RESEAUX DE NEURONES ET MODELES STATISTIQUES

Antonio Ciampi

Department of Epidemiology and Biostatistics

Mc Gill University, Montreal, Quebec, Canada

Yves Lechevallier

INRIA - Rocquencourt Domaine de Voluceau

78153 Le Chesnay Cédex, France

1. Introduction

Les réseaux de neurones formels se sont imposés, dans plusieurs domaines, durant ces dernières années comme un outil universel. En statistique, ils sont utilisés en tant que classificateurs (analyse discriminante), détecteurs de classes (classification automatique), estimateurs non-paramétriques de régression non linéaire et comme estimateurs de fonctions de densité.

Leur utilisation repose sur un théorème d'approximation, beaucoup cité, qui, sous des conditions de régularité relativement modestes, affirme qu'un réseau à trois couches de neurones formels, dont une couche cachée, peut donner une approximation aussi bonne que possible d'une fonction quelconque de plusieurs variables. La qualité de cette approximation augmente en fonction du nombre de neurones formels utilisés.

Comme le montre, entre autres, les travaux de MacKay [McK92] et White [Whi89], les techniques fondamentales de l'inférence statistique, telles que les tests d'hypothèse et l'estimation, s'appliquent aux réseaux de neurones formels. Ceux-ci peuvent être considérés comme des modèles statistiques d'une très grande flexibilité, cette flexibilité étant liée à l'architecture et aux poids des connexions choisis. Toutefois, malgré leur flexibilité et leur universalité, les réseaux de neurones ne peuvent se soustraire aux limites intrinsèques de toute modélisation statistique, et plus particulièrement à celles de l'estimation non-paramétrique, notamment au dilemme " biais-variance " [Gem92]. Ceci constitue une sorte de principe

d'indétermination selon lequel, pour une base de données D_N de taille N fixée on ne peut pas diminuer le biais d'un estimateur sans augmenter sa variance, et vice-versa. Dans le contexte neuronal, on peut toujours augmenter la qualité apparente de l'approximation, c'est-à-dire la qualité jugée sur D_N, en augmentant le nombre de neurones et de connexions, ce qui entraîne la diminution du biais. Cependant, le gain ainsi obtenu est malheureusement compensé par une augmentation de la variance, c'est à dire une diminution du pouvoir de généralisation de cette décision. On observe, en effet, que quand on mesure les performances d'un réseau sur des données ne faisant pas partie de la base d'apprentissage D_N, alors un réseau plus riche en neurones et connexions peut se montrer moins performant qu'un réseau plus simple. Cette performance décroît sensiblement pour tout D_N, à partir d'un certain seuil de complexité qui est fonction de N.

Une manière très générale d'augmenter la qualité réelle de cette approximation est d'augmenter la taille de la base de données. Mais malheureusement la recherche de nouvelles données s'avère souvent, dans la pratique, coûteuse et même impossible. Une autre approche, moins générale et dépendante du domaine et du problème spécifique, est d'introduire de l'information supplémentaire, ce qui revient, dans le contexte neuronal, à dessiner des architectures adaptées au phénomène étudié.

La dernière démarche est analogue, dans le cas de la modélisation classique, à l'introduction de modèles paramétriques ou semi-paramétriques. Elle implique une intervention au niveau de la " boîte noire " qui transforme l'entrée en sortie, afin de la rendre lisible en terme de structures correspondant à des modèles statistiques classiques.

Les travaux de Sethi [Set90] et de Chabanon, Lechevallier et Milleman [Cha92] s'inscrivent dans cette optique. Dans les deux cas, un réseau de neurones est dessiné à partir de l'architecture issue d'un arbre de classification. Cet arbre de classification donne les valeurs initiales des pondérations et celles-ci sont optimisées par l'algorithme de rétropropagation. Dans la même optique, Irino et Kaivahara [Iri90] utilisent comme point de départ de la construction d'un classificateur neuronal, un modèle de régression logistique.

Un point de vue analogue mais dans un contexte plus abstrait, est développé par Geva et Sitte [Gev92], qui indique comment dessiner un réseau utilisant un théorème d'approximation et certaines caractéristiques des données. Ces dernières déterminent des blocs à l'intérieur du réseau, de telle sorte que celui-ci prend la forme d'un réseau de réseaux. L'idée du réseau de réseaux est avancée explicitement chez Cotter [Cot90], où le théorème de Stone-Weierstrass est utilisé afin de réaliser une algèbre de fonctions denses dans l'espace des fonctions que l'on désire approximer.

Nous proposons une architecture qui permet d'associer les réseaux à des modèles statistiques bien connus, tels que les arbres de classification, les modèles logistiques et les modèles additifs généralisés. L'architecture ainsi construite, pourra être utilisé de façon très flexible, soit pour analyser le modèle suggéré directement par les

données en terme de structures facilement interprétables, soit comme simple point de départ d'une procédure itérative.

2. Perceptrons multicouches

2.1. Principes

L'algorithme de rétropropagation du gradient tire son origine du Perceptron présenté par Rosenblatt [Ros57]. Le perceptron est un réseau ayant une seule couche de connexions et donc deux couches de neurones; l'une des couches représente les entrées du système et l'autre les sorties. Rumelhart [Rum86] et Le Cun [LeC85] élargissent l'architecture du perceptron aux réseaux multicouches et règlent le problème de la propagation des erreurs dans les couches cachées. Comme nous nous plaçons dans le cadre de la discrimination ces réseaux appartient à la famille des algorithmes supervisés. Vous pouvez trouver une présentation détaillée des réseaux dans [Cha90], [Mil94], [Gal95], [Mon94] et des discussions entre les méthodes de discrimination et les réseaux dans [Bin94], [Rip94].

Le déroulement de cet algorithme est le suivant : on lui présente séquentiellement des observations, il évolue pour arriver à un certain état S qui est comparé à la réponse désirée Y. Le réseau adapte alors les pondérations W_{ij} des connexions entre les neurones pour réaliser la correspondance souhaitée entre l'état S obtenu et la réponse désirée Y. La base d'apprentissage est présentée plusieurs fois et de manière séquentielle jusqu'à l'obtention d'un minimum acceptable de la fonction de coût.

Les neurones ou cellules d'une même couche ne sont pas connectés entre eux mais sont connectés à la couche ou aux couches suivantes. Le mécanisme d'apprentissage repose sur la minimisation d'une fonction de coût par un algorithme adaptatif de type gradient. Cette fonction de coût évalue l'écart entre la sortie calculée et la sortie désirée sur la dernière couche du réseau. Souvent la fonction de coût choisie est l'erreur moyenne quadratique E :

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{J}\left(Y_j^{(i)} - S_j^{(i)}\right)^2$$

Avec

- N le nombre d'observations de la base d'apprentissage
- J le nombre de neurones de la couche de sortie
- $S_j^{(i)}$ la valeur de sortie du neurone j de la dernière couche obtenue lors de la présentation numéro i.
- $Y_j^{(i)}$ la valeur désirée de la présentation numéro i à la sortie du neurone j.

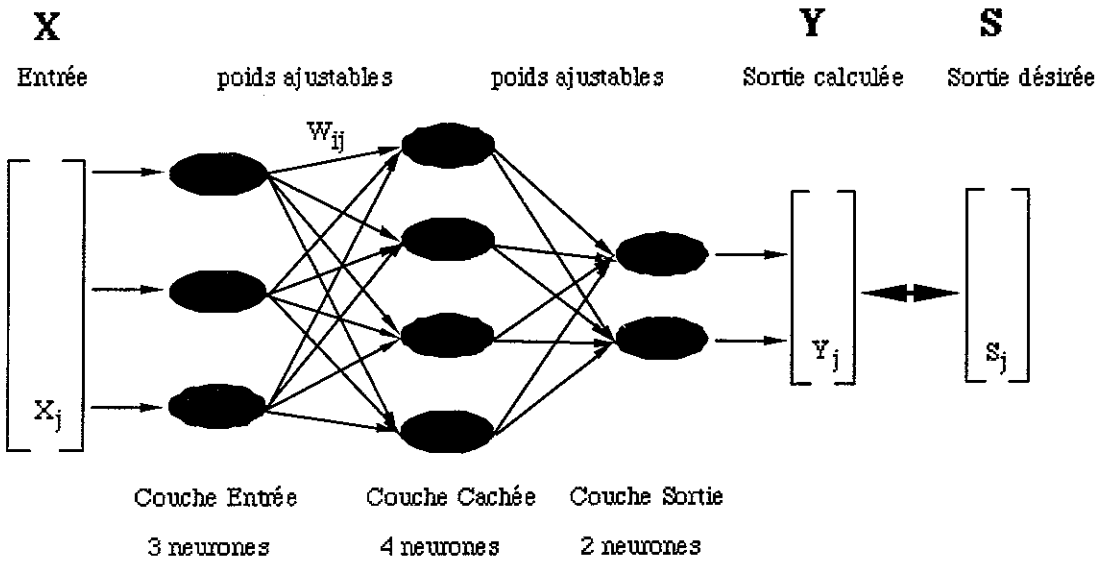


Figure 5. Exemple de réseau

Le réseau de la figure 5 n'utilise que des automates linéaires. Ce réseau possède une seule couche cachée de neurones et deux couches de connexions avec des poids ajustables. Le réseau est dit *réseau à deux couches*. Il est entièrement défini par les liaisons W_{ij} de voisinage, la fonction f de transition (en général une sigmoïde) et la fonction g de changement d'état (fig 6).

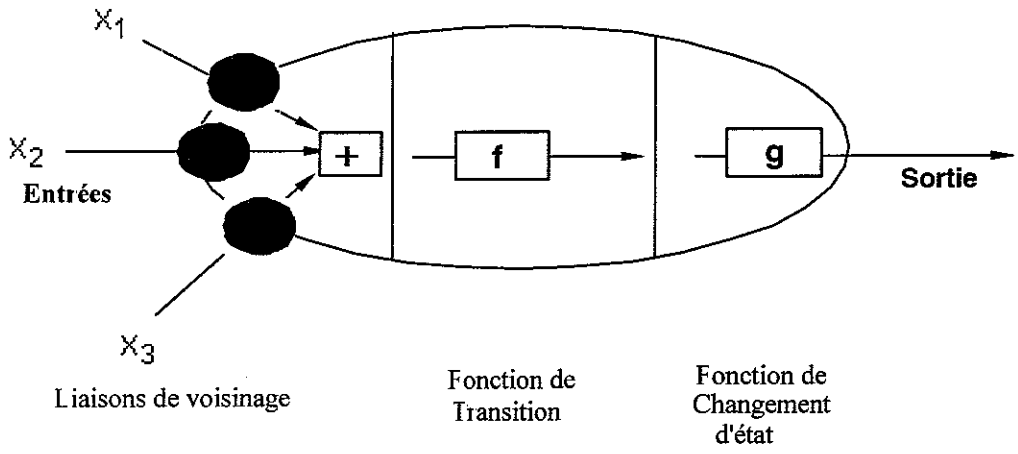


Figure 6. Une cellule ou un neurone

Avec les liaisons de voisinage $X_1, \cdots, X_m$ l'état interne se caractérise par $e_i = \sum_j W_{ij} X_j$. Lorsque e_i est grand (respectivement petit) alors $s_i = f(e_i)$ est grand (respectivement petit) il y a donc activation (respectivement inhibition). La fonction de transition est la fonction sigmoïde:

$$f(x)=\frac{(1-e^{-ax})}{(1+e^{-ax})}=\frac{2}{(1+e^{-ax})}-1$$

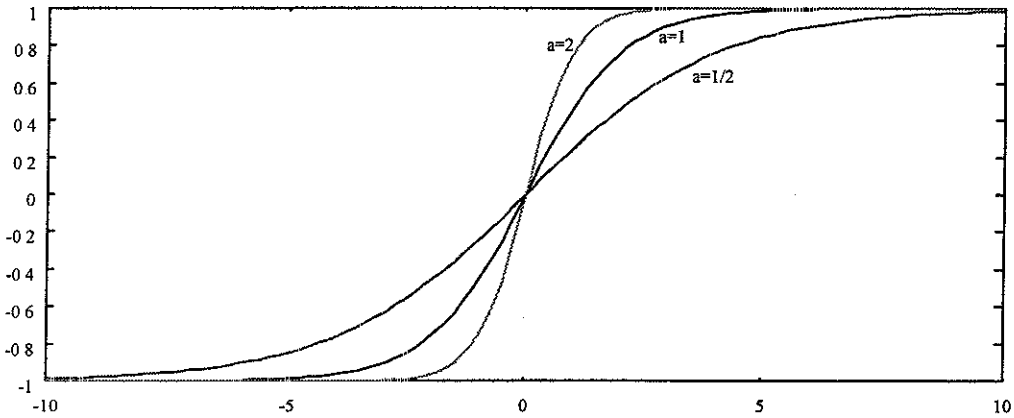


Figure7. La fonction sigmoïde

La fonction de changement d'état est ici la fonction identité. Alors les deux caractéristiques principales d'un neurone ou d'une cellule sont :

- la fonction de transition continue et dérivable
- la rétropropagation de l'erreur de manière récurrente de la couche de sortie vers la couche d'entrée.

Le choix du nombre de couches, du nombres de neurones et de la fonction de transition définit un modèle de prédiction. Le réseau, défini par ces choix, peut approximer toute fonction de prédiction.

Le dilemme « biais-variance » mentionné dans l'introduction joue ici son rôle; le meilleur réseau n'est pas forcément celui qui minimise la fonction de coût sur l'ensemble d'apprentissage mais plutôt celui qui offre une bonne généralisation. La fonction de prédiction à choisir s'obtient donc par un compromis entre l'objectif de garder un coût acceptable sur l'ensemble d'apprentissage et celui d'avoir un nombre de connexions réduit ce qui assure une meilleure généralisation. Un rapport suffisamment grand entre le nombre d'observations ou d'exemples et le nombre de connexions à calculer garantit la qualité de la prédiction. On cherchera donc à éliminer les connexions surabondantes en utilisant les informations extraites de l'arbre de décision construit sur le même échantillon ou ensemble d'apprentissage.

2.2. Algorithme de rétropropagation

A chaque élément ω de la population générale est associé un vecteur de représentation $\mathbf{z}^{(\omega)}$ et les J valeurs $Y_1^{(\omega)},\ldots,Y_J^{(\omega)}$ caractérisant la sortie désirée. On notera par

- $f_i^{(c)}$ la fonction de transition du neurone i de la couche c; cette fonction de transition est continue et dérivable.
- $s_i^{(c)}(\omega)$ la valeur de la sortie du neurone i de la couche c pour l'élément présenté ω.

- $e_i^{(c)}(\omega)$ la valeur de l'entrée du neurone i de la couche c pour l'élément présenté ω.
- $W_{ij}^{(c)}(\omega)$ le poids de la connexion entre la cellule i de la couche (c+1) et la cellule j de la couche c pour l'élément présenté ω.

n_c le nombre de neurones de la couche c et par NC le nombre de couches de ce réseau.

La relation entre les sorties de la couche c et l'entrée du neurone i de la couche (c+1) est une relation linéaire égale à

$$e_i^{(c+1)}(\omega) = \sum_{j=1}^{n_c} W_{ij}^{(c)} \; s_j^{(c)}(\omega)$$

La relation entre la sortie et l'entrée du neurone i de la couche (c+1) est donnée par la fonction de transition $f_i^{(c+1)}$, d'où

$$s_i^{(c+1)}(\omega) = f_i^{(c+1)}\left(e_i^{(c+1)}(\omega)\right) = f_i^{(c+1)}\left(\sum_{j=1}^{n_c} W_{ij}^{(c)} \; s_j^{(c)}(\omega)\right)$$

Si toutes les fonctions de transition sont égales alors le réseau est déterminé uniquement par les poids $W_{ij}^{(c)}$ et par le choix de la fonction de transition f. Ces choix étant fixés, la fonction Φ qui définit le réseau nous permet de calculer les K valeurs de sortie $S_1^{(\omega)}, \ldots, S_K^{(\omega)}$ à partir du vecteur $\mathbf{z}^{(\omega)}$ par:

$$S(\mathbf{z}) = \Phi(W, f)(\mathbf{z})$$

Pour évaluer la performance de ce réseau on se donne une fonction de coût E qui mesure l'adéquation entre la sortie S calculée par ce réseau et la sortie Y désirée par l'utilisateur. Dans la phase rétropropagation la modification des poids du réseau se fait de la façon suivante :

$$W_{ij}^{(c)}(t+1) = W_{ij}^{(c)}(t) + \alpha \frac{\partial E(\omega_t)}{\partial W_{ij}^{(c)}}$$

Il faut calculer pour chaque poids $W_{ij}^{(c)}$ le gradient de E, d'où

$$\frac{\partial E}{\partial W_{ij}^{(c-1)}} = \frac{\partial E}{\partial e_i^{(c)}} \frac{\partial e_i^{(c)}}{\partial W_{ij}^{(c-1)}} = \frac{\partial E}{\partial e_i^{(c)}} \frac{\partial}{\partial W_{ij}^{(c-1)}} \left(\sum_{l=1}^{n_{c-1}} W_{il}^{(c-1)} \; s_l^{(c-1)}\right) = \frac{\partial E}{\partial e_i^{(c)}} s_j^{(c-1)}$$

Pour la cellule i de la couche de sortie il faut calculer $\frac{\partial E}{\partial e_i^{(NC)}}$ qui ne dépend que de la fonction de coût E. Par contre pour les neurones des autres couches nous avons la relation :

$$\frac{\partial E}{\partial e_i^{(c)}} = \sum_{l=1}^{n_{c+1}} \frac{\partial E}{\partial e_l^{(c+1)}} \frac{\partial e_l^{(c+1)}}{\partial e_i^{(c)}}$$

et comme

$$\frac{\partial e_l^{(c+1)}}{\partial e_i^{(c)}} = \frac{\partial}{\partial e_i^{(c)}} \left(\sum_{j=1}^{n_c} W_{lj}^{(c)} \; s_j^{(c)}\right)$$

et que seul $s_i^{(c)}$ dépend de $e_i^{(c)}$, nous avons:

$$\frac{\partial e_l^{(c+1)}}{\partial e_i^{(c)}} = W_{li}^{(c)} \, f_i'^{(c)}$$

d'où

$$\frac{\partial E}{\partial e_i^{(c)}} = \left(\sum_{l=1}^{n_{c+1}} \frac{\partial E}{\partial e_l^{(c+1)}} W_{li}^{(c)} \right) f_i'^{(c)}$$

et

$$\frac{\partial E}{\partial W_{ij}^{(c-1)}} = \left(\sum_{l=1}^{n_{c+1}} \frac{\partial E}{\partial e_l^{(c+1)}} W_{li}^{(c)} \right) f_i'^{(c)} \, s_j^{(c-1)}$$

Ainsi pour les couches cachées les poids de ces couches se calculent en fonction des poids de la couche suivante. Donc la modification de la fonction de coût n'entraîne que la modification du mode de calcul des poids de la dernière couche.

2.3. Choix de la fonction de coût

Souvent la fonction de coût choisie est celle qui minimise l'erreur quadratique. Ainsi les paramètres du réseau sont déterminés de façon à faire décroître la fonction de coût suivante :

$$E = \int \sum_{j=1}^{J} \left(Y_j(\mathbf{z}) - S_j(\mathbf{z}) \right)^2 P(Y_1, \cdots, Y_J, \mathbf{z}) dY_1, \cdots, dY_J d\mathbf{z}$$

Nous prendrons plutôt comme *fonction de coût la quantité de Kullback-Leibler*. Cette fonction mesure la dissimilarité entre deux distributions:

$$E = \int \log \left(\frac{P(Y_1, \cdots, Y_J, \mathbf{z})}{P(Y_1, \cdots, Y_J, \mathbf{z}; S_1, \cdots, S_J)} \right) P(Y_1, \cdots, Y_J, \mathbf{z}) dY_1, \cdots, dY_J d\mathbf{z}$$

3. Représentation neuronale d'un modèle de discrimination

3.1. La régression logistique

Dans le cas de deux classes le modèle de régression logistique est souvent utilisé pour construire une règle de classification. Ceci s'écrit :

$$\text{logit}\,(P(c = c_1 | z)) = \beta \, z \quad (2.1)$$

où c est la variable de classe prenant les valeurs c_1 et c_2 et où il est entendu que le vecteur des covariates contient une composante constante, égale à 1. Il est évident que ce modèle peut se représenter comme un réseau de neurones sans couche cachée (fig1.). Il suffit d'identifier le vecteur β des coefficients de régression avec le vecteur w des poids du réseau. Si l'on choisit comme fonction de coût du réseau le négatif du logarithme de la vraisemblance, l'algorithme de rétro-propagation détermine les poids à partir de l'estimateur de vraisemblance maximale des coefficients de régression.

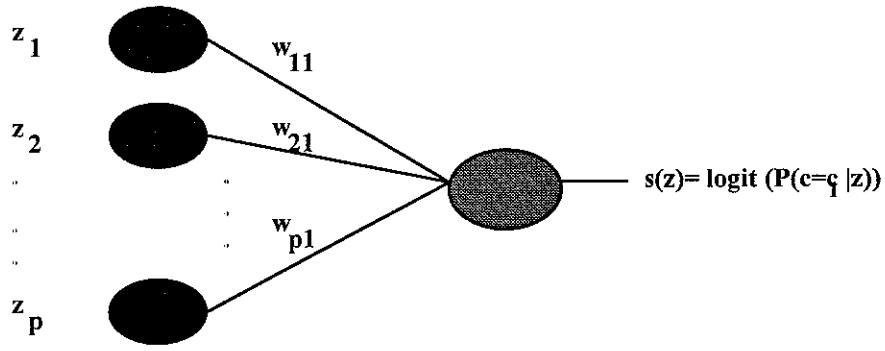


Figure 1. Réseau associé à la régression logistique

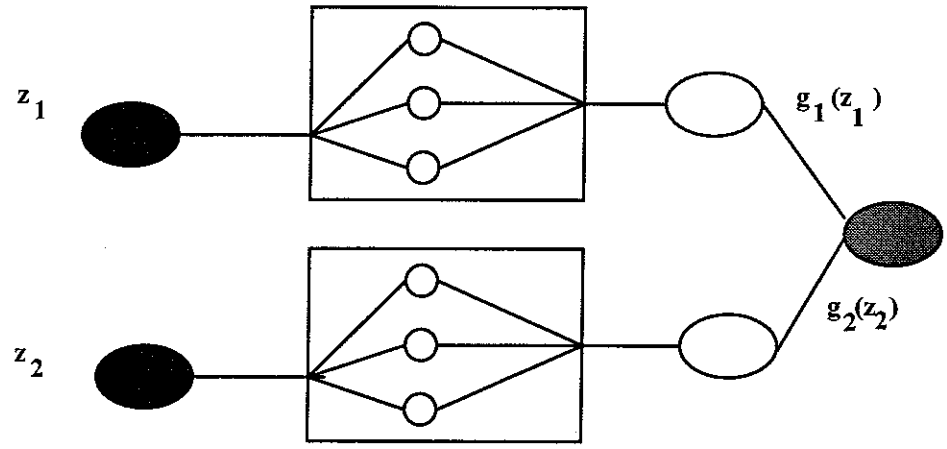
Pour les phénomènes non linéaires, le modèle (2.1) est clairement inadéquat. Plusieurs approches ont été proposées pour dépasser ces contraintes de linéarité; nous en considérerons deux parmi les plus fréquemment employées. L'une de ces approches relâche la contrainte de linéarité logistique en la substituant à celle, plus souple, de l'additivité logistique :

$$\text{logit}\,(P(c=c_1|z)) = g_1(z_1)+g_2(z_2)+\ldots+g_p(z_p) \qquad (2.2)$$

où les g sont des fonctions arbitraires que les données doivent déterminer. Ces dernières, dans la pratique, sont souvent exprimées comme combinaisons linéaires d'une base de dimension finie dans un espace de fonctions, par exemple les B-splines [DeB 1978], de telle sorte que leur estimation se réduit à l'estimation des coefficients d'un modèle linéaire.

Un exemple de réseau de neurones représentant le modèle de l'équation (2.2) est visualisé par la figure 2. Il s'agit d'un réseau à quatre couches de neurones. Cet exemple est décrit par deux variables continues. La première couche cachée consiste de deux blocs non connectés, dont chacun transforme chacune des deux variables en un nombre de fonctions de sortie égale à la dimension de la base choisie.

En conséquence les fonctions de transition de cette couche sont les fonctions de la base, par exemple les B-splines. En effet, au lieu d'introduire cette couche, l'on peut tout simplement transformer préalablement les variables de description. La deuxième couche cachée calcule les coefficients des combinaisons linéaires des fonctions de base; ses fonctions de transition sont linéaires : f(x) = x. Finalement le seul neurone de la couche de sortie (classification binaire) a la fonction logistique comme fonction de transition.



Blocs représentant les B-splines

Figure 2. Réseau pour le modèle additif

3.2. L'arbre de régression généralisé

Bien que le modèle additif logistique de l'équation (2.2) ajoute beaucoup de flexibilité dans le traitement pratique de phénomènes non linéaires, il demeure néanmoins sévèrement limité, car il ne peut pas prendre en compte les interactions possibles entre variables. Le deuxième type de généralisation est l'arbre de régression. Cet arbre permet de dépasser cette limite en imposant que les interactions assument un rôle dominant dans cette analyse. Ce modèle [Cia91] s'écrit selon l'équation (2.3):

$$\text{logit}\,(P(c=c_j|z)) = \gamma_1 I_1(z) + \gamma_2 I_2(z) + \ldots + \gamma_L I_L(z) \qquad (2.3)$$

où les I sont les fonctions indicatrices des L sous-ensembles de l'espace de représentation formant une partition. Cette partition est obtenue par algorithme récursif à partir d'un jeu de données et peut être représentée par une structure d'arbre (fig3).

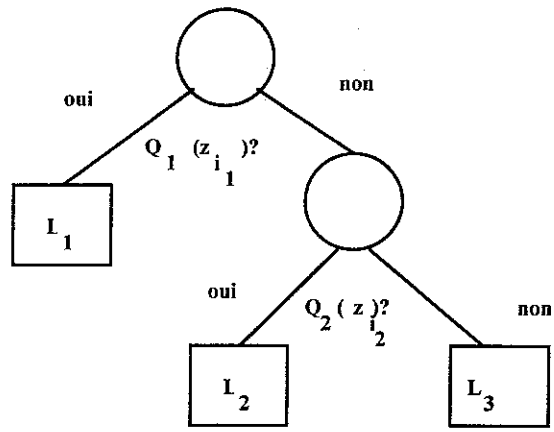


Figure 3. Arbre de régression généralisé

Les feuilles de l'arbre représentent les ensembles de la partition. Le chemin conduisant à une feuille détermine le contour de la région associée à cette feuille dans l'espace de description. Un chemin est défini par une série de questions binaires, chacune est construite à partir d'une seule variable. La question sélectionnée à chaque nœud de l'arbre est celle qui est la plus informative en fonction des probabilités des classes a priori et du tableau de données représentant l'ensemble d'apprentissage. La partition est définie par l'intersection d'hyperplans parallèles aux axes de l'espace de description.
Dans des travaux précédents [Set90], [Bre91], [Cha92] il a été remarqué qu'un arbre de décision peut être représenté par un réseau. Notons qu'une autre manière d'utiliser la structure d'arbre dans un réseau a été proposée sous le nom *de réseau hybride* dans [Dal93]. La Figure 4 montre la représentation neuronale de l'arbre de décision de la figure 3.

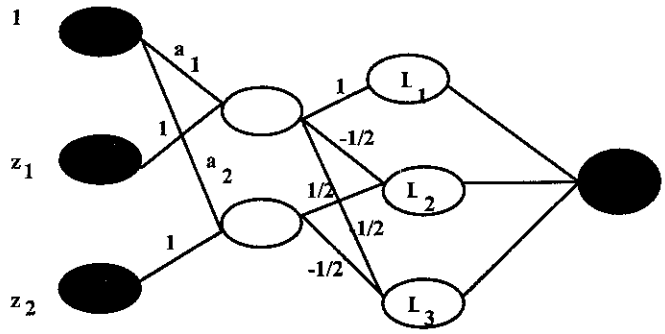


Figure 4. Réseau défini à partir de l'arbre de la figure 3

Il s'agit d'un réseau ayant deux couches de neurones cachés. A chaque cellule de la première couche cachée est associée un nœud non terminal de l'arbre. La seconde représente les feuilles de l'arbre. Dans la figure 4, les poids des connexions internes sont marqués. Ce réseau posséde trois couches de connexions. Chaque connexion de la première couche relie la cellule contenant un nœud de l'arbre à la variable utilisée par la question associée ce nœud, les poids des connexions de cette couche sont tous égaux à un. A chaque cellule contenant une feuille de l'arbre est associé un nombre de connexions égal au nombre de nœuds reliant la racine à cette feuille. Chaque connexion représente un élément du chemin associé à cette feuille. Les poids sont positifs si la connexion représente une réponse positive et négatifs si la réponse est négative. La dernière couche de connexions est complète. Les poids sont déterminés par les données et valent le logit de la probabilité de la classe a priori sur la feuille correspondante.
Toutefois, l'intérêt du réseau de la figure 4 est dans la possibilité de permettre des seuils 'flous' dans les couches cachées, c'est-à-dire des fonctions sigmoïdes comprises entre -1 et 1. Ce faisant, il est utile de centrer les données autour des points de coupures et de les réduire à l'échelle de la variance empirique. Dans ce cas, les poids reliant le biais (neurone unité) de la couche d'entrée à ceux de la première couche cachée, valent tous zéro.

Les poids marqués sur la figure 4 sont les initialisations de l'algorithme de rétropropagation, lequel détermine les poids de toutes ces connexions. Le résultat est encore facile à interpréter, puisque il représente le même arbre, mais avec des seuils flous aux nœuds.

3.3. Mise en œuvre de l'algorithme de rétropropagation

Comme nous l'avons vu au paragraphe 2.2, la fonction Φ d'un réseau de neurones permet de calculer les valeurs de sortie $S_1^{(\omega)},\ldots,S_J^{(\omega)}$ du réseau en fonction des valeurs d'entrée qui sont, ici, les valeurs du vecteur de description $\mathbf{z}$. Pour que les valeurs de sortie soient interprétables en terme de probabilité d'appartenance aux K classes a priori il faut que ces valeurs vérifient les conditions suivantes :

$$(a) \quad \forall k = 1,\cdots,K \quad o \le p_k(\mathbf{z}) \le 1$$

$$(b) \quad \sum_{k=1}^{K} p_k(\mathbf{z}) = 1$$

Pour intégrer ces contraintes dans la définition des sorties du réseau nous proposons ici le codage suivant :

$$\forall k = 1,\cdots,K-1 \quad S_k = \log\left(\frac{p_k}{p_K}\right)$$

Dans ce cas le nombre J de neurones de sortie est égal à K-1. Aussi les valeurs $Y_1,\ldots,Y_J$ vont être simplement $Y_j = c_j \quad \forall j = 1,\cdots,K-1$. Par exemple si l'élément i est affecté à la classe a priori k nous avons:

$$Y_k^{(i)} = 1 \quad et \quad Y_l^{(i)} = 0 \quad pour\ l \neq k$$

La fonction de coût que nous utiliserons sur l'ensemble d'apprentissage est le logarithme de la vraisemblance L.

$$E = \log L = \sum_{i=1}^{N} \log\left(P(\mathbf{c}^{(i)},\mathbf{z}^{(i)})\right)$$

$$= \sum_{i=1}^{N} \left(\log(P(\mathbf{z}^{(i)}) + \log(P(\mathbf{c}^{(i)}/\mathbf{z}^{(i)})\right)$$

Le premier terme de cette équation est indépendante du réseau donc il suffit de rendre maximum

$$\sum_{i=1}^{N} \log(P(\mathbf{c}^{(i)}/\mathbf{z}^{(i)}))$$

et ceci s'écrit en fonction des Y_k :

$$\sum_{i=1}^{N}\sum_{k=1}^{K} Y_k^{(i)} \log\left(p_k(\mathbf{z}^{(i)})\right)$$

où $p_k(\mathbf{z}^{(i)})$ est la probabilité d'appartenir à la classe k sachant $\mathbf{z}^{(i)}$.

Si le réseau doit estimer ces probabilités conditionnelles, alors la maximisation de la vraisemblance de cet échantillon correspond à minimiser une fonction de coût calculée à partir des sorties $S_1(\mathbf{z}),\ldots,S_{K-1}(\mathbf{z})$ du réseau de la manière suivante :

$$E=-\sum_{i=1}^{N}\sum_{k=1}^{K}\left(Y_k^{(i)}\,log\left(p_k(\mathbf{z}^{(i)})\right)+\left(1-\sum_{l=1}^{K-1}Y_l^{(i)}\right)log\left(1-\sum_{l=1}^{K-1}p_k(\mathbf{z}^{(i)})\right)\right)$$

Alors les probabilités $p_1,\ldots,p_{K-1}$ se calculent ainsi :

$$p_k(\mathbf{z})=\frac{e^{S_k(\mathbf{z})}}{1+\sum_{l=1}^{K-1}e^{S_l(\mathbf{z})}}\quad\forall k=1,\ldots,K-1$$

et p_K par

$$p_K(\mathbf{z})=1-\sum_{k=1}^{K-1}p_k(\mathbf{z})=\frac{1}{1+\sum_{l=1}^{K-1}e^{S_l(\mathbf{z})}}$$

Comme nous l'avons précisé au paragraphe 2.2 les gradients des poids des connexions non liées à la couche de sortie ne dépendent pas de la fonction de coût. Il suffit de calculer la valeur $\frac{\partial E}{\partial e_j^{(NC)}}$ pour les neurones j de la dernière couche de 1 à $K-1$. Ce calcul s'effectue de trois manières différentes:

1.**Pour** $k\neq j$ nous avons

$$\frac{\partial E_i}{\partial e_j^{(NC)}}=-\frac{Y_k^{(i)}}{p_k(\mathbf{z}^{(i)})}\frac{\partial p_k(\mathbf{z}^{(i)})}{\partial e_j^{(NC)}}$$

$$\frac{\partial p_k(\mathbf{z}^{(i)})}{\partial e_j^{(NC)}}=-\frac{e^{S_k}}{\left(1+\sum_{l=1}^{K-1}e^{S_l}\right)^2}e^{S_j}\frac{\partial S_j}{\partial e_j^{(NC)}}=-p_k(\mathbf{z}^{(i)})\,p_j(\mathbf{z}^{(i)})\frac{\partial S_j}{\partial e_j^{(NC)}}$$

et comme $\frac{\partial S_j}{\partial e_j^{(NC)}}=2\,S_j\left(1-S_j\right)$ alors nous avons

$$\frac{\partial E_i}{\partial e_j^{(NC)}}=2\,p_j(\mathbf{z}^{(i)})\,S_j(\mathbf{z}^{(i)})\left(1-S_j(\mathbf{z}^{(i)})\right)$$

2.**Pour** $k=j$ nous avons

$$\frac{\partial p_k(\mathbf{z}^{(i)})}{\partial e_k^{(NC)}}=-\left(\frac{e^{S_k}}{\left(1+\sum_{l=1}^{K-1}e^{S_l}\right)}-\frac{e^{S_k}.e^{S_k}}{\left(1+\sum_{l=1}^{K-1}e^{S_l}\right)^2}\right)\frac{\partial S_k}{\partial e_k^{(NC)}}=-p_k(\mathbf{z}^{(i)})\left(1-p_k(\mathbf{z}^{(i)})\right)\frac{\partial S_k}{\partial e_k^{(NC)}}$$

d'où

$$\frac{\partial E_i}{\partial e_k^{(NC)}} = -2\left(1 - p_k(\mathbf{z}^{(i)})\right) S_k(\mathbf{z}^{(i)}) \left(1 - S_k(\mathbf{z}^{(i)})\right)$$

3 **Pour** $k = K$ et pour toutes les neurones j de la dernière couche de 1 à K nous avons

$$\frac{\partial p_K(\mathbf{z}^{(i)})}{\partial e_j^{(NC)}} = \frac{e^{S_j}}{\left(1 + \sum_{l=1}^{K-1} e^{S_l}\right)^2} \frac{\partial S_j}{\partial e_j^{(NC)}} = p_K(\mathbf{z}^{(i)}) p_j(\mathbf{z}^{(i)}) \frac{\partial S_j}{\partial e_j^{(NC)}}$$

$$= 2\, p_K(\mathbf{z}^{(i)}) p_j(\mathbf{z}^{(i)}) S_j(\mathbf{z}^{(i)}) \left(1 - S_j(\mathbf{z}^{(i)})\right)$$

4.Un exemple en discrimination

Les échantillons sont issus d'un problème de reconnaissance des formes ou de discrimination introduit par L. Breiman, J. H. Friedman, R. A. Olshen et C. J. Stone [Bre84]. Ce problème est très fréquemment abordé dans la littérature [PRC94] et consiste à reconnaître trois formes d'ondes. Chaque forme d'onde est une courbe modélisée par une série chronologique composée de 21 instants régulièrement espacés. Chaque exemple ou observation d'une de ces formes d'ondes est représenté par un vecteur ayant 21 valeurs.

Les trois formes d'ondes sont définies à partir de combinaisons deux à deux de trois modèles de base h_1, h_2 et h_3. La figure 8 présente ces modèles de bases.

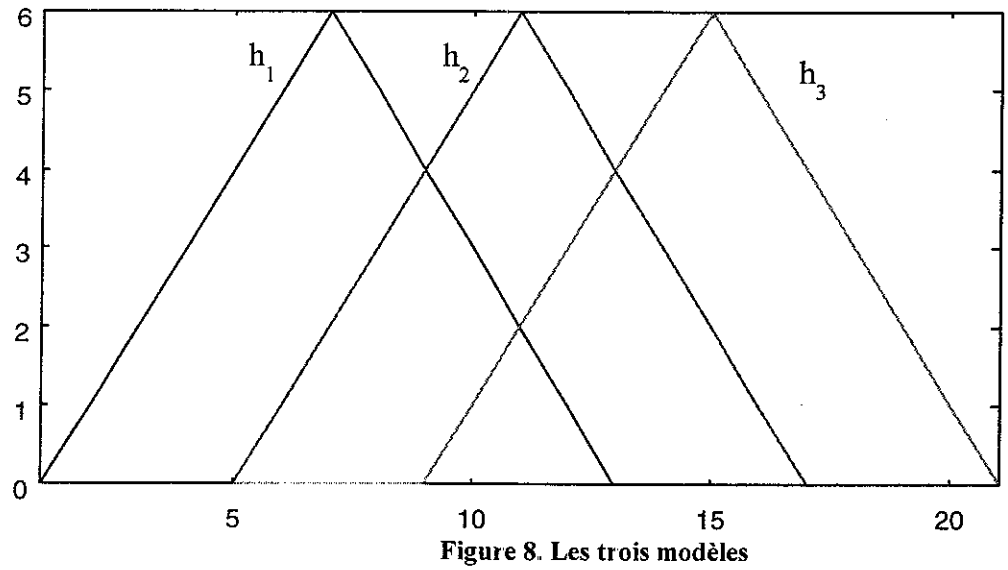


Figure 8. Les trois modèles

Un exemple ou observation, représenté par le vecteur $\mathbf{x} = (x_1, x_2, \cdots, x_{21})$ de la première classe, est généré par

$$x_j = u\, h_1(j) + (1-u)\, h_2(j) + \varepsilon_j, \text{ pour } j = 1, \ldots, 21$$

Un exemple de la seconde classe est généré par :

$$x_j = u\, h_1(j) + (1-u)\, h_3(j) + \varepsilon_j, \text{ pour } j = 1, \ldots, 21$$

Un exemple de la dernière classe est généré par :

$$x_j = u\, h_2(j) + (1-u)\, h_3(j) + \varepsilon_j, \text{ pour } j = 1, \ldots, 21$$

où u est une loi uniforme sur l'intervalle unité et les ε_j sont des variables aléatoires indépendantes de loi normale centrée et réduite. Si l'on suppose que les probabilités des classes a priori sont égales, alors, selon Breiman et al. [Bre84] le taux d'erreur de classement est égal à 0.14.

Le rapport final d'activité [PRC94] du projet Inter-PRCs « Méthodes Symbolique-Numériques de Discrimination » présente les résultats obtenus sur cet ensemble de données par un certain nombre de méthodes symboliques ou numériques de discrimination.

Notre ensemble d'apprentissage est constitué de 300 exemples tirés de façon indépendantes et en supposant les classes a priori équiprobables.
En utilisant l'analyse factorielle discriminante nous avons obtenu le premier plan factoriel suivant :

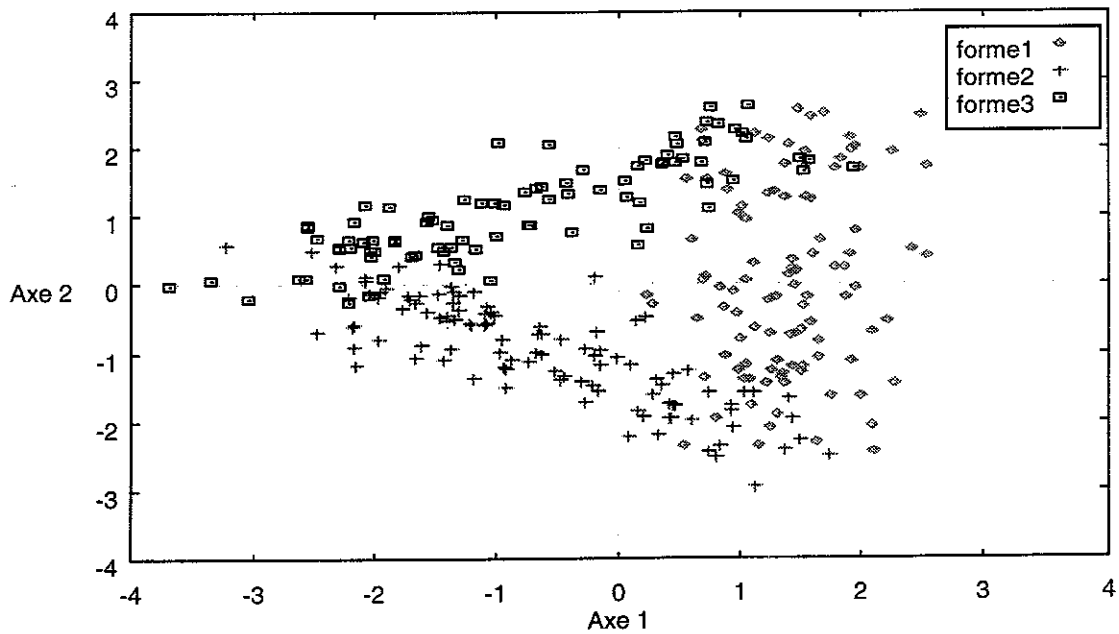


Figure 9. L'analyse factorielle discriminante

Pour cet ensemble des 300 exemples nous avons construit l'arbre de décision suivant:

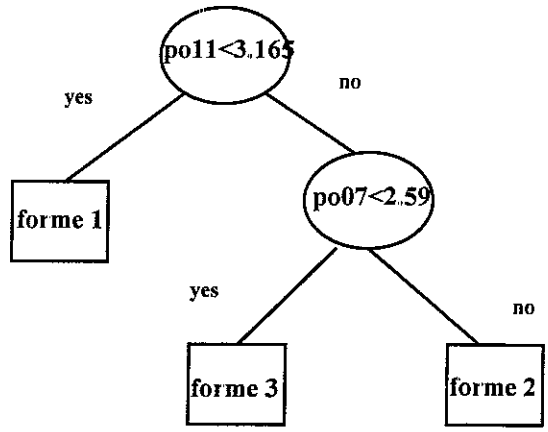


Figure 10. L'arbre de décision

Avec cet arbre nous obtenons le tableau de confusion (fig 11) sur l'ensemble d'apprentissage:

	forme1	forme2	forme3	Total
Région1	93	24	22	139
Région2	5	63	5	73
Région3	5	21	62	88
Total	103	108	89	300

Figure 11. Tableau de confusion avec l'arbre

Comme l'arbre de décision n'utilise que deux variables nous pouvons représenter dans un plan les régions de décision obtenues par cet arbre.

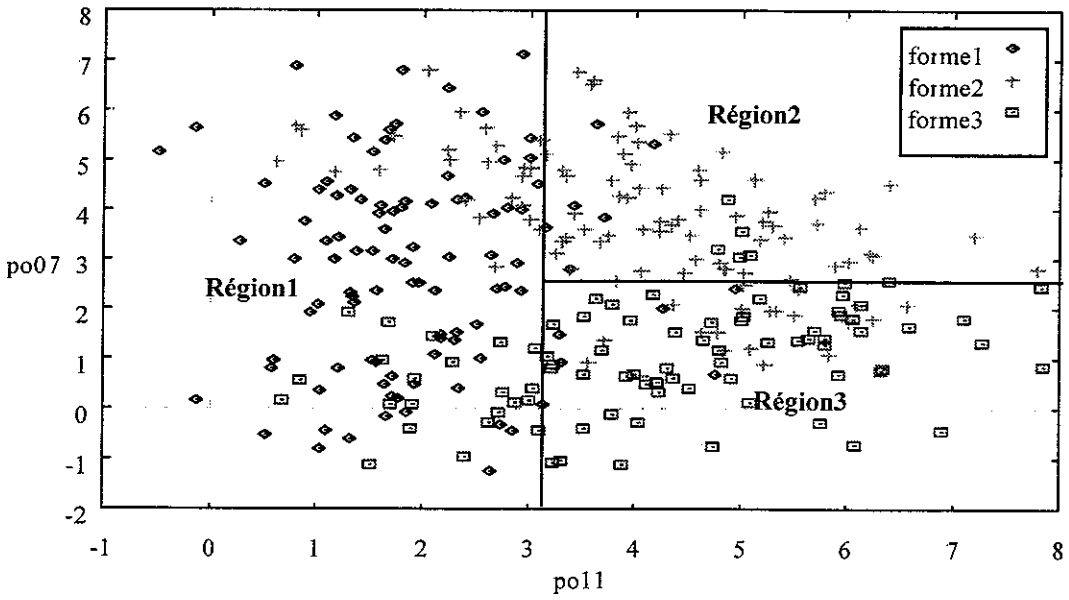


Figure 12. Les régions d'affectation obtenues par l'arbre

A partir de cet arbre de décision nous pouvons construire le réseau suivant:

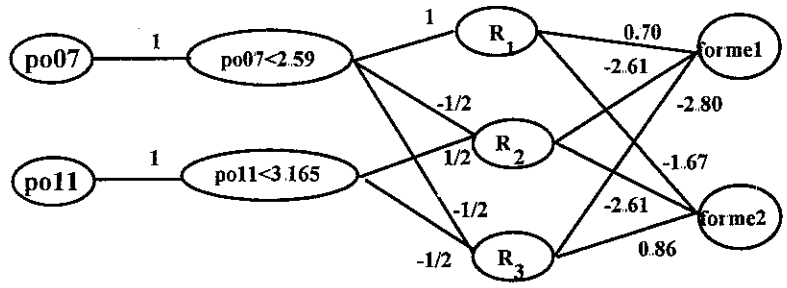


Figure 13. Le réseau construit à partir de l'arbre

En optimisant ce réseau par l'algorithme de rétropropagation, et au bout de 1000 présentations de l'ensemble d'apprentissage, nous obtenons une classification. La figure 14 indiquent les classes d'affectation des exemples déterminées par le réseau. Sur la figure 14 nous pouvons observer la déformation des frontières réalisée par le réseau.

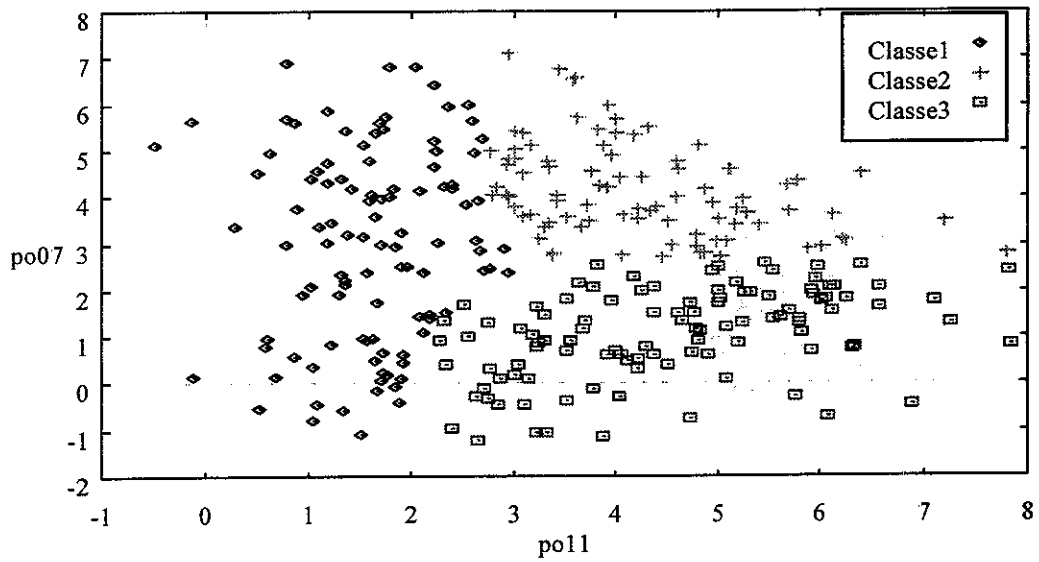


Figure 14. Les régions d'affectation obtenues parle réseau

Et ce tableau de confusion sur l'ensemble d'apprentissage est égal à:

	forme1	forme2	forme3	Total
Classe1	72	7	13	92
Classe2	22	78	5	105
Classe3	9	23	71	103
Total	103	108	89	300

Figure 15. Tableau de confusion associé au réseau

Le tableau de comparaison de la figure 16 a été construit sur l'ensemble d'apprentissage constitué de 300 exemples. La première colonne de ce tableau représente les effectifs des trois formes a priori. La seconde colonne et troisième colonne représentent le nombre d'exemples bien classés dans chacune de ces formes et en fonction de la méthode utilisée. Les deux dernières colonnes représentent le taux d'exemples mal classés par la méthode d'arbre de décision seule et avec le réseau associé.

	Effectif	Effectif Arbre	Effectif Réseau	% Arbre	% Réseau
forme1	103	93	72	9.7	30.1
forme2	108	63	78	58.3	27.8
forme3	89	62	71	30.3	20.2
Total	300	218	221	27.3	26.3

Figure 16. Tableau de comparaison

Dans cet exemple nous pouvons remarquer que le taux de mauvaise classification diminue un peu mais surtout il y a un rééquilibrage des taux de mauvaise classification entre les formes a priori. Avec le réseau ces taux sont très proches.

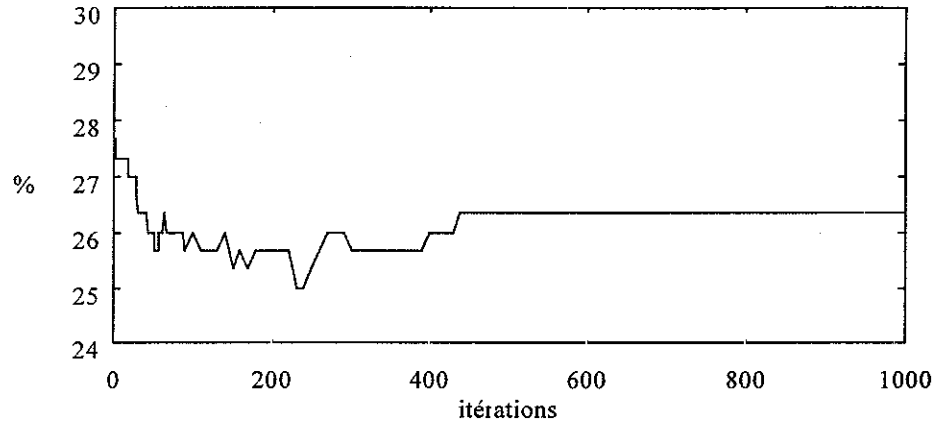


Figure 17. Evolution du taux apparent de mauvaise classification

La figure 17 montre que le taux de mauvaise classification sur l'ensemble d'apprentissage décroît jusqu'à l'itération 250 et puis croît et se stabilise à partir de l'itération 425.

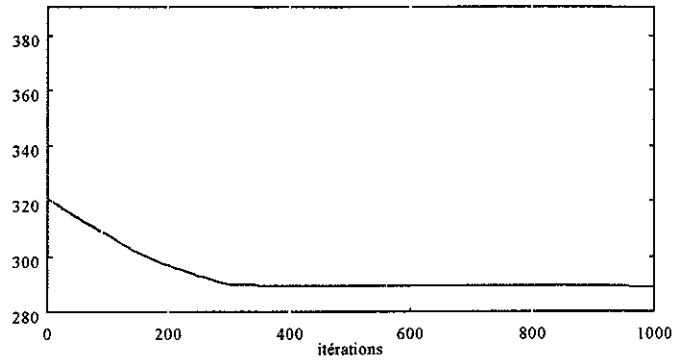


Figure 18. Evolution de la vraisemblance

Nous pouvons remarquer que les dix premières itérations font décroître très rapidement le critère optimisé par l'algorithme de rétroprogation, puis nous avons une décroissance jusqu'à l'itération 250 et ensuite cette décroissance devient très faible.

A partir d'un échantillon-test de 5000 exemples nous avons les résultats suivants:

	Effectif	Effectif Arbre	Effectif Réseau	% Arbre	% Réseau
forme1	1678	1510	1152	10.01	31.34
forme2	1643	830	1091	49.48	33.59
forme3	1679	1080	1254	35.67	25.31
Total	5000	3420	3497	31.16	30.06

Nous retrouvons sur l'échantillon-test le même phénomène que sur l'ensemble d'apprentissage. Nous pouvons remarquer que le fait d'utiliser un réseau après la méthode de classification par arbre fait décroître sur l'ensemble test le taux de mauvaise classification de 1,1 % et que ce résultat est très semblable au résultat de 1% obtenu sur l'ensemble d'apprentissage. Ceci prouve que cette stratégie ne diminue pas la capacité de généralisation.

5.Conclusion

Nous avons montré que les réseaux de neurones et la modélisation statistiques peuvent interagir de façon fructueuse. La recherche d'une flexibilité dans la modélisation et d'une prédiction optimale ne s'oppose pas à la démarche d'interprétation, ni au pouvoir de généralisation, préoccupations fondamentales chez les statisticiens. La clef est la correspondance entre modèle statistique et architecture neuronale. Cette correspondance peut être utilisée, soit pour accélérer la convergence de l'algorithme de rétropropagation, soit pour construire de nouvelles architectures.

6.Bibliographie

[Ara92] **Araya R. and Gibon P.** (1992) " Segmentation Trees/ A New help for Building Expert Systems and Neural Networks ", COMPSTAT92.

[Bar93] **Barndorff-Nielsen O.E., Cox D.R., Jensen L., Kendall W.S.** (eds.) (1993). *Chaos and Networks - Statistical Probabilistic Aspects*, Chapman & Hall, London:.

[Bin94] **Bing Cheng and Titterington D. M.** (1994) " Neural networks: a review from a statistical perspective " ,*Statistical Science*, 9, 2-54.

[Bre84] **Breiman L., Friedman J.H., Olshen R.A. and Stone C.J.** (1984) *Classification and Regression Trees*, Wadsworth.

[Bre91] **Brent R.P.** (1991) " Fast training algorithms for multilayer neural nets ", *IEEE Trans. on Neural Networks*, 2, 346-354.

[Cha90] **Chabanon C. et Dubuisson B.** (1990) " Méthodes non probabilistes ", Dans: *Analyse discriminante sur variables continues,* Collection Didactique INRIA.

[Cha92] **Chabanon C., Lechevallier Y. et Millemann S.** (1992). " An efficient neural network by a classification tree ". In: *Computational Statistics*. Proccedings of the 10 Symposium on Computational Statistics, Vol.1, 227-232, Physica-Verlag.

[Cia91] **Ciampi A.** (1991) " Generalized Regression Trees ", *Computational Statist. and Data Analysis*, 12, 57-78.

[Cot90] **Cotter N. E.** (1990) " The Stone-Weierstrass Theorem and Its Application to Neural Networks ", *IEEE Trans. on Neural Networks*, 1, 290-295.

[Dal93] **D'Alché-Buc F.** (1993)*Modèles neuronaux et algorithmes constructifs pour l'apprentissage de règles de décision.* Thèse Paris XI.

[DeB78] **De Boor C.D.** (1978) *A practical guide to splines*. Springer, New York.

[Dub90] **Dubuisson B.** (1990) *Diagnostic et Reconnaissance des formes,* Hermés.

[Gal95] **Gallinari P. et Gascuel O.** (1995) " Statistiques, apprentissage et généralisation: applications aux réseaux de neurones ", RIA, à paraître.

[Gem92] **Geman S., Bienenstock E. and Dorsat R.** (1992) " Neural Networks and the Bias/Variance dilemma ". *Neural Computation*, 4,1-58.

[Gev92] **Geva S. and Sitte J.** (1992) " A Constructive Method for Multivariate Function Approximation by Multilayer Perceptrons ", *IEEE Trans. on Neural Networks*, 4,621-624.

[Has90] **Hastie T. and Tibshirani R.** (1990) *Generalized Additive Models*, Chapman & Hall, London.

[Hec90] **Hecht-Nielsen R.** (1990), *Neurocomputing*, Addison-Wesley, Reading, Mass.

[Hos90] **Hosmer D.W. and Lemenshow S.** (1990) *Applied Logistic Regression,* J. Wiley, NewYork.

[Iri90] **Irino T. and Kawahara** (1990) " A Method for Desining Neural Network Using Nonlinear Multivariate Analysis: Application to Speaker-Independent Vowel Recognition". *Neural Computation*, 2,386-397.

[LeC85] **Le Cun Y.** (1985) " A learning scheme for asymetric threshold network" *Cognitiva*, 85,599-604.

[McK92] **MacKay D. J. C.** (1992) " Bayesian Interpolation ". *Neural Computation*, 4,415-447.

[Mil93] **Milgram M.** (1993) *Reconnaissance des formes, méthodes numériques et connexionnistes,* Armand Colin.

[Mon94] **Monrocq C.** (1994) *Approche probabiliste pour l'élaboration et la validation de systèmes de décision.* Thèse Paris IX.

[Pol94] **Poli I. and Jones R. D.** (1994) " A neural net model for prediction ", *J. Amer. Stat. Asso.*, 89, 117-121.

[PRC94] **Gascuel O. et Gallinari P.** (1994) *Méthodes Symbolique-Numériques de Discrimination*, Rapport final d'activité. Disponible sur demande.

[Rip94] **Ripley B. D.** (1994) " Neural networks and related methods for classification " (with discussion) *J. Royal. Stat Soc.* B, 56, 409-456.

[Ros57] **Rosenblatt** (1957) " The perceptron a perceiving and recognizing automation", *Tech report, Cornell Aeronautical Laboratory Report No 85*-460-1.

[Rum86] **Rummelhart D., Hinton G. E. and Williams R. J.** (1986) " Learning internal representations by error retropropagation, Paralled distributed processing/ exploration in the micro-structure of cognition ", *MIT Press.*

[Set90] **Sethi I. K.** (1990) " Entropy nets: from decision trees to neural networks " *Proc. IEEE*, 78, 1605-1613.

[Whi89] **White H.** (1989) " Some asymptotic results for learning in single hidden layer feedforward networks ", *J. Amer. Stat. Asso.*, 84, 1008-1013.

[Whi92] **White H.** (1992). *Artificial neural networks. Approximation and learning theory,* Basil Blackwell, Oxford.