des valeurs de vérité α_i

La conjonction floue utilisée dans le réseau de Nomura est celle définie par Lukasiewicz comme suit :

$$ET(X,Y)= \max [0,\ \mu A(x)+\mu B(x)-1] \qquad \text{(III.2)}$$

III.2.4 Algorithme d'apprentissage

C'est une méthode extrêmement générale basée sur la descente du gradient, où chaque paramètre T_i est modifié par rétropropagation en fonction de sa part dans l'erreur globale E :

$$E = \frac{1}{2}(y - y_r)^2 \qquad \text{(III.3)}$$

$$T_i(k+1) = T_i(k) - \eta\ (\partial E / \partial T_i) \qquad \text{(III.4)}$$

$$a_{ij}(t+1) = a_{ij}(t) - K_a \frac{\partial(E)}{\partial(a_{ij})} \qquad \text{(III.5)}$$

$$b_{ij}(t+1) = b_{ij}(t) - K_b \frac{\partial(E)}{\partial(b_{ij})} \qquad \text{(III.6)}$$

$$W_i(t+1) = W_i(t) - K_w \frac{\partial(E)}{\partial(W_i)} \qquad \text{(III.7)}$$

Après dérivation on obtient :

$$a_{ij}(t+1) = a_{ij}(t) - \frac{K_a.\mu_i}{\sum_{i=1}^{n}\mu_i}.(y - y_r) - (W_i(t) - y).\text{sgn}(x_j - a_{ij}(t)).\frac{2}{b_{ij}(t).A_{ij}(x_j)} \quad \text{(III.8)}$$

$$b_{ij}(t+1) = b_{ij}(t) - \frac{K_b.\mu_i}{\sum_{i=1}^{n}\mu_i}.(y - y_r) - (W_i(t) - y).\frac{1 - A_{ij}(x_j)}{A_{ij}(x_j)}.\frac{1}{b_{ij}(t)} \quad \text{(III.9)}$$

$$W_i(t+1) = W_i(t) - \frac{K_w.\alpha_i}{\sum_{i=1}^{n}\mu_i}.(y - y_r) \quad \text{(III.10)}$$

Il y'a d'autre travaux qui utilise le produit pour le calcule le degré de vérité α_i dans l'architecture de Nomura, soit le contrôleur neuro-floue qui contient deux entré (e, Δe) avec deux fonction d'appartenance de type triangle pour chaque entré, et deux sous ensemble floue (small, Large)

Régle R1: If e est Small and Δe is Small then y is b_1 ,
Régle R2: If e est Small and Δe is Large then y is b_2 ,
Régle R3: If e est Large and Δe is Small then y is b_3 ,
Régle R4: If e est Large and Δe is Large then y is b_4 ,

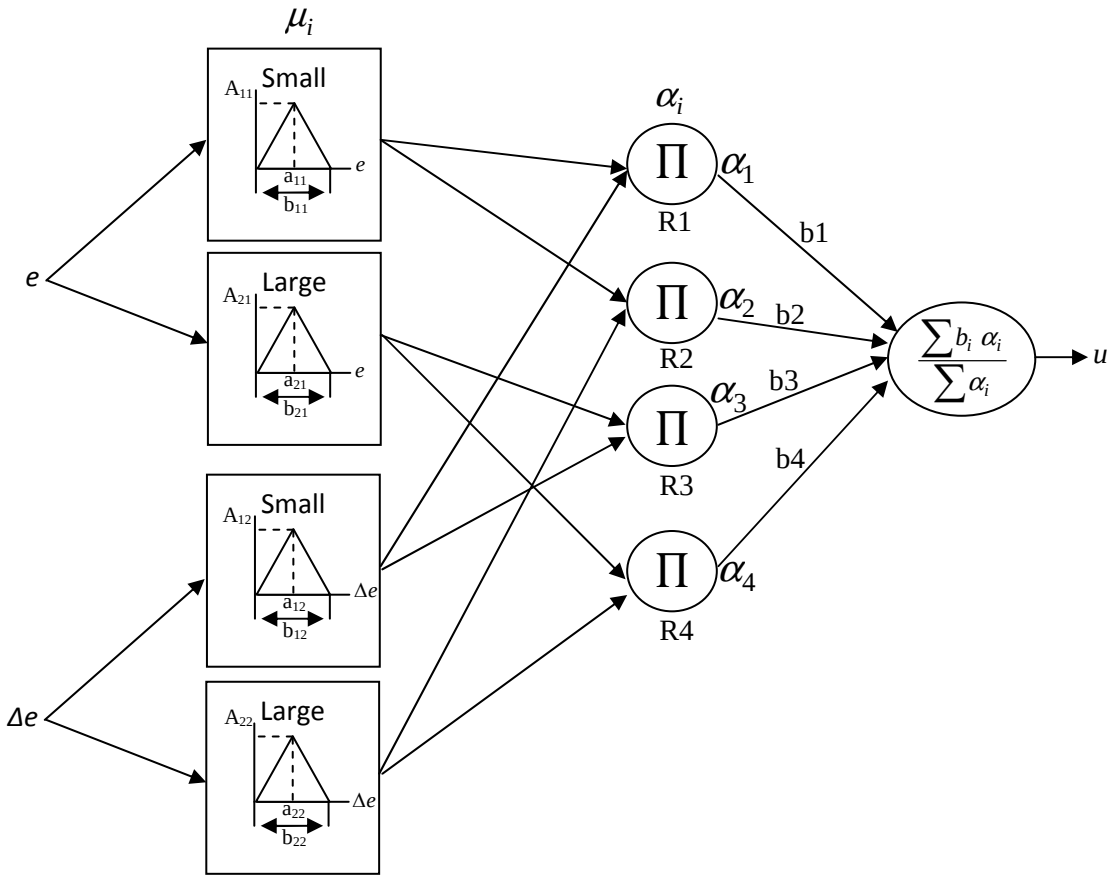

Figure III.3 Représentation du système d'inférence floue

L'adaptation des paramètres des fonctions d'appartenances des prémisses et des conclusions est donnée par les lois suivantes :

$$a_{ij}(t+1) = a_{ij}(t) - K_a \frac{\partial(E)}{\partial(a_{ij})} \quad \text{(III.5)}$$

$$b_{ij}(t+1) = b_{ij}(t) - K_b \frac{\partial(E)}{\partial(b_{ij})} \quad \text{(III.6)}$$

$$b_i(t+1) = b_i(t) - K_w \frac{\partial(E)}{\partial(W_i)} \quad \text{(III.7)}$$

III.3 L'architecture LSC

L'architecture LSC vient du nom du laboratoire Systèmes Complexes de l'Université d'Evry Val d'Essonne qui est à l'origine de sa proposition [Benreguieg 97]. Elle ressemble à l'architecture Nomura, la différence subsiste au niveau de l'algorithme d'apprentissage qui est effectué entièrement en ligne en minimisant une fonction coût *E* pour générer les paramètres w_i de la partie conclusion des règles et les ajuster. Les fonctions d'appartenance de la prémisse proposées par cette architecture sont de type gausienne définies par la relation (III.8):

$$\mu_{ij}(x_i) = \exp\left\{\frac{-(x_i - c_{ij})^2}{2\sigma_{ij}}\right\} \quad \text{(III.8)}$$

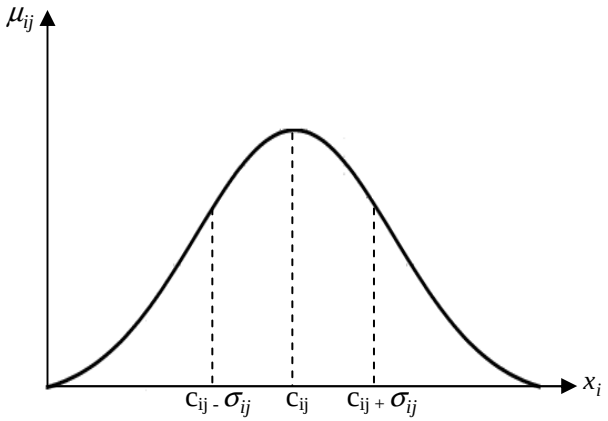

Figure III.4 Représentation du système d'inférence floue

Soit le contrôleur neuro-floue qui contient deux entré (*e*, Δ*e*) avec trois fonction d'appartenance de type gaussienne pour chaque entré, et neuf fonction d'appartenance de type singleton en sortie l'architecture du réseau et présenté par la figure suivante :

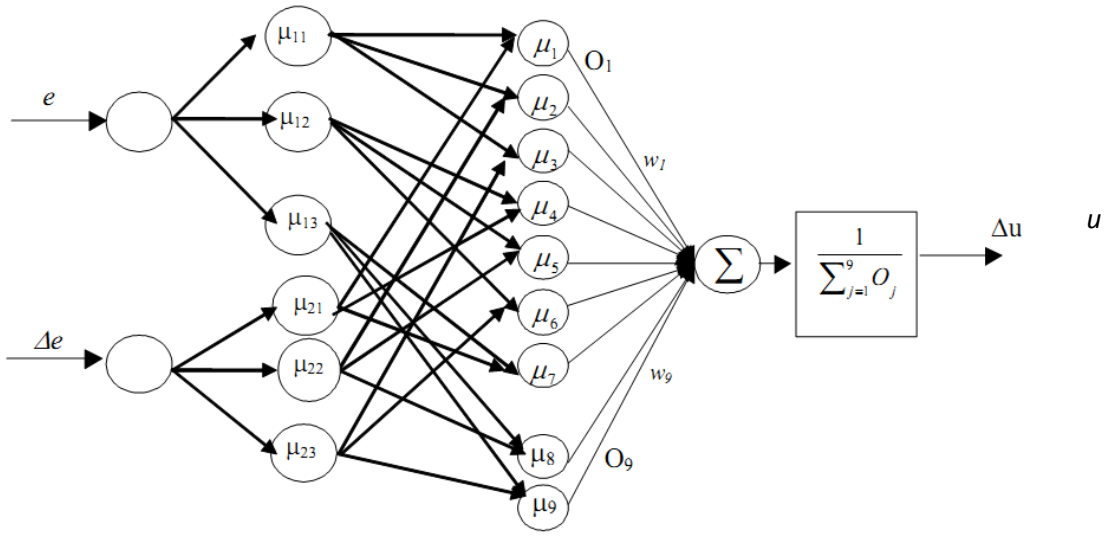

Figure III.5 Représentation du système d'inférence floue

L'apprentissage est basé sur la méthode de la descente du gradient, où chaque paramètre w_i est modifié par rétropropagation en fonction de sa part dans l'erreur globale E :

$$E = \frac{1}{2}(y - y_r) \qquad \text{(III.9)}$$

$$W_i(t+1) = W_i(t) - \frac{K_w.O_i}{\sum_{k=1}^{9} O_k}.(y - y_r) \qquad i = 1,2,....,9 \qquad \text{(III.10)}$$

III.4 L'architecture ANFIS

Cette architecture appartient à une classe de réseaux adaptatifs RBF [Jang 93a] fonctionnellement équivalent à un système d'inférence floue de type Sugeno, connue sous le nom de ANFIS (Adaptive-Network-based Fuzzy Inference System) [Jang93b]. Cette technique applique la méthode des moindres carrée combinée à la rétropropagation du gradient dans l'apprentissage.

Pour simplifier la compréhension et sans perte de généralité, nous considérons un système à deux entrées x_1 et x_2 et une sortie y. Considérons aussi un modèle flou de TS1 de ce système, composé des deux règles suivantes :

Si x_1 est A_1 **et** x_2 est B_1 **alors** $y_1 = F_1(x_1,x_2) = a_1x_1+b_1x_2+c_1$ (III.11)

Si x_1 est A_2 **et** x_2 est B_2 **alors** $y_2 = F_2(x_1,x_2) = a_2x_1+b_2x_2+c_2$ (III.12)

Jang a proposé de représenter cette base de règles par le réseau adaptatif de la figure (III.7)

La méthode ANFIS est basée sur l'utilisation de réseaux multicouches (adaptative networks) où chaque cellule réalise une fonction propre respectant les paramètres qui lui ont

été fournis. L'application de l'exemple précédent peut donc être représentée de la manière suivante (figure III.6):

Le réseau adaptatif ANFIS est un réseau multi-couches dont les connexions ne sont pas pondérées, ou ont toutes un poids de 1. Les nœuds sont de deux types différents selon leur fonctionnalité : les nœuds carrés (adaptatifs) contiennent des paramètres, et les nœuds circulaires (fixes) n'ont pas de paramètres. Toutefois chaque nœud (carré ou circulaire) applique une fonction sur ses signaux d'entrées. La sortie O_i^k du nœud i de la couche k (appelé nœud (i,k)) dépend des signaux provenant de la couche $k–1$ et des paramètres du nœud (i,k), c'est à dire,

$$O_i^k = f(O_1^{k-1}, \ldots, O_{n_{k-1}}^{k-1}, a, b, c, \ldots) \qquad \text{(III.13)}$$

où n_{k-1} est le nombre de nœuds dans la couche $k-1$, et $a, b, c, \ldots$ sont les paramètres du nœud (i,k). Pour un nœud circulaire, ces paramètres n'existent pas.

Dans le réseau de la figure (III.6), les nœuds d'une même couche ont des fonctions issues d'une même famille que nous explicitons ci-dessus :

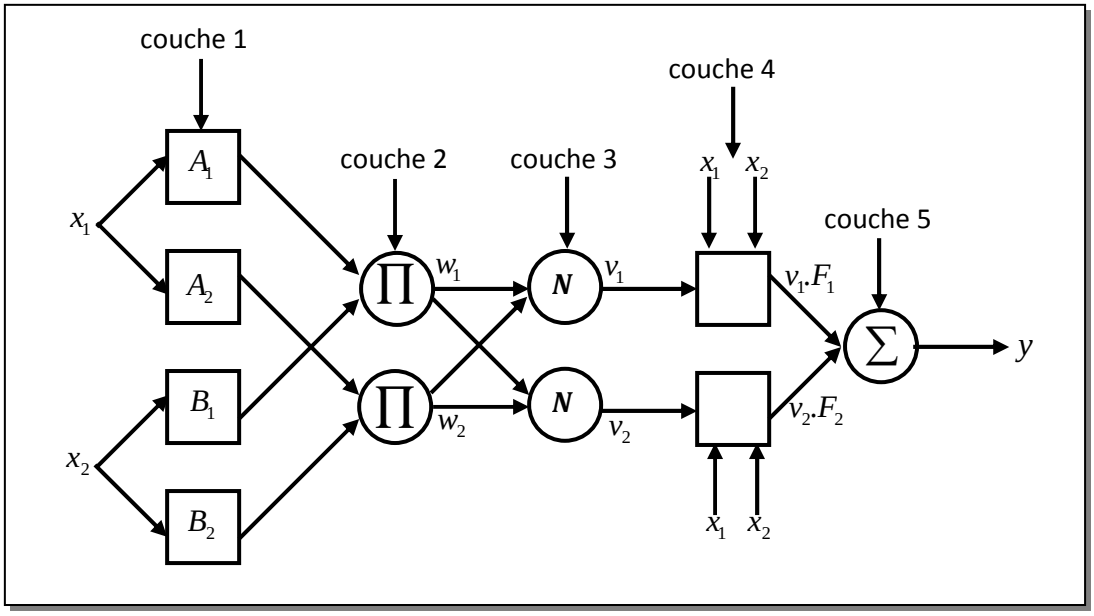

Figure III.6 Réseau ANFIS équivalent au raisonnement flou cité

Couche 1 : Chaque nœud de cette couche est un nœud carré avec une fonction :

$$O_i^1 = \mu_{A_i}(x) \qquad \text{(III.14)}$$

où x est l'entrée du nœud i, et A_i le terme linguistique associé à sa fonction. Donc le résultat que donne cette cellule représente le degré avec lequel x satisfait le qualificatif A_i .En d'autres

termes, O_i^1 est le degré d'appartenance de x à A_i. Les paramètres d'un nœud de cette couche sont ceux de la fonction d'appartenance correspondante. Généralement les fonctions d'appartenance utilisées par la méthode ANFIS sont des gaussiennes ou des fonctions cloches.

Couche 2 : Chaque nœud i de cette couche est un nœud circulaire appelé Π qui engendre en sortie le produit de ses entrées. Ce produit représente le degré d'activation d'une règle :

$$w_i = \mu_{A_i}(x_1).\mu_{B_i}(x_2),\ i=1..2 \qquad \text{(III.15)}$$

Couche 3 : Chaque nœud de cette couche est un nœud circulaire appelé N.

Dans cette couche, les cellules calculent le rapport entre les valeurs de vérité de chaque règle par rapport à la somme de toutes les valeurs de vérité données par chaque règle. Elles estiment donc le poids "normalisé" de chaque règle.

$$v_i = \frac{w_i}{w_1 + w_2} \qquad \text{(III.16)}$$

Couche 4 : Chaque nœud de cette couche est un nœud carré avec une fonction réalisant le calcul :

$$O_i^4 = v_i.F_i = v_i\,(a_i x_1 + b_i x_2 + c_i),\ \ i=1..2 \qquad \text{(III.17)}$$

où v_i est la sortie de la couche 3, et $\{a_i, b_i, c_i\}$ est l'ensemble des paramètres de sortie de la règle i.

Couche 5 : Le seul nœud de cette couche est un nœud circulaire qui effectue la somme des signaux provenant de la couche 4, c'est à dire,

$$O_1^5 = y = \sum_i v_i.F_i \qquad \text{(III.18)}$$

Nous remarquons que la sortie globale du réseau est équivalente à la sotie du modèle de TS1.

La généralisation du réseau à un système à n entrées ne pose aucun problème particulier. Le Nombre de nœuds de la couche 1 est toujours égal au nombre total de termes linguistiques définis. Le Nombre de nœuds des couches 2, 3 et 4 est toujours égal au nombre de règles floues.

La figure III.7 montre un exemple de réseau ANFIS correspondant à un SIF à deux entrées avec 9 règles. A chaque entrée, correspondent trois fonctions d'appartenance ce qui implique que l'espace d'entrée est partagé en 9 sous-espaces flous où chacun couvre une règle floue.

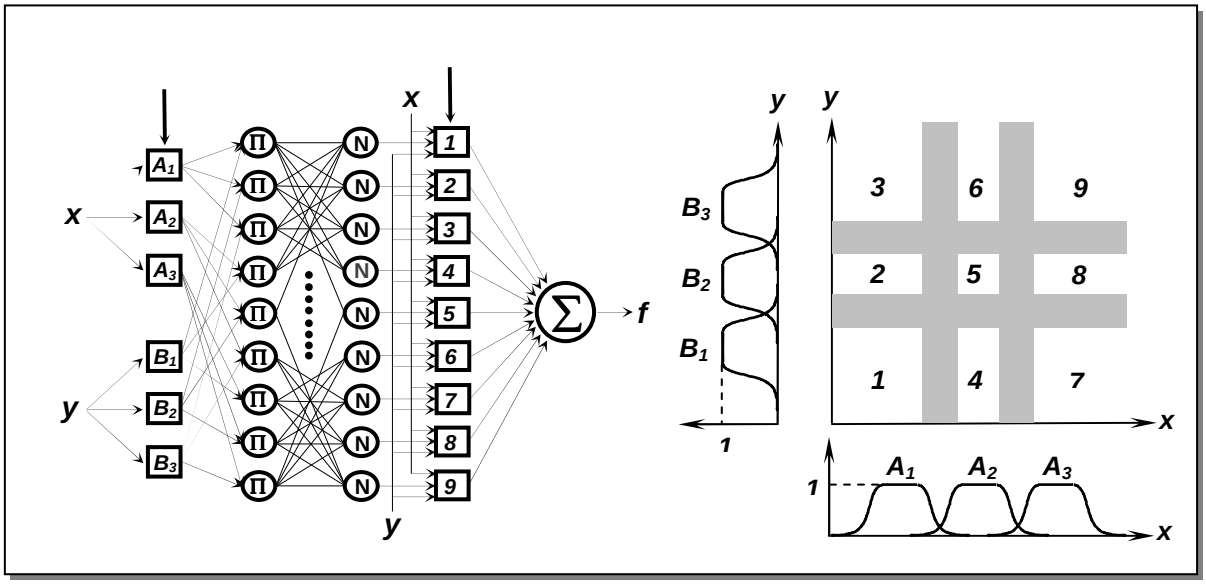

Figure III.6 Une structure ANFIS 2-entrées avec 9 règles

III.4.1 Algorithme d'apprentissage

L'apprentissage à partir d'un ensemble de données concerne l'identification des paramètres des prémisses et des conséquences, la structure du réseau étant fixée. L'algorithme d'apprentissage commence par construire un réseau initial, en suite nous appliquons une méthode d'apprentissage par rétro-propagation de l'erreur. Jang a proposé d'utiliser une règle hybride d'apprentissage qui combine un algorithme de descente de gradient avec une estimation par moindres carrés (MC).

Dans l'architecture ANFIS, l'apprentissage est basé sur une méthode hybride qui applique la méthode des moindres carrée combinée à la rétropropagation du gradient tout au long de l'apprentissage. Les paramètres des conclusions des règles qui sont linéaires par rapport à la sortie du système d'inférence floue sont ajustés par la méthode des moindres carrées dans le sens "*forward*" du réseau, puis vient l'ajustement des paramètres des prémisses des règles par la méthode du gradient dans le sens "*backward*" du réseau, et ce, pour chaque itération de l'algorithme. Le tableau III.1 nous résume les différentes étapes de l'algorithme :

-	Forward (une fois)	Backward (une fois)
Paramètres de la prémisse	Fixés	Descente du gradient
Paramètres de la conclusion	Moindres carrées	Fixés

Tablaux.III.1 La méthode hybride utilisé dans ANFIS

L'identification des paramètres non-linéraires de la partie prémisse des règles floues avec des fonctions d'appartenance de type gaussienne définies par la relation (III.18):

$$\mu_{A_i^j}(x_i) = \exp\left\{-0.5\left(v_i{}^j\left(x_i - c_i{}^j\right)\right)^2\right\} \quad \text{Avec } v_i{}^j = \frac{1}{\sigma_i{}^j} \tag{III.18}$$

où *c* est la moyenne, *v* est l'inverse de la variance (cette dernière est adoptée afin d'éviter la division par zéro lors de son adaptation) s'effectue comme suit:

Considérons l'ensemble des paramètres des prémisses caractérisé par le vecteur θ, et soit un ensemble de données entrée-sortie $(\underline{x}(k), y_d(k))$. Notre objectif est de trouver les valeurs du vecteur θ pour que la sortie du système flou approche le mieux possible la sortie désirée $y_d(k)$, en minimisant le critère :

$$J = \frac{1}{2}\sum\left(f(\underline{x}(k), \underline{\theta}) - y_d(k)\right)^2 \tag{III.20}$$

L'algorithme de rétro-propagation est donné par:

$$\underline{\theta}(k+1) = \underline{\theta}(k) - \lambda\frac{\partial J}{\partial \theta} \tag{III.21}$$

avec λ est le gain d'apprentissage.

Cet algorithme est souvent appelé algorithme de rétro-propagation du gradient, utilisé dans les réseaux de neurones. Cependant, ce dernier est connu par ses inconvénients d'être souvent lent et peut être piégé par des minima locaux lors de l'apprentissage.

Dans un système flou de type Sugeno, nous avons $\theta = [c \;\; v]^T$, où [c, v] sont les paramètres des prémisses, nous avons donc:

$$\frac{\partial J}{\partial \theta} = \left[\frac{\partial J}{\partial c} \;\; \frac{\partial J}{\partial v}\right]^T \tag{III.22}$$

En utilisant l'expression de la sortie du système flou, il vient:

$$\frac{\partial J}{\partial c_i^j} = \sum_{k \in I_{c_i^j}} \frac{\partial J}{\partial \mu_k}\frac{\partial \mu_k}{\partial \mu_{A_i^j}}\frac{\partial \mu_{A_i^l}}{\partial c_i^j} \tag{III.23}$$

où $I_{c_i^j}$: est l'ensemble des indices (*l*) des règles floues (R_l) dont lesquelles apparaît l'ensemble flou A_i^j.

$$\frac{\partial J}{\partial \mu_k} = \frac{\mu_k(y_k - y_d)}{\sum_{l=1}^{M}\mu_l} \tag{III.24}$$

$$\frac{\partial \mu_k}{\partial \mu_{A_i^j}} = \prod_{\substack{l=1 \\ l \neq i}}^{n} \mu_{A_l^{kl}}(x_l) \tag{III.24}$$

Pour une fonction d'appartenance gaussienne équation (III.18), on a :

$$\frac{\partial \mu_{A_i^j}}{\partial c_i^j} = v_i^{j\,2}(x_i - c_i^j)\mu_{A_i^j}(x_i) \tag{III.25}$$

Après calcul, il vient :

$$\frac{\partial J}{\partial c_i^j} = \frac{v_i^{j\,2}(x_i - c_i^j)\sum_{k \in I_{c_i^j}} \mu_k (y_k - y_d)}{\sum_{l=1}^{M} \mu_l} \tag{III.26}$$

De la même manière, on trouve :

$$\frac{\partial J}{\partial v_i^j} = \frac{-v_i^j (x_i - c_i^j)^2 \sum_{k \in I_{c_i^j}} \mu_k (y_k - y_d)}{\sum_{l=1}^{M} \mu_l} \tag{III.27}$$