Les réseaux de neurones

Chapitre II

Depuis une vingtaine d'année, on observe une augmentation importante du nombre de satellites dédiés à l'observation de la Terre. Ces nombreux satellites fournissent d'importantes quantités de données permettant, une amélioration de la connaissance de notre environnement qu'il s'agisse de l'atmosphère, du sol ou de l'océan et ainsi, une meilleur compréhension des mécanismes qui régissent le climat. Un problème qui se pose à l'heure actuelle est la définition et la mise en place de méthodes numériques performantes pour le traitement de ces masses de données. Ce problème est abordé par des méthodes statistiques afin de restituer les paramètres d'environnement à partir des observations de télédétection. Les réseaux de neurones entrent dans ce cadre. Les réseaux tels que nous les utiliserons sont à classer parmi les techniques de régression non linéaire.

Les techniques neuronales que nous allons présenter dans ce chapitre ne couvrent qu'un domaine restreint des applications potentielles des réseaux de neurones, elles se limitent à des applications d'approximation de fonction par apprentissage supervisé. Les grands domaines d'application des réseaux de neurones comprend aussi, la modélisation de processus dynamique non linéaire (par exemple : prédictions financières, prédiction de consommation ou encore détection d'anomalies de fonctionnement), la reconnaissance de formes (par exemple : lecture des codes postaux, lecture des montants des cheques), et la commande de processus (pilotage « neuronal » automatique de véhicule (Rivals, 1995)). Une vue générale des réseaux de neurones et de leurs applications est donnée par exemple dans le livre de Bishop (1995).

L'organisation de ce chapitre est la suivante : nous allons décrire dans un premier temps le Perceptron MultiCouche (PMC) défini par Rumelhart *et al.* (1986). Nous décrirons ensuite les réseaux Multi-Expert (ME) définie par Jordan *et al.* (1994). Ces deux variétés de réseaux sont utilisées dans la suite de ce travail. Nous détaillerons pour chacun d'entre eux le concept d'approximation de fonction en rappelant leurs principales propriétés statistiques théoriques.

II.1 Préambule.

C'est en 1943 que W.S McCulloch et W. Pitts donne naissance au concept de réseaux de neurones formels. 'Comprendre les mécanismes à l'origine des fonctions supérieures du cerveau est l'objet de recherches au carrefour de la neurobiologie, de la psychologie, de l'informatique et de la physique. Dans ce vaste champ d'étude, le domaine particulier des réseaux de neurones s'est tout particulièrement développé durant les années 1980.' [Nadal, 1993].

Néanmoins, le terme de « neurones » ne doit pas faire illusion ; les succès récent des réseaux de neurones formels ne doivent rien à la biologie, mais sont dus exclusivement à une meilleur compréhension des propriétés mathématiques fondamentales de ces réseaux. Nous sommes encore extrêmement éloignés de la réalisation de machine susceptibles de reproduire une partie, même infime, des capacités de calcul des systèmes nerveux les plus simple.

A l'heure actuelle, les réseaux de neurones formels doivent être considérés comme des extensions puissantes de techniques statistiques « classique ».

Définition :

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.

II.2 Le Perceptron MultiCouche (PMC)

II.2.1 Description d'un neurone formel

Un neurone formel ou artificiel est un opérateur mathématique très simple. Un neurone possède des entrées qui peuvent être les sorties d'autres neurones, ou des entrées de signaux extérieures, et une sortie. La valeur de la sortie résulte du calcul de la somme des entrées, pondérées par des coefficients (dits poids de connexions ou poids synaptiques) et du calcul d'une fonction non linéaire (dite fonction d'activation) de cette somme pondérée. L'état du neurone, appelé aussi activité, est définie comme la somme pondérée de ses entrées. Son schéma de fonctionnement est donné en Figure II-1. L'information est ainsi transmise de manière unidirectionnelle. Un neurone se caractérise par trois concepts : son état, ses connexions avec d'autres neurones et sa fonction d'activation.

Nous utiliserons par la suite les notations suivantes.

S_i :l'état du neurone *i*.

f_i :la fonction d'activation associée au neurone *i*.

W_{ij} :le poids de la connexion entre les neurones *j* et *i*.

W_{i0} :le poids de la connexion entre le neurone biais (+1) et les neurones *i*.

Ainsi, le neurone *i* recevant les informations de *p* neurones effectue l'opération suivante :

$$S_i = f_i\underbrace{\left(\sum_{j=1}^{p} W_{ij} S_j - W_{i0}\right)}_{x} \qquad (1)$$

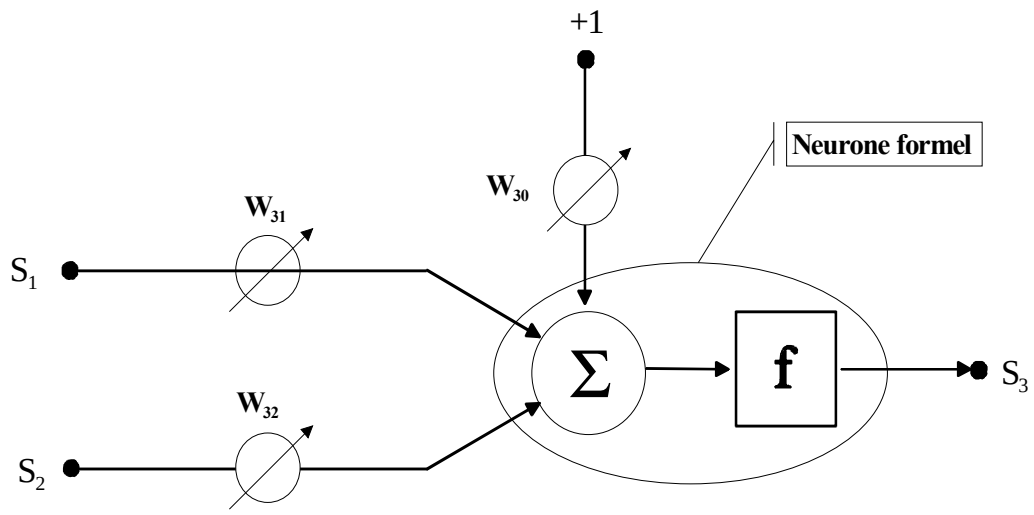

Figure II-1. Schéma de fonctionnement d'un neurone formel.

Pour le PMC, on utilise le plus souvent les fonctions d'activations suivantes :

la **fonction identité** : $f(x) = x$

les neurones dont la fonction d'activation est la fonction linéaire sont appelés neurones linéaires

la **fonction sigmoïde** : $f(x) = \dfrac{\exp(x)-1}{\exp(x)+1} = \tanh(x/2)$

c'est la plus utilisée car elle introduit de la non linéarité, mais c'est aussi une fonction continue, différentiable et bornée. La fonction sigmoïde a des asymptotes horizontales en $-\infty$ et en $+\infty$. Elles permettent d'éviter que ne se propagent des valeurs trop grandes dans le réseau (Figure II-2). La fonction d'activation peut également être une gaussienne, un échelon, etc.

L'utilisation des fonctions d'activation non linéaires permet l'obtention de modèles statistiques non linéaires. Les réseaux multicouches qui utilisent comme fonction d'activation les sigmoïdes, sont appelés réseaux multicouches quasi linéaires.

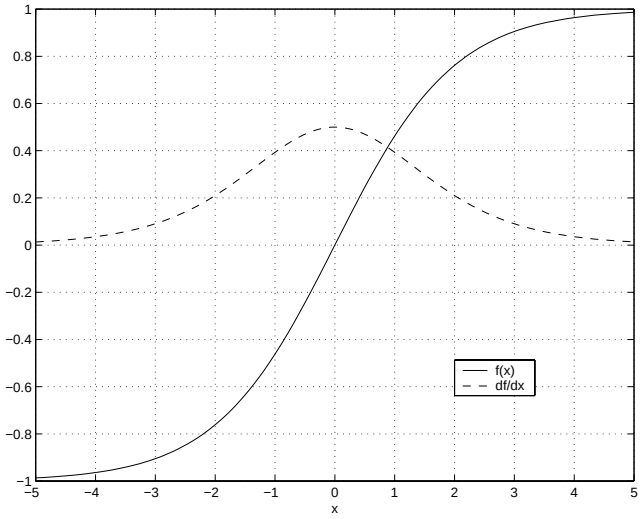

Figure II-2. Fonction de transfert sigmoïde et sa dérivée première.

II.2.2 Architecture des PMC

Le PMC a une structure bien particulière : ses neurones sont organisés en couches successives comme présenté sur la Figure II-3. Chaque neurone d'une couche reçoit des signaux de la couche précédente et transmet le résultat à la suivante, si elle existe. Les neurones d'une même couche ne sont pas interconnectés. Un neurone ne peut donc envoyer son résultat qu'à un neurone situé dans une couche postérieure à la sienne. L'orientation du réseau est fixée par le sens, unique, de propagation de l'information, de la couche d'entrée vers la couche de sortie. Pour les réseaux considérés, les notions de couches d'entrée et de sortie sont donc systématiques. Ces dernières constituent l'interface du réseau avec l'extérieur. La couche d'entrée reçoit les signaux (ou variables) d'entrée et la couche de sortie fournit les résultats. Enfin, les neurones des autres couches (couches cachées) n'ont aucun lien avec l'extérieur et sont appelés neurones cachés.

Par convention, les neurones d'entrée ont toujours une fonction d'activation « identité », laissant passer l'information sans la modifier. En ce qui concerne le neurone de sortie, on peut lui associer une fonction d'activation linéaire ou non, dérivable ou non, suivant la nature du problème à résoudre. En ce qui concerne la fonction d'activation associée aux neurones cachés, on utilise dans le cadre de cette thèse une fonction d'activation de la famille des sigmoïdes.

Le perceptron multicouches décrit Figure II-3 comporte *p* unités en entrée, recevant respectivement *p* variables $\{X_1, X_2,..., X_p\}$, et une seule unité de sortie, qui produit la variable *Y*. Ce modèle réalise une application de $\Re^p$ dans $\Re$. L'architecture du réseau, déterminée par le schéma de connexion des neurones, fige ainsi une composition de fonction élémentaire et représente une famille *G(.,W)* de fonctions non linéaires et dont les paramètres sont les poids de connexions du réseau *W*.

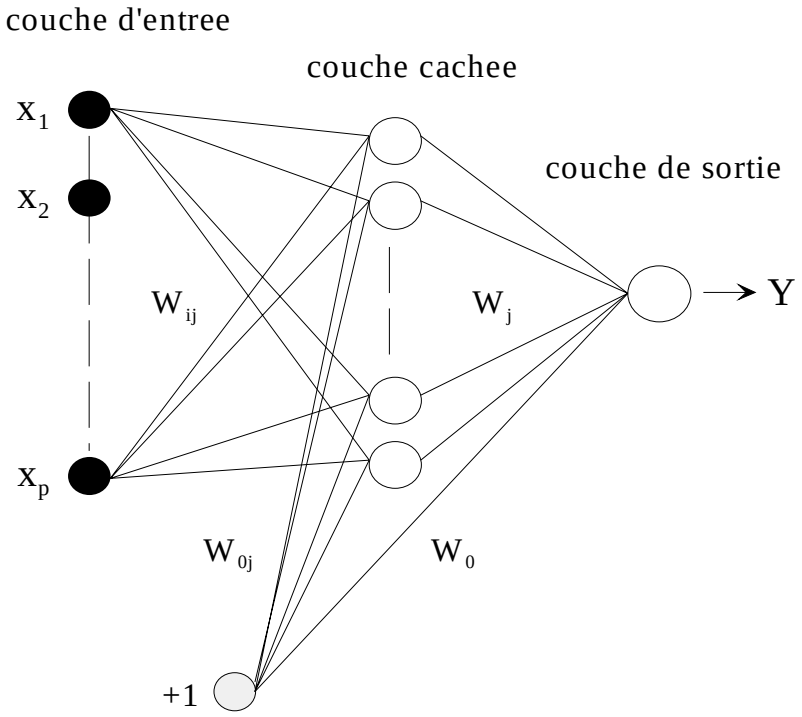

Figure II-3. Architecture d'un PMC à une couche cachée (n neurones cachés sigmoïdes) avec p neurones d'entrée et (q=1) neurone de sortie linéaire.

La sortie du réseau aura une expression de la forme suivante dépendant du nombre de couches le composant :

$$Y = \sum_{i=1}^{n} W_i . f_i (\sum_{j=1}^{p} W_{ij} . X_j + W_{i0}) + W_0 \qquad (2)$$

II.2.3 Approximation de fonction

La détermination d'une fonction neuronale à partir d'observations se ramène au problème classique de la détermination des paramètres du modèle (les poids de connexions) par régression au sens des moindres carrés. Pour construire cette fonction, deux choses sont

indispensables : un ensemble de fonctions candidates (définie par l'architecture du réseau), parmi lesquelles on va rechercher celle qui nous intéresse, et un critère d'apprentissage permettant d'en choisir une. Le but de l'apprentissage est donc de déterminer un modèle qui va généraliser un processus décrit par un nombre fini *N* de comportements qui composent l'ensemble d'apprentissage.

II.2.4 Apprentissage

Approximer une fonction *T* de $\Re^p$ dans $\Re^q$ à l'aide d'un PMC revient à utiliser une fonction *g* choisie au sein d'une famille *G(., W)* associée à une architecture de réseau dont les couches d'entrée et de sortie comportent respectivement *p* et *q* neurones.

$$\begin{aligned} A \in \Re^p &\rightarrow \quad B \in \Re^q \\ \vec{x} &\rightarrow \quad \vec{y} = G(\vec{x}, W) \end{aligned}$$

La famille $G(.,W)$ est donc un système paramétré qui associe un espace de sortie $B \in \Re^q$ à un espace d'entrée $A \in \Re^p$. Approximer *T* à partir de $G(.,W)$ revient ainsi à rechercher la fonction $g \in G(.,W)$ telle que

$$g(\vec{x}) = G(\vec{x}, W^*) \approx T(\vec{x}) \quad \forall \vec{x} \in A$$

où *W* représente l'ensemble des poids du réseau.

Le nombre de couches cachées et le nombre de neurones des couches cachées sont à déterminer de manière optimale suivant la difficulté de la fonction à approximer et l'ensemble d'apprentissage dont on dispose. Le choix de l'architecture du réseau définit de façon implicite la famille de fonction $G(.,W)$.

L'étape suivante est la détermination des poids optimaux *W** et donc de la fonction *g* dans $G(.,W)$ qui approche au mieux la fonction *T* étudiée. Ces poids sont déterminés par un algorithme dit d'apprentissage qui correspond à la phase d'estimation des paramètres du modèle. Cette détermination se fait à partir de *N* exemples $((\vec{x}_k, \vec{y}_k), k = 1, N)$ qui décrivent la fonction recherchée. La fonction $g(\vec{x})$ obtenue en fin d'apprentissage est continue, elle permet donc d'interpoler la fonction entre les *N* points utilisés durant l'apprentissage.

L'algorithme d'adaptation des poids est connu sous le nom d'algorithme de rétropropagation du gradient (Rumelhart *et al.*,1986 ; Le Cun *et al.*, 1985). La précision de l'approximation va dépendre de l'ensemble d'apprentissage et donc de la manière dont le problème est décrit par les données.

II.2.5 Approche statistique de l'apprentissage

L'estimation des poids du réseau implique la minimisation au moindre carré de l'erreur (une fonction coût) définie sur la base d'apprentissage. Cette erreur est donnée par :

$$E(W) = \sum_{k=1}^{N} \left[G(\vec{x}_k, W) - y_k \right]^2 \qquad (3)$$

en supposant, pour simplifier l'écriture, que l'espace de sortie est de dimension $q = 1$.
Supposons maintenant que la base d'apprentissage est composée d'une infinité d'exemples $(N \to \infty)$, la somme finie dans l'équation (3) est remplacée par une intégrale sur la densité de probabilité jointe $p(y, \vec{x})$:

$$E(W) = \iint \left[G(\vec{x}, W) - y \right]^2 p(y, \vec{x}) dy d\vec{x} \qquad (4)$$

Minimiser l'erreur revient à différentier l'équation (4) par rapport à $G(\vec{x}, W)$:

$$\frac{\delta E}{\delta G(\vec{x}, W)} = 0 \qquad \forall \vec{x} \qquad (5)$$

En substituant (4) dans (5) et en utilisant la règle de Bayes, on obtient l'expression suivante :

$$G(\vec{x}, W^*) = \int y \, p(y \mid \vec{x}) dy \equiv E(y \mid \vec{x})$$

où W^* représente les poids optimaux après apprentissage. $E(y \mid \vec{x})$ est l'espérance mathématique de *y* sachant $\vec{x}$, définie comme la moyenne conditionnelle de *y*, conditionnée par $\vec{x}$.

Dans le cas pratique, où la base d'apprentissage est de dimension finie, on montre de façon similaire que minimiser l'équation (3) conduit à l'obtention d'une bonne estimation de $E(y \mid \vec{x})$. La sortie du MLP est alors telle que :

$$g(\vec{x}) \equiv G(\vec{x}, W^*) \approx E(y \mid \vec{x}) \qquad (6)$$

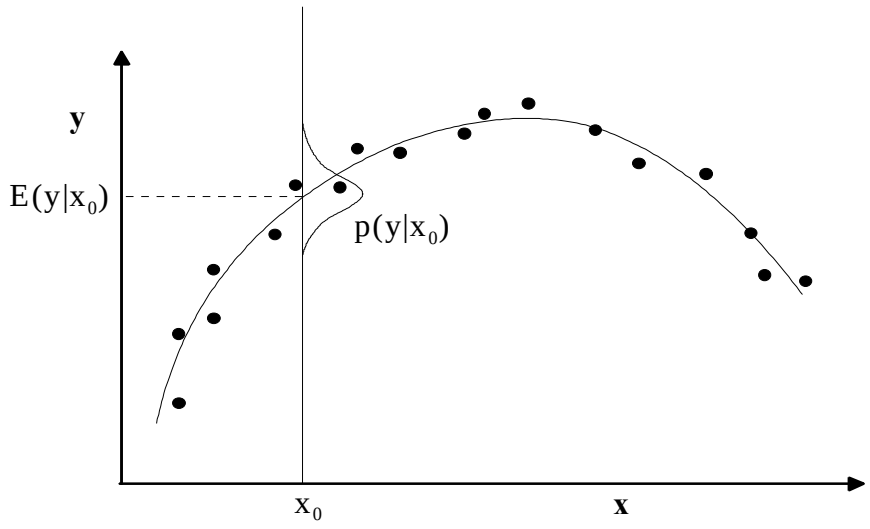

*Figure II-4. Illustration d'un ensemble d'apprentissage (représenté par les points) composé de couples de valeurs (x, y), où x représente la variable d'entrée et y la variable de sortie. La fonction optimale au sens des moindres carrés (*G(x, W*)*) est représentée en trait plein, elle est donnée par la moyenne conditionnelle de la variable de sortie. Pour une valeur donnée de x, tel que la valeur x_0, la fonction $G(x_0, W^*)$ est donnée par la moyenne de y relative à la densité de probabilité $p(y \mid x_0)$.*

L'apprentissage d'un PMC par la méthode des moindres carrées permet l'approximation de deux paramètres statistiques (Bishop, 1994) :

-- la moyenne conditionnelle, fonction de l'entrée x , donnée par $G(\vec{x},W^*)$.

-- la variance moyenne autour de cette moyenne conditionnelle, donnée par le résidu de la fonction erreur (3).

Connaissant ces deux paramètres statistiques, on peut représenter la distribution de la probabilité conditionnelle des sorties par une fonction gaussienne de centre $G(\vec{x},W^*)$ et de variance globale σ déterminée par le résidu de l'erreur (Figure II-4).

II.2.6 Principe de la rétropropagation du gradient

Lorsque que la fonction g est non linéaire, la minimisation de la fonction coût (3) fait appel à des algorithmes itératifs basés sur un calcul du gradient. Un algorithme très simple et devenu célèbre, la rétropopagatrion du gradient, à été proposé par Rumelhart et al.(1986) et LeCun, (1985). Cette algorithme est présenté dans la plupart des livres et thèses consacrés aux réseaux de neurones (par exemple, Tran, 1999), il est basé sur l'idée suivante : au

départ, les poids $\{W_{ij}\}$ sont initialisés à des valeurs aléatoires. Le but de la méthode d'apprentissage est de les faire évoluer de façon à ce que le réseau soit capable, étant donné un vecteur d'entrée, de calculer le bon vecteur de sortie.

L'algorithme de rétropropagation du gradient est un algorithme itératif, les poids sont modifiés à chaque étape selon la règle suivante :

$$W_{ij}(t) = W_{ij}(t-1) + \Delta W_{ij}(t) \tag{7}$$

Les poids à l'itération *t* correspondent aux poids à l'itération *t-1* plus une correction dépendant du signal d'erreur.

Définissons la fonction de coût suivante (algorithme stochastique) :

$$E_N(W) = \sum_{k=1}^{N} E_N^k(W) \tag{8}$$

L'erreur est minimisée à chaque présentation d'un exemple $\vec{x}_k$ en faisant évoluer les $\{W_{ij}\}$ suivant la courbe de plus grande pente sur la surface définie par la fonction d'erreur $E_N^k(W)$. $\Delta W_{ij}(t)$ est proportionnelle à l'opposé du gradient :

$$\Delta W_{ij}(t) = -\eta \frac{\partial E_N^k(\vec{x}_k, W)}{\partial W_{ij}} \tag{9}$$

avec η un facteur choisi de pondération, nommé pas d'apprentissage. En adoptant une formulation quadratique de l'écart, on a :

$$E_N^k(W) = (y_k - G(\vec{x}_k, W))^2 \tag{10}$$

en supposant que la sortie est à une dimension.

II.2.7 Des approximateurs universels

Vingt ans après la publication de l'ouvrage où Minsky et Papert (1969) exposaient les limitations de Perceptron simple, Cybenko *et al.* (1989) et Hornik *et al.* (1989) établissent les réseaux de neurones comme une classe d'approximateurs universels. Il a été ainsi démontré qu'un perceptron multicouches avec une seule couche cachée pourvue d'un nombre suffisant de neurones, peut approximer n'importe quelle fonction avec la précision souhaitée. Néanmoins, cette propriété ne permet pas de choisir, pour un type de fonction donné, le nombre de neurones optimal dans la couche cachée. Autrement dit ce résultat ne mène pas vers une technique de construction d'architecture.

II.2.8 La base d'apprentissage

La base d'apprentissage est une base de données contenant des couples d'entrées-sorties servant à déterminer les valeurs des paramètres d'un réseau de neurones lors de la phase d'apprentissage supervisé. Le PMC est un interpolateur imparfait des observations contenues dans cette base, puisqu'il commet une erreur aux points d'observations. Or, la base d'exemples n'échantillonne jamais l'espace des données de manière parfaite. Il est souhaitable, pour bien contraindre un PMC, que le nombre de contraintes (nombre d'exemples dans la base d'apprentissage) imposées soit très supérieur au nombre de degrés de liberté du réseau (nombre de poids). Le nombre minimum souhaitable d'exemples est lié à la complexité de la fonction à simuler et à l'architecture du réseau choisie. En effet, un bon estimateur g est caractérisé par une bonne précision, c'est-à-dire un faible biais et une bonne stabilité, c'est-à-dire une variance faible. Or, ces deux objectifs sont contradictoires. Geman *et al.* (1992) ont développé l'idée suivante : pour un problème donné et des échantillons de taille fixe, un réseau sous-dimensionné aura un biais important et un terme de variance faible. A contrario, un réseau surdimensionné possédera un grand nombre de degrés de liberté et l'optimisation conduira à des solutions pouvant être très différentes, ce qui correspond à une composante de variance importante. L'idée est donc que le biais diminue et que la variance augmente lorsque la taille du réseau augmente. Il y aurait donc une « zone » de bon compromis correspondant à une bonne taille du réseau pour le problème traité et le nombre d'exemples d'apprentissage. Il est généralement bien accepté qu'il soit nécessaire de disposer d'un échantillon de taille N qui soit au minimum de l'ordre de dix fois le nombre de paramètres à déterminer (les poids).

Derrière ces considérations générales se dissimule l'irrégularité fréquente de l'échantillonnage : plus dense dans certaines régions de l'espace des données que dans d'autres. La méthode d'échantillonnage apparaît primordiale.

II.2.9 La base de test

Du fait des capacités d'approximation universelle des modèles neuronaux, l'apprentissage peut mener à un sur-ajustement de la fonction, on parle aussi de sur-apprentissage. On observe ce genre de problème lorsque l'on utilise un modèle comportant un grand nombre de paramètres pour modéliser une fonction de trop faible complexité. Pour mettre en évidence ce problème on utilise une base de test, autre échantillonnage de l'espace des données. Lors de l'étape d'estimation des paramètres, le phénomène de sur-apprentissage

ce traduit par une croissance de l'erreur sur les données de la base de test. Au finale, le réseau sélectionné est alors celui qui minimise l'erreur commise sur la base de test.

II.3 Modèle neuronal modulaire

II.3.1 Préambule

On présente ici une classe de modèle appelé modèle neuronal modulaire (MNM), introduit dans la communauté des réseaux de neurones par Jacobs *et al.* (1991). L'idée de base, inhérente aux modèles modulaires, est de diviser un problème complexe en un certain nombre de sous-problèmes plus simples et plus spécifiques. De plus, puisque l'on ne connaît pas la partition à l'avance, la résolution des sous-problèmes et du partitionnement s'opère de manière simultanée. Dans l'approche que l'on a choisie, on utilise des réseaux du type perceptron multicouches pour émuler à la fois les sous modèles (appelés *experts*) et le modèle de partitionnement (appelé *réseau contrôleur*). Il est important de souligner que la partition peut être non-linéaire, et que les sous-problèmes à résoudre peuvent l'être tout autant.

A chaque sous modèle, on associe enfin un niveau de bruit. Cette caractéristique apparaît être essentielle pour deux raisons (Mangeas, 1997) :

- la segmentation de l'espace s'opère de manière plus efficace si les niveaux de bruit sont contrastés,
- elle permet au modèle trouvé d'être plus résistant au sur-apprentissage.

Le MNM apparaît particulièrement bien adapté aux problèmes pour lesquelles la fonction à estimer est multi-modale.

II.3.2 Principe général

Dans cette section, on détaille le formalisme mathématique du modèle modulaire. De nature aléatoire, celle-ci est essentiellement basée sur la notion de distribution conditionnelle. La part la plus importante de la théorie revient à Jordan *et al.* (1994), ainsi d'ailleurs que l'adaptation de l'algorithme d'optimisation associé (appelé « Expectation Maximisation »).

Dans notre étude, le but est d'estimer une fonction *T* de $\Re^p$ dans $\Re^q$ en utilisant une base d'apprentissage composée d'un ensemble fini d'exemples $D = \{(\vec{x}_k^{obs}, \vec{y}_k^{obs}), k = 1 \cdots N^{obs}\}$. En pratique, les valeurs numériques de l'ensemble d'exemples *D* sont issues de mesures expérimentales ou sont des approximations (simulations) de ces mesures. On fait l'hypothèse que les données $(\vec{x}^{obs}, \vec{y}^{obs})$ sont des réalisations de variables aléatoires avec une fonction de densité de probabilité :

$$p(x, y) = p(y \mid x)\, p(x) \tag{11}$$

où $p(x, y)$ est la densité de probabilité jointe de *x* et *y*.

$p(y \mid x)$ est la densité de probabilité conditionnelle de *y*, conditionnée par *x*.

$p(x)$ est la densité de probabilité de *x*.

Lorsque la relation entre *x* et *y* n'est pas univoque, le meilleur moyen pour décrire le phénomène sous-jacent est d'approximer la distribution de probabilité conditionnelle des variables de sortie, conditionnée par les variables d'entrées. L'idée principale du MNM est d'estimer la probabilité conditionnelle *p(y | x)* en utilisant une combinaison de K fonctions de densité de probabilité gaussienne, telle que :

$$p(y \mid x) = \sum_{k=1}^{K} \alpha_k(x)\, g_k(y / x) \tag{12}$$

où g_k est une fonction de densité gaussienne de moyenne $\mu_k(x)$ et d'écart type $\sigma_k(x)$

$$g_k(y / x) = \frac{1}{(2\pi)^{\frac{q}{2}} \sigma_k^q(x)} \exp\left(-\frac{\left\| y - \mu_k(x) \right\|^2}{2\sigma_k^2(x)}\right) \tag{13}$$

et où les $\alpha_k(x)$, appelés *coefficients de mélange*, sont normalisés : $\sum_{k=1}^{K} \alpha_k(x) = 1$. Voire illustration (Figure II-5) dans le cas de deux experts.

Le problème est alors d'estimer les $K.(q+2)$ paramètres du modèle : $\mu_k(x), \alpha_k(x)$ et $\sigma_k(x)$.

Toute distribution de densité de probabilité peut ainsi être modélisée à n'importe quel degré de précision, à condition que le nombre K de fonctions gaussiennes soit suffisant.

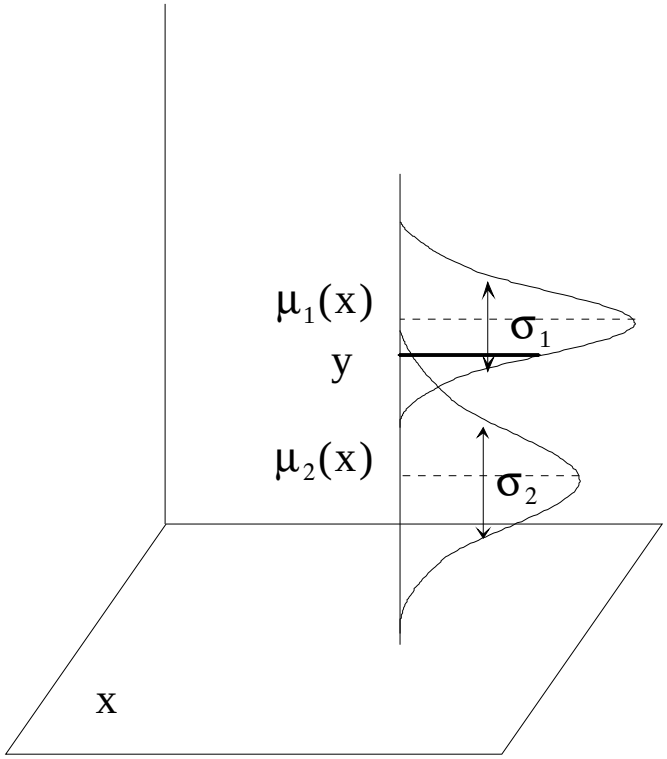

Figure II-5. Densité de probabilité associée à la distribution des mélanges de gaussiennes (équation (12)). Les sorties de chaque expert correspondent aux centres de chaque gaussienne $\mu_k(x)$ *et varient en fonction des entrées. Les variances des gaussiennes (définies par* σ_k*) sont indépendantes des entrées et constantes après convergence du modèle (en phase d'apprentissage). Les centres des deux gaussiennes représentées ici sont pondérés par les sorties du contrôleur* $\alpha_k(x)$*, représentant les probabilités que l'expert et sa gaussienne associée calquent au « vrai » modèle. Dans cette illustration, la séparation des experts n'est pas complète.*

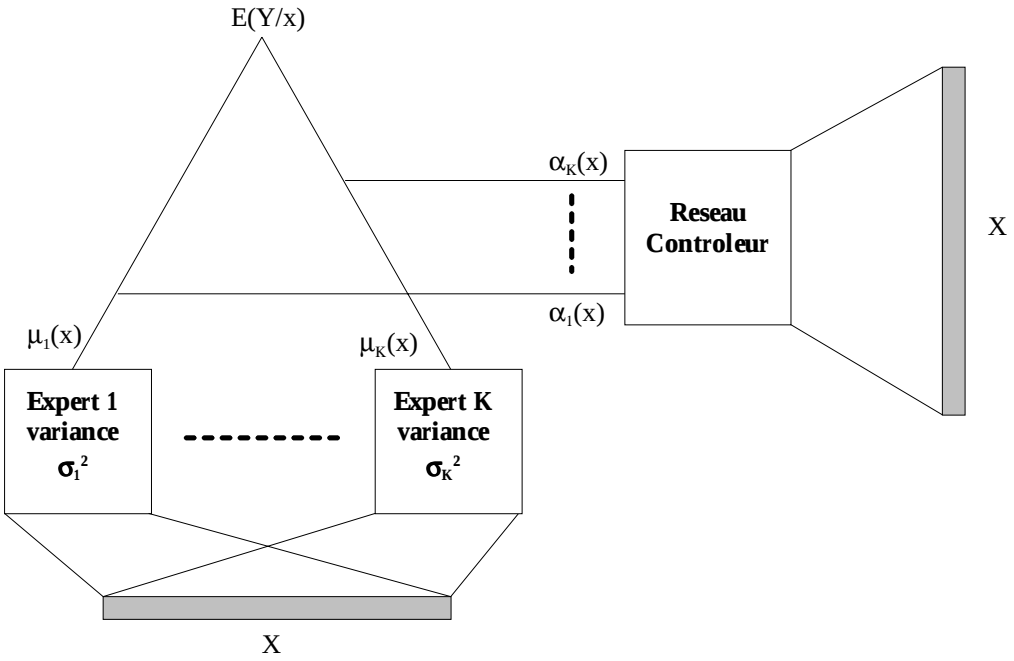

Figure II-6. Architecture d'un réseau Multi-Expert (ME). X représente les variables d'entrée. Chaque Expert est un Perceptron MultiCouche (PMC). σ_k *sont les variances (13). Les sorties* $\alpha_k(x)$ *du réseau contrôleur pondèrent les sorties* $\mu_k(x)$ *de chaque expert. La sortie globale du réseau est alors :*

$$E(Y/x) = \sum_{k=1}^{K} \alpha_k(x)\mu_k(x).$$

II.3.3 Architecture

Dans notre étude, nous avons utilisé une architecture particulière, proposée par Jordan *et al.* (1994) appelée « mixture of experts » et nommé par la suite Multi-Expert (ME). L'architecture de ce réseau est composée de K réseaux experts, et d'un réseau contrôleur « gate » (Figure II-6). Par la suite, par souci de concision, nous n'emploierons plus le terme réseau et nous nommerons ces modules simplement « expert » et « contrôleur ». Experts et contrôleur ont accès aux variables d'entrées. Ces entrées *x* sont le plus souvent communes aux experts et au contrôleur (ce qui est le cas dans notre étude), mais peuvent aussi êtres distribuées de manière non-exhaustive suivant la connaissance à priori de variables. Ainsi, certaines peuvent contenir des informations utiles au partitionnement mais inutiles pour la prévision au sein de chaque partition.

La tâche de chaque expert est de résoudre un problème de régression non-linéaire dans une région de l'espace d'entrée. Cette région est définie au cours de l'apprentissage par le contrôleur, qui engendre des sorties $(\alpha_k(x))_{1<k<K}$ positives, de somme égale à 1, qui pondèrent les sorties des experts $(y_k(x))_{1<k<K}$. Ces pondérations sont modifiées en cours d'apprentissage, en fonction des performances des experts sur les régions ainsi définies. Ceci implique un apprentissage supervisé pour les experts qui « apprennent » à estimer des valeurs de sortie connues, et un apprentissage non-supervisé pour le contrôleur dans la détermination de la segmentation. Cet apprentissage est non-supervisé dans la mesure où la segmentation n'est pas connue à priori et se fonde sur la capacité des experts à se spécialiser dans la prévision d'un certain segment des données.

Précisons le fonctionnement des experts :

Chaque Expert k « apprend » une fonction de $\Re^p$ dans $\Re^q$:

$$x \mapsto y_k(x) = \mathrm{F}_k(\mathrm{x}, \mathbf{W}_k)$$

Un expert est un PMC avec un neurone de sortie de type linéaire, et une ou plusieurs couches cachées composées de neurones de type sigmoïdes. La sortie $y_k(x)$ d'un expert peut être interprétée (en supposant g_k gaussienne), comme la moyenne de la distribution de probabilité : $y_k(x) \approx \mu_k(x)$. L'autre paramètre de la gaussienne, la déviation standard σ_k, est une propriété de l'expert, elle ne dépend pas de la variable d'entrée x. La déviation standard est estimée durant la phase d'apprentissage du réseau, elle s'adapte au niveau de bruit du régime de l'expert.

Précisons le fonctionnement du contrôleur :

Le contrôleur définit une fonction de $\Re^p$ dans $\Re^K$:

$$x \mapsto \alpha(x) = \mathrm{F}_{gate}(\mathrm{x}, \mathbf{W}_{gate})$$

Le contrôleur possède un neurone de sortie pour chaque expert. La k^{ieme} sortie du contrôleur représente l'estimation de la probabilité que le k^{ieme} expert soit activé sachant l'entrée. Pour cela, et puisque ces probabilités sont de somme 1, on utilise en dernier traitement du réseau une fonction « exponentielle normalisée » aussi appelée « softmax ».

Soit $(s_k)_{k=1,2,...,K}$ le vecteur formé par les sorties classiques d'un PMC, avec plusieurs couches cachées munies de fonctions de transfert sigmoïdes et d'une couche de sortie linéaire. Le vecteur $(s_k)_{k=1,2,...,K}$ est alors transformé par la fonction « softmax », afin de fournir des valeurs positives, de somme égale à 1 :

$$\alpha_k(x) = \frac{\exp(s_k)}{\sum_{j=1}^{K} \exp(s_j)} \quad \forall k = 1,2,...K \tag{14}$$

A la fin de la phase d'apprentissage, le ME a estimé la fonction de densité de probabilité conditionnelle *p(y|x)* définie en (12). A partir de cette distribution on peut calculer la valeur moyenne de la densité de probabilité, définie comme la combinaison linéaire des valeurs moyennes de chaque experts *k*, pondérées par les coefficients de mélanges :

$$E(y|x) \equiv \sum_{k=1}^{K} \alpha_k(x)\mu_k(x). \tag{15}$$

Il est possible d'évaluer aussi la variance de la densité de probabilité :

$$s^2(x) = \sum_i \alpha_i(x)\left\{\sigma_i^2 + \left\| \mu_i(x) - \sum_j \alpha_j(x)\mu_{ji}(x) \right\|^2\right\} \tag{16}$$

La valeur moyenne (15) apparaît être un paramètre intéressant dans le cas où la distribution de probabilité (11) est plus ou moins uni-modale. Dans l'hypothèse où chaque exemple (x^{obs}, y^{obs}) de la base d'apprentissage soit généré par un et un seul expert, les coefficients de mélanges seraient devenus binaires au cours de l'apprentissage. Dans ce cas, la sortie globale du ME correspond à la sortie de l'un des experts du réseau et la valeur moyenne a un sens.

Dans les cas où les coefficients de mélanges ne sont pas binaires, lorsque le contrôleur assigne, par exemple, des probabilités égales à 0,5 à deux experts, deux situations sont possibles.

Premièrement, les valeurs de sorties $\mu_k(x)$ des deux experts sont très proches. Dans ce cas, la sortie globale du ME est équivalente à la sortie des deux experts en question et cela ne pose pas de problème.

Deuxièmement, les valeurs de sorties des deux experts sont « éloignées » et la valeur moyenne n'a pas forcément de sens. Dans ce cas, il est préférable de calculer la valeur la

plus probable, qui est donnée par le maximum de la densité de probabilité conditionnelle *p(y | x)*.

Lors de l'application de cette méthode à notre étude, une attention particulière est donnée à la distribution des valeurs des coefficients de mélange.

II.3.4 Fonction de coût

La phase d'apprentissage permet de déterminer les poids $\mathbf{W} = (\mathbf{W_1}, \cdots, \mathbf{W_K}, \mathbf{W_{gate}})$ de chaque experts et du contrôleur ainsi que les K déviations standards $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_K)$, en minimisant une fonction coût définie comme l'opposé du logarithme de la vraisemblance $E(\mathbf{W}, \boldsymbol{\sigma}) = -\ln p(D / \mathbf{W}, \boldsymbol{\sigma})$ qui est donné en utilisant (12) par:

$$E(\mathbf{W},\boldsymbol{\sigma}) = -\sum_{i=1}^{N^{obs}} \ln\left(\sum_{k=1}^{K} \alpha_k(x_i^{obs}) g_k(y_i^{obs} / x_i^{obs})\right) \qquad (17)$$

L'expression (17) est minimisée en utilisant l'algorithme EM (Expectation Maximisation) proposé par Jordan *et al.* (1995).

II.4 Résumé et limitations

Les réseaux de neurones présentés dans ce chapitre (PMC et ME) sont des systèmes paramétrés non linéaires qui relient des variables d'entrée à des variables de sortie. Les paramètres sont déterminés statistiquement lors d'une phase d'apprentissage à partir d'une base de données d'exemples, appelée base d'apprentissage.

En pratique, l'utilisation de méthodes neuronales pose certaines difficultés. La principale difficulté est l'optimisation de la phase d'apprentissage. Le choix de l'architecture adéquate ou la détermination du « pas d'apprentissage » se fait par essais successifs. L'utilisation d'une base indépendante de celle d'apprentissage, appelée base de test permet de déterminer le réseau optimal. On détermine les poids du réseau à partir de la base d'apprentissage et on calcule les performances sur la base de test. Le réseau « optimal » est celui qui minimise l'erreur commise sur la base de test. On utilise ensuite une troisième base indépendante des deux autres, pour donner les performances du réseau et ainsi faire des comparaisons avec d'autres variétés de réseaux ou de méthodes statistiques. La création de trois bases est donc nécessaire pour notre étude.

La deuxième difficulté est liée aux caractéristiques de la base d'apprentissage, aussi bien en terme de taille et de représentativité que de répartition des exemples. Le nombre d'exemples doit être suffisamment grand devant les paramètres (les poids) à déterminer. Le domaine de validité de l'algorithme neuronal est directement lié à la représentativité des exemples de la base d'apprentissage. L'irrégularité des exemples dans certaines régions de l'espace des données peut conduire à une mauvaise optimisation du réseau. Dans ces conditions, un soin particulier doit être appliqué lors de la création des bases de données ; une méthode d'échantillonnage apparaît primordiale.

Les réseaux de type PMC et ME permettent d'obtenir une approximation de la moyenne de la distribution de probabilité conditionnelle des variables de sortie, conditionnée par les variables d'entrée. La sensibilité aux biais statistiques éventuels, présents dans les données, est alors importante. Une attention particulière est donnée afin d'éliminer toutes sources de biais dans les données synthétiques qui serviront à l'apprentissage des réseaux.