Les réseaux de neurones: principes et application à la détection financière des faillites

M. Schyns*

Février 1997

1 Introduction

Dans l'état actuel de nos connaissances dans le domaine de l'intelligence artificielle et plus généralement de l'informatique, certains problèmes restent complexes à résoudre. Citons par exemple:

- permettre à un robot de conduire un véhicule dans un environnement variable;
- réaliser la lecture automatique d'un texte manuscrit produit par n'importe quelle personne;
- reconnaître la parole quelle que soit la personne qui parle;
- réaliser une prévision financière.

Les ordinateurs sont extrêmement rapides et précis pour exécuter des séquences d'instructions qui ont été formulées pour eux. Malheureusement, les problèmes cités nécessitent la considération simultanée d'un très grand nombre de données, parfois mal définies. C'est pourquoi il est difficile de leur trouver une formulation informatique. Par ailleurs, on sait que le système humain de traitement de l'information est composé de neurones qui travaillent à des vitesses à peu près un million de fois plus lentes que les circuits d'un ordinateur et que malgré cela les humains sont beaucoup plus efficaces pour résoudre des problèmes complexes comme ceux mentionnés plus haut. L'organisation du cerveau humain semble donc une des clefs du problème. Ceci explique pourquoi de nombreux chercheurs ont tenté depuis quelques années de reproduire artificiellement une partie de ses structures élémentaires. Différentes approches ont déjà été

*Université de Liège, Ecole d'Administration des Affaires, Email: M.Schyns@ulg.ac.be

une partie de ses structures élémentaires. Différentes approches ont déjà été présentées dans la littérature. Toutefois, comme la compréhension des systèmes neuronaux biologiques n'est pas encore très avancée, nous devons nous limiter à des modèles très simplifiés, mais on peut supposer que cette technique ne fera que se développer avec l'amélioration de nos connaissances sur le cerveau.

Cette capacité de traitement théorique et les premiers résultats obtenus dans la pratique méritent que l'on s'intéresse aux réseaux de neurones. Nous décrirons brièvement un modèle généralement utilisé: les réseaux multicouches. Nous en déduirons quelques autres avantages des réseaux de neurones par rapport à des modélisations plus traditionnelles, mais aussi certains inconvénients.

2 Historique

La première modélisation d'un neurone date de 1943. Elle a été présentée par McCulloch et Pitts. L'interconnexion de ces neurones permet le calcul de plusieurs fonctions logiques. En 1949, Hebb propose le premier mécanisme d'évolution des connexions, appelées (par analogie aux systèmes biologiques) des synapses. L'association de ces deux méthodes permit à Rosenblatt en 1958 de décrire le premier modèle opérationnel de réseaux de neurones: le perceptron. Celui-ci est capable d'apprendre à calculer un grand nombre de fonctions booléennes, mais pas toutes. Ses limites théoriques furent mises en évidence par Minsky et Papert en 1969. Depuis 1985, de nouveaux modèles mathématiques ont permis de les dépasser et ont donné naissance aux réseaux multicouches que nous étudierons plus particulièrement.

3 Un neurone artificiel

3.1 Le modèle de McCulloch et Pitts

La première définition formelle d'un neurone artificiel basée sur le modèle biologique a été formulée par McCulloch et Pitts. Il s'agit d'une unité élémentaire de calcul (le neurone) reliée via des connexions (les synapses) à une ou plusieurs entrées (les stimuli) et fournissant une seule sortie (la réponse à l'environnement). Un poids est associé à chaque synapse pour établir l'importance de l'entrée correspondante. La sortie de chaque neurone est le résultat d'une fonction mathématique prenant en argument la somme des entrées pondérées.

Dans le modèle de McCulloch et Pitts, les entrées $X_i (i = 1, 2, \ldots, n)$ sont booléennes (présence ou absence d'impulsion à l'instant k). La sortie est identifiée par le symbole O. W_i est le poids associé aux connexions. Une représentation en est donnée à la Figure 1a. La fonction calculée par ce neurone est définie comme suit:

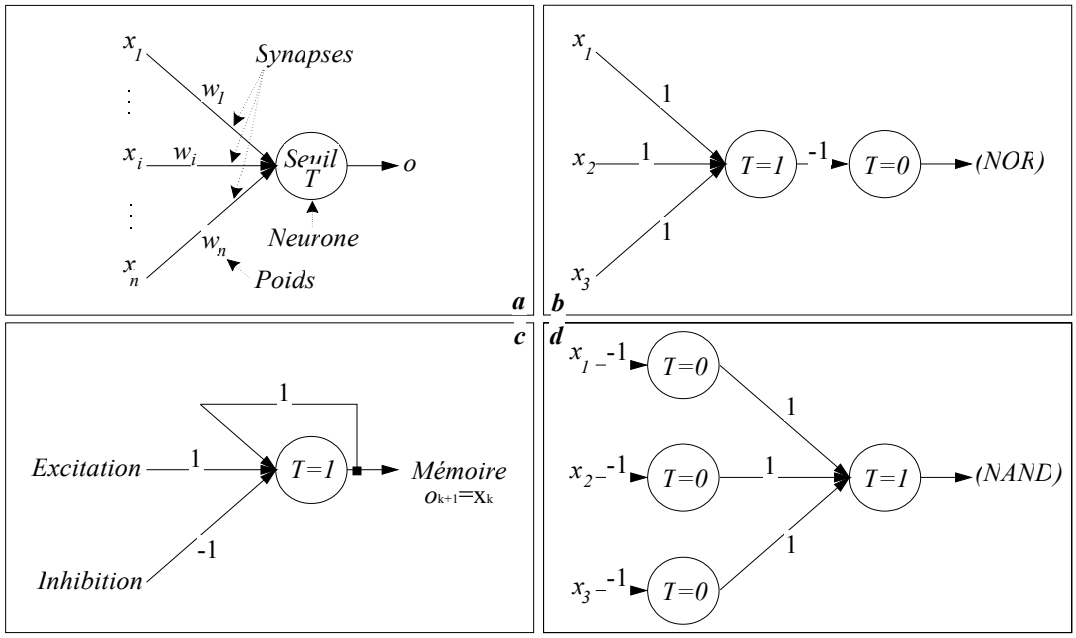

Figure 1: Des portes élémentaires

$$O^{k+1} = \begin{cases} 1 & \text{si} \sum_{i=1}^{n} W_i X_i^k \geq T \\ 0 & \text{si} \sum_{i=1}^{n} W_i X_i^k < T \end{cases}$$

où T est un seuil d'acceptation.

Notons qu'un poids négatif inhibe une connexion, au contraire d'un poids positif qui la renforce. C'est le choix des valeurs de ces poids qui permettra de réaliser la fonction recherchée. La Figure 1 montre comment on peut construire des portes électroniques élémentaires (les opérateurs "négation du ET logique", "négation du OU logique" et une cellule de mémorisation) grâce à ces "coefficients synaptiques". On peut en déduire que ce modèle simpliste permet déjà la réalisation d'un ordinateur digital de complexité arbitraire.

Dans la figure b, le premier neurone représente la fonction "OU" logique de 3 variables. Pour que le résultat soit 1, il faut qu'au moins une des 3 entrées soit à 1. Dans ce cas la somme à l'entrée du neurone sera supérieure ou égale à 1. Avec un seuil unitaire, on discrimine alors les deux solutions. Le deuxième neurone réalise la négation. En multipliant la sortie du neurone par -1, on inverse le signe. Une valeur négative (inférieure au seuil 0) donnera une sortie nulle au contraire d'une valeur nulle qui donnera une sortie unitaire. L'explication de la figure d est identique. Il suffit de se rappeler qu'en logique la négation du "ET" de 3 variables est le "OU" des négations de ces variables. La cellule de mémorisation

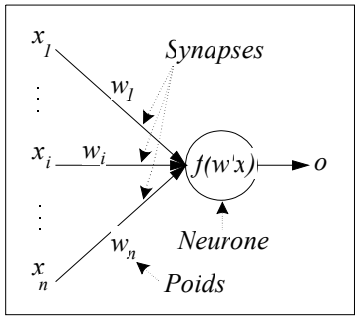

Figure 2: Forme générale d'un neurone artificiel

est plus compliquée. Une excitation (sans inhibition au même moment) active la sortie car le seuil est atteint. Une inhibition (sans activation) réinitialise la sortie à 0. Si aucun signal n'est fourni aux entrées, la sortie au temps k est le seul excitateur du système et elle est alors reproduite au temps $k+1$.

3.2 Généralisation

Le précédent modèle a plusieurs inconvénients. Il est binaire et statique. Les coefficients et les seuils T sont fixés définitivement. Il est assez simple de résoudre ces problèmes. La Figure 2 représente la nouvelle forme du neurone.

Chaque neurone consiste en un élément de traitement (processeur) de plusieurs entrées et calculant une seule sortie. Les poids synaptiques sont représentés par un vecteur W dont les éléments sont modifiables. La sortie est calculée par une fonction d'activation f. Différentes fonctions sont envisageables. Les plus courantes sont la fonction binaire signe et celles continues de type sigmoïde. Notons que le seuil du modèle de McCulloch et Pitts sera représenté en ajoutant une connexion dont le poids est -1 et l'entrée le seuil T.

Le problème, pour un énoncé donné, consiste à trouver les coefficients synaptiques idéaux.

3.3 Exemple

On désire obtenir un avis semblable à celui d'un expert sur la santé financière d'un ensemble d'entreprises. Pour cela, on peut utiliser le modèle développé par Altman en 1968, mis à jour en 1983 et basé sur l'analyse discriminante. On pourrait le résumer comme suit:

$$y = 0.717X_1 + 0.847X_2 + 3.107X_3 + 0.42X_4 + 0.998X_5 - 2.065 \qquad (1)$$

où:

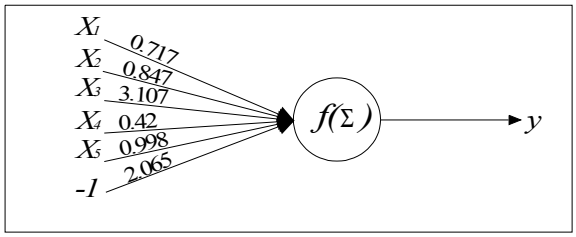

Figure 3: Neurone linéaire estimant la situation financière d'une entreprise

X_1 = Fonds de Roulement Net / Actif Total
X_2 = Réserves / Actif Total
X_3 = EBIT / Actif Total
X_4 = Fonds Propres / Dettes Totales
X_5 = Chiffre d'Affaires / Actif Total

et où y représente l'estimation de la santé financière de l'entreprise. Si y est positif, la société est jugée saine.

Cette équation correspond exactement à un neurone linéaire à 6 entrées. Il est représenté à la Figure 3; f est dans ce cas la fonction identité appliquée à la somme des entrées multipliées par un poids synaptique.

4 Les réseaux

Une fois la structure d'un neurone établie, la définition d'un réseau est immédiate: un réseau de neurones est une interconnexion de neurones telle que leur sortie est connectée, avec un poids synaptique, aux entrées d'autres neurones. Le choix des neurones connectés entre eux détermine l'architecture du réseau. Différents modèles existent: multicouches, Hopfield, Kohonen, Boltzmann. Chacun a ses propres techniques d'apprentissage et s'applique plus particulièrement à certains domaines. Disposant d'une bonne méthode d'apprentissage pour les réseaux multicouches et ceux-ci répondant à nos besoins, nous laisserons le loisir au lecteur intéressé de consulter les ouvrages de références pour les autres modèles. Ajoutons que ce type de réseau est celui le plus utilisé, en particulier dans le domaine de la finance. Mais revenons-en à la notion d'apprentissage.

5 Apprentissage et adaptation

L'apprentissage consiste à trouver le réseau qui fournit la meilleure solution au problème qu'il doit résoudre, pour un ensemble de données dites d'apprentissage. Cela est réalisé en adaptant, grâce à certaines règles, les vecteurs de poids. Ce concept d'apprentissage n'est vraiment intéressant que si le réseau possède des capacités de généralisation, c'est-à-dire s'il est capable de fournir une bonne solution pour d'autres données.

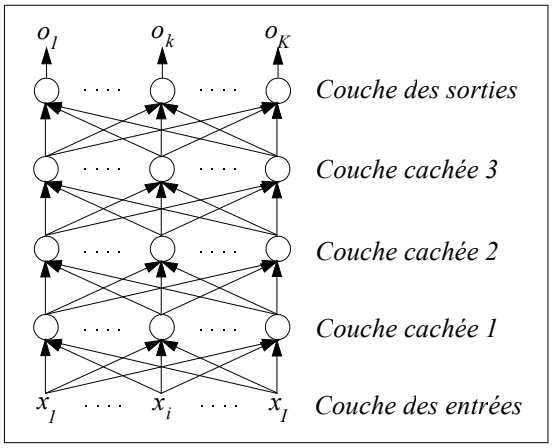

Figure 4: Réseau multicouches

De nombreuses règles d'adaptation des poids existent: loi de Hebb (renforcer une connexion entre deux neurones actifs), règle du Perceptron (modifier les poids en fonction de l'erreur entre la sortie souhaitée et celle obtenue), loi Delta, de Widrow-Hoff, de corrélation, ou encore du type "le gagnant prend tout". Plus généralement, on peut répartir ces méthodes d'adaptation en deux classes:

1. Les apprentissages supervisés ou avec professeur. Dans cette catégorie, on suppose que chaque fois qu'une entrée est appliquée au réseau, la sortie désirée est fournie par le professeur. Celui-ci pourra alors récompenser ou punir le réseau selon l'erreur commise, en ajustant les poids.

2. Les apprentissages non supervisés. Dans cette catégorie, la réponse désirée est inconnue. Le réseau est alors supposé découvrir lui-même la meilleure représentation de l'information fournie.

Il est important de remarquer que c'est l'ensemble des poids ainsi obtenus qui détermine le réseau résolvant le problème posé. La dégradation accidentelle d'un de ces poids n'aura en général que peu d'influence sur le résultat final; les autres assurant la convergence vers la solution.

6 Les réseaux multicouches

6.1 Description

Il est ici question d'un réseau non récurrent de neurones. Sa structure est définie par les caractéristiques suivantes:

1. L'ensemble des neurones du réseau peut être divisé en $N \geq 3$ sous-ensembles disjoints. Chacun de ces ensembles s'appelle une couche.

2. Il existe une numérotation de 1 à N de ces couches telle que:

   - deux neurones appartenant à des couches différentes ne peuvent être directement connectés que si ces deux couches sont adjacentes; c'est-à-dire si leur indice diffère d'une unité.
   - la couche 1 s'appelle la couche d'entrée et est la seule dont les neurones ont leur état directement influencé par l'environnement. En réalité, ces neurones sont fictifs et se contentent de jouer le rôle d'interface entre le réseau et l'environnement. Elle n'est d'ailleurs pas toujours numérotée. Cela provoque parois une ambiguïté dans les termes.
   - la couche N s'appelle la couche de sortie et est la seule qui fournisse directement une réponse à l'environnement.
   - les autres couches sont appelées couches cachées et ne peuvent communiquer directement avec les données du problème: l'environnement.
   - les réseaux non récurrents ne possèdent pas de cycle; c'est-à-dire que les connexions se font toujours d'un neurone vers un autre de la couche immédiatement supérieure.

Un exemple d'un tel réseau est représenté à la Figure 4. Ici, l'information progresse de couche en couche en partant de celle d'entrée pour arriver à celle de sortie où elle représente alors la réponse fournie par le système à l'environnement.

6.2 Objectifs.

La structure par couches qui vient d'être présentée est très générale et peut être utilisée pour résoudre un grand nombre de types de problèmes. En général, on considère deux grandes familles: la classification et l'approximation de fonctions.

En ajoutant systématiquement une entrée supplémentaire à chaque neurone pour représenter un seuil d'acceptation (comme cela a déjà été fait dans l'exemple de la Figure 3), on constate que chaque neurone partitionne l'espace en deux zones, deux classes (signe de la somme pondérée des entrées). En interconnectant des neurones entre eux selon le schéma qui vient d'être énoncé, on peut ainsi réaliser une partition de l'espace des données en un nombre arbitraire de classes. Il peut être montré que pour n'importe quel problème de classification, il existe un réseau non récurrent à trois couches qui résout ce problème. Cependant, il n'est pas sans intérêt d'utiliser un plus grand nombre de couches intermédiaires. En effet, cet accroissement du nombre de couches s'accompagne généralement d'une amélioration des capacités de généralisation et d'une meilleure efficacité quant à la représentation interne construite par le réseau pour stocker les données lors de l'apprentissage. Par contre, un réseau de seulement deux couches ne pourra réaliser ques des partitions linéaires de l'espace des entrées.

La capacité des réseaux de neurones à approximer des fonctions découle des propriétés de classification. La fonction à représenter peut-être échantillonnée en sous-intervalles. Un réseau multicouches, dont les neurones possèdent une entrée de seuillage, pourra coder ces zones et approximer correctement la fonction. Les valeurs des seuils seront liées à la largeur et au mileu de chaque fenêtre d'échantillonage. Il y a cependant un problème supplémentaire. Typiquement, comme nous le verrons plus loin, la fonction associée aux neurones a un intervalle de variation compris entre 0 et 1 ou -1 et 1. Si le domaine de variation de la fonction à approximer n'est pas le même, deux solutions sont envisagées: soit on effectue une mise à l'échelle, soit (par exemple si la fonction n'est pas bornée) on utilise la fonction identité pour le neurone de sortie.

6.3 Apprentissage par rétropropagation d'erreurs.

L'algorithme de la rétropropagation d'erreurs est actuellement la procédure d'apprentissage la plus utilisée. Cette popularité est une conséquence des résultats généralement bons qu'elle permet d'obtenir pour un grand nombre de problèmes différents et cela en réalisant des simulations sur machines séquentielles, en des temps raisonnables vis-à-vis de ceux requis par d'autres algorithmes tel que, par exemple, celui du recuit simulé.

Dans cet algorithme, de même que l'on est capable de propager un signal provenant des neurones d'entrée, on peut, en suivant le chemin inverse, rétropropager l'erreur mesurée en sortie vers les couches internes et modifier en conséquence les poids pour diminuer l'erreur. La plus connue des fonctions de correction est la loi Delta généralisée. Cette minimisation de l'erreur permet de mémoriser la relation définie entre l'entrée et la sortie désirée. En pratique, on construit un ensemble de données, dit d'apprentissage, composé de paires entrée-sortie désirée. Chacune des paires sera présentée au réseau et les poids corrigés en fonction dans une phase de rétropropagation. Ce schéma est répété plusieurs fois jusqu'à l'obtention d'une erreur globale acceptable. Le nombre d'itérations nécessaires n'est malheureusement pas estimable.

Voici une présentation de l'algorithme de rétropropagation des erreurs et de la loi Delta.

Partons d'abord du réseau présenté à la Figure 5. Toutes les entrées, tous les neurones et toutes les sorties sont numérotés séquentiellement.

Les entrées et les sorties des neurones sont dès lors respectivement représentées par les vecteurs suivants:

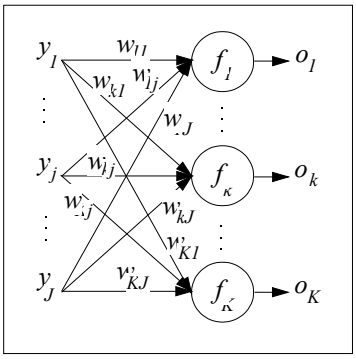

Figure 5: Réseau élémentaire

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_j \\ \vdots \\ y_J \end{pmatrix} \text{ et } o = \begin{pmatrix} o_1 \\ \vdots \\ o_k \\ \vdots \\ o_K \end{pmatrix} \tag{2}$$

En désignant par w_{kj} le poids de la connexion partant de l'entrée y_j et aboutissant au k^{eme} neurone, on peut définir une matrice de poids:

$$W = \begin{pmatrix} w_{11} & \dots & w_{1j} & \dots & w_{1J} \\ \vdots & & \vdots & & \vdots \\ w_{k1} & \dots & w_{kj} & \dots & w_{kJ} \\ \vdots & & \vdots & & \vdots \\ w_{K1} & \dots & w_{Kj} & \dots & w_{KJ} \end{pmatrix} \tag{3}$$

Notons aussi d le vecteur des sorties désirées:

$$d = \begin{pmatrix} d_1 \\ \vdots \\ d_k \\ \vdots \\ d_K \end{pmatrix} \tag{4}$$

L'entrée totale du neurone k est définie par:

$$net_k = \sum_{j=1}^{J} w_{kj} y_j \tag{5}$$

La sortie correspondante est alors donnée par:

$$o_k = f_k(net_k) \tag{6}$$

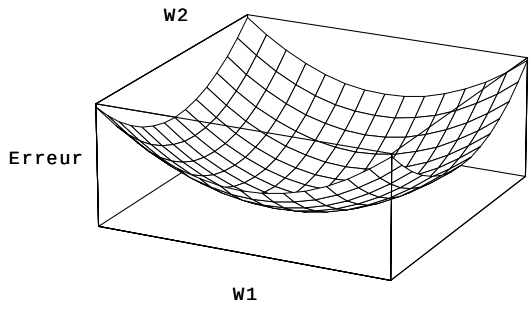

Figure 6: Erreur du réseau en fonction de deux poids.

On cherche à minimiser une fonction d'erreur mesurant la différence entre les sorties obtenues o et les sorties attendues d; c'est-à-dire:

$$E = \frac{1}{2}\sum_{k=1}^{K}(d_k - o_k)^2 \tag{7}$$

Un exemple est illustré à la Figure 6 pour un réseau comprenant deux poids.

Pour minimiser cette fonction, on peut agir sur les poids W. Etant donné la surface d'erreur définie par (7), la variation optimale des poids est proportionnelle à la direction opposée à celle du gradient:

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} \tag{8}$$

où η est une constante positive d'apprentissage.

Développons maintenant cette expression (8):

$$\begin{aligned} \frac{\partial E}{\partial w_{kj}} &= \frac{\partial E}{\partial (net_k)} \frac{\partial (net_k)}{\partial w_{kj}} \\ &= \frac{\partial E}{\partial (net_k)} y_j \end{aligned} \tag{9}$$

En posant,

$$\delta_{o_k} = -\frac{\partial E}{\partial (net_k)} \tag{10}$$

la variation d'un poids (8) peut se réécrire:

$$\Delta w_{kj} = \eta \delta_{ok} y_j \tag{11}$$

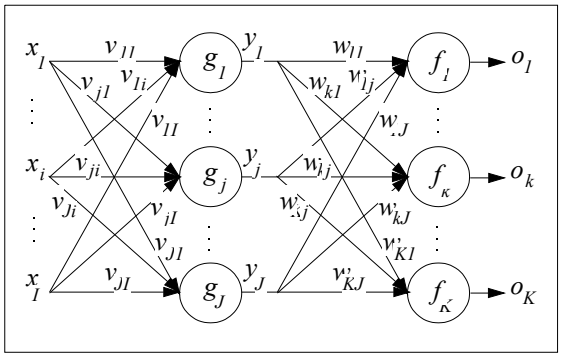

Figure 7: Réseau multicouches

Cette formule d'apprentissage est connue sous le nom de loi Delta et le terme δ est appelé signal d'erreur.

Dans le cas de notre réseau de départ, le terme δ_{o_k} vaut:

$$\begin{aligned} \delta_{o_k} &= -\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial (net_k)} \\ &= (d_k - o_k) f_k'(net_k) \end{aligned} \tag{12}$$

Le signal d'erreur δ_{o_k} décrit l'erreur locale $(d_k - o_k)$ à la sortie du neurone k, multipliée par un facteur d'échelle $f_k'(net_k)$, qui est la pente de la fonction d'activation calculée en la valeur net_k.

On obtient le résultat final:

$$\Delta w_{kj} = \eta (d_k - o_k) f_k'(net_k) y_j \tag{13}$$

$$w_{kj} \rightarrow w_{kj} + \Delta w_{kj} \tag{14}$$

Nous sommes intéressés par des réseaux multicouches et, en particulier, par un réseau à trois couches. Ce dernier est suffisant pour permettre la classification de vecteurs d'entrées non linéairement séparables. Agrandissons le réseau précédent comme indiqué sur la Figure 7.

En suivant le même développement que pour le réseau de la Figure 5, nous devons minimiser l'erreur E en appliquant la loi Delta. Pour la troisième couche, appelée couche de sortie, les résultats sont ceux obtenus au paragraphe précédent; c'est-à-dire les équations (13) et (14). Pour la deuxième couche, en modifiant les notations, l'équation (10) décrivant le signal d'erreur devient:

$$\delta_{y_j} = -\frac{\partial E}{\partial (net_j)} \tag{15}$$

et la variation optimale d'un poids (11) peut se réécrire:

$$\Delta v_{ji} = \eta \delta_{y_j} x_i \tag{16}$$

En remarquant que la fonction d'erreur (7) peut aussi se noter:

$$E = \frac{1}{2}\sum_{k=1}^{K}(d_k - f_k(net_k(y)))^2 \tag{17}$$

on obtient pour la deuxième couche le développement suivant:

$$\begin{aligned} \delta_{y_j} &= -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial(net_j)} \\ &= -\frac{\partial E}{\partial y_j} g'_j(net_j) \end{aligned} \tag{18}$$

où g_j est la fonction d'activation associée au neurone j de la deuxième couche.

$$\begin{aligned} \frac{\partial E}{\partial y_j} &= -\sum_{k=1}^{K}(d_k - o_k) f'_k(net_k)\frac{\partial(net_k)}{\partial y_j} \\ &= -\sum_{k=1}^{K}\delta_{o_k} w_{kj} \end{aligned} \tag{19}$$

et donc:

$$\delta_{y_j} = g'_j(net_j)\sum_{k=1}^{K}\delta_{o_k} w_{kj} \tag{20}$$

$$\Delta v_{ji} = \eta x_i g'_j(net_j)\sum_{k=1}^{K}\delta_{o_k} w_{kj} \tag{21}$$

Cette formule est appelée loi Delta généralisée. Il est important de remarquer que la variation optimale des poids d'une couche cachée est obtenue en utilisant le signal d'erreur de la couche immédiatement supérieure. L'erreur est d'abord calculée pour chaque neurone de la couche de sortie et ensuite rétropropagée vers les couches inférieures. C'est l'origine du nom de la méthode. On peut augmenter facilement le nombre de couches cachées en utilisant pour chacune une formule d'adaptation similaire.

Il est intéressant de comparer la structure des équations (13) et (21). Ainsi on peut, en généralisant les notations, résumer l'algorithme par:

$$\Delta w_{kj} = \eta \delta_k y_j \tag{22}$$

où

$$\delta_k = (d_k - o_k) f'_k(net_k) \text{ pour une couche de sortie} \tag{23}$$

$$\delta_k = \sum_{n=1}^{N}\delta_n w_{nk} f'_k(net_k) \text{ pour une couche cachée} \tag{24}$$

Cet algorithme a l'avantage d'être local, c'est-à-dire que la plupart des calculs d'apprentissage peuvent être effectués indépendamment au niveau de chaque neurone (avec un minimum de contrôle global).

Une règle à respecter pour faciliter la convergence est d'utiliser un ensemble d'apprentissage contenant au moins dix fois plus de vecteurs de données, souvent appelés patterns, que de synapses dans le réseau.

6.4 Conception

Nous avons défini jusqu'à présent un type d'architecture et une méthode d'apprentissage généraux. Il en existe de nombreuses variations et améliorations, mais nous nous limiterons ici au cas présenté.

Pour un problème posé, si l'on veut utiliser un réseau de neurones multicouches et la loi Delta généralisée, il faut encore répondre à un certain nombre de questions, pour lesquelles il n'existe malheureusement pas de réponse précise, notamment:

- Nombre de couches.

  Pour un problème non linéaire, il a été démontré qu'un minimum de 3 couches (entrées - couche cachée - sorties) est nécessaire. Un plus grand nombre permet une meilleure généralisation, mais a l'inconvénient de complexifier la structure (temps d'apprentissage supérieur, clarté du système réduite,...). C'est au concepteur à trouver le juste milieu en fonction du problème à résoudre. Il est rare que plus de 3 couches se justifie.

- Nombre de neurones dans chaque couche.

  Pour la couche d'entrée, le nombre de neurones est directement déterminé par la dimension des données (bien que l'on puisse envisager des prétraitements: compression, suppression,...). Ainsi, si on veut utiliser les mêmes données qu'Altman pour établir la santé d'une entreprise, on utilisera 5 neurones en entrées (plus le seuil): un pour chaque ratio.

  Pour la couche de sortie, le nombre de neurones est également aisé à fixer puisqu'il est défini par le format de la réponse souhaitée. Les variations proviennent du codage choisi. Par exemple, toujours pour notre distinction entre sociétés saines et sociétés à risques, on peut utiliser 2 neurones en sortie. Le premier représente la bonne santé et le second la situation malsaine. Une entreprise appartiendra à la classe associée au neurone dont la valeur de sortie est la plus grande. Une autre possibilité est d'utiliser un seul neurone en sortie. La valeur obtenue correspondrait au numéro de la classe.

  Le problème devient plus sérieux pour les couches cachées. Il est impossible de trouver à priori le nombre parfait de neurones. Il existe différentes

techniques de décision. La plus simple et la plus grossière est de procéder par essais et erreurs. Un mauvais choix peut avoir des répercussions graves. S'il y a trop peu de neurones, le réseau sera incapable de représenter correctement le problème. S'il y en a trop, des bruits parasites pourraient perturber les résultats, en particulier lors d'approximation de fonctions.

- Les données

  Il ne faut surtout pas oublier que c'est l'apprentissage du réseau qui déterminera l'importance de chaque entrée en lui fixant un poids approprié. Si une entrée est fournie alors qu'elle n'a aucun rapport avec le problème à résoudre, le poids associé devrait être fixé à zéro par l'algorithme d'apprentissage. Il n'est donc pas nécessaire de réaliser au préalable une analyse pour déterminer les données utiles. Au contraire, le monde étant très complexe et parfois mal compris, il vaut parfois mieux laisser l'algorithme d'apprentissage déterminer les relations entre entrées et sorties plutôt que de risquer de retirer une information importante. Ainsi, dans notre exemple et dans la mesure du possible, on utilisera des ratios supplémentaires à ceux d'Altman pour étudier la santé de la société.

  Remarquons aussi que nous n'avons pas eu besoin de poser une hypothèse à priori sur la forme des distributions des variables observées, sur la forme de la relation qui unit ces variables, sur la forme des résidus du modèle ou sur l'existence d'interaction entre les variables (au contraire des modèles de régression qui supposent la normalité des variables aléatoires étudiées, l'absence de colinéarité, des résidus de variance constante et non-autocorrélés). Le réseau reçoit des données brutes et l'algorithme d'apprentissage analyse leur structure.

- Fonction associée à chaque neurone.

  Il est possible d'affecter des fonctions différentes à chaque neurone. Cela n'est cependant fait que dans des cas particuliers.

  La fonction d'activation la plus courante est:

$$f(net) = \frac{1}{1 + \exp(-\lambda net)} \tag{25}$$

  où λ est une constante caractérisant la pente. Une représentation en est donnée à la Figure 8.
  Cette fonction est choisie car elle est de type sigmoïde et possède une dérivée facile à manipuler:

$$f'(net) = \lambda f(net)(1 - f(net)) \tag{26}$$

  Le signal d'erreur delta (23)devient simplement pour $\lambda = 1$:

$$\delta_k = (d_k - o_k)o_k(1 - o_k) \tag{27}$$

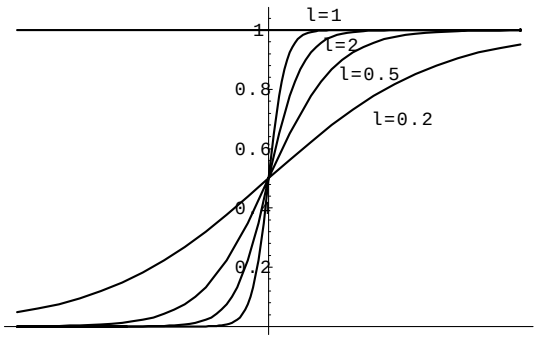

Figure 8: Fonctions d'activation

On se limitera généralement à cette valeur de λ, car on pourra obtenir des résultats similaires pour d'autres λ en adaptant en conséquence la constante d'apprentissage η.

- Constante d'apprentissage η.

  A l'aide du gradient, on a déterminé la direction à suivre pour diminuer l'erreur. La constante d'apprentissage η va elle quantifier l'importance de cette correction. De nouveau la valeur optimale est liée au problème à résoudre. Il faudra procéder par essais pour déterminer sa meilleure valeur. Celle-ci devra être petite pour assurer une vraie descente de gradient et permettre de trouver le minimum même s'il est au fond d'une vallée étroite (sinon on risque de le dépasser à chaque fois). Cependant, une valeur faible allonge significativement le temps nécessaire à l'apprentissage. Un juste milieu doit être trouvé.

- La valeur initiale des poids.

  La valeur initiale des poids peut avoir une grande importance. S'ils sont proches de la valeur idéale, l'apprentissage sera rapide. S'ils en sont éloignés, la convergence sera plus lente, mais surtout pourrait, si le réseau est mal conçu, s'arrêter au premier minimum local de la fonction d'erreur rencontré. Si le réseau est suffisament important et les poids initiaux faibles, la loi Delta généralisée est assez robuste. Une fixation aléatoire est la plupart du temps utilisée.

6.5 Améliorations

L'algorithme de rétropropagation du gradient tel qu'il vient d'être présenté constitue la base. De nombreuses améliorations ont été présentées dans la littérature. Entre autres:

- Utilisation d'un moment.

  En plus de la correction des poids actuels selon le gradient, on ajoute une fraction de l'ajustement qui a été réalisé aux itérations précédentes. On accélère ainsi la convergence de l'algorithme.

$$\Delta w(t) = -\eta \nabla E(t) + \alpha \Delta w(t-1) \tag{28}$$

- Validation croisée pour la détection de surapprentissage (d'où incapacité à généraliser).
- Utilisation de la matrice Hessienne (méthode de Newton et quasi-Newton).
- Evolution de la constante d'apprentissage η au cours du temps.
- Utilisation d'une ligne de recherche.
- Réinitialisation de l'algorithme après un certain temps.

La structure du réseau est aussi revue. Pour éviter une taille trop importante du réseau et un apprentissage trop long, on peut utiliser la technique du "pruning" pour supprimer les connexions peu importantes.

Enfin, les données peuvent être prétraitées. Pour améliorer la généralisation du réseau, les entrées sont parfois légèrement bruitées. Elles peuvent aussi être compressées. On obtiendra ainsi un réseau plus petit. Outre le fait qu'un réseau avec peu de connexions sera plus facilement implémentable et à meilleur marché qu'un réseau complexe, on obtient surtout un gain appréciable de traitements.

7 Analyse complète: étude de cas

7.1 Présentation du problème

Reprenons le problème présenté précédemment concernant un estimateur de la santé financière des entreprises. Nous allons tenter de prédire à l'aide des mêmes ratios qu'Altman le risque de faillite des sociétés commerciales belges en 1993 et cela 1, 2 et 3 ans à l'avance. Pour cela, un simulateur de réseaux de neurones a été écrit en respectant scrupuleusement la structure multicouches et l'algorithme de rétropropragation tels qu'ils ont été définis plus haut (sans aucune optimisation).

7.2 Phase 1: collecte des données

Les 5 ratios utilisés par Altman ont été extraits de la banque de données diffusée par la Centrale des Bilans de la Banque Nationale Belge pour les entreprises commerciales (NACE 06) et cela pour les exercices bilantaires de 1990 à 1992. Les données ont été réparties pour chaque année entre un fichier utilisé lors de l'apprentissage et un fichier utilisé ensuite pour tester la capacité de généralisation.

Année	Apprentissage	Test
1990	8 282	2 363
1991	10 034	2 868
1992	13 329	3 810

Table 1: Nombre d'entreprises dans chaque fichier

6 ensembles de données ont ainsi été crées où chaque ligne décrit une société et chaque colonne est un des ratios. Un expert financier, sur base des ratios d'Altman en 1993, a alors réparti chacune de ces entreprises en deux catégories: celles qui seront saines en 1993 (groupe 1) et celles qui au contraire risquent la faillite à partir de 1993 (groupe 0). Cet indicateur a été ajouté comme sixième colonne aux différents fichiers. Les 5 premières lignes du fichier d'apprentissage de 1990 sont présentées dans la table 2.

X_1	X_2	X_3	X_4	X_5	Santé
0.23	0.18	-0.01	0.31	0.13	0
0.84	0.78	0.06	5.77	0.47	1
-0.06	0.08	0.01	0.15	0.38	0
0.16	0.11	-0.01	0.15	0.42	0
0.52	0.54	0.04	1.43	0.46	1

Table 2: Début du fichier d'apprentissage de 1990

7.3 Phase 2: conception du réseau

En utilisant la structure multicouches et l'algorithme élémentaire tels qu'ils ont été décrits, il ne reste que 2 paramètres à définir: le nombre de neurones en couche cachée et la constante d'apprentissage η. On procède par essais successifs pour déterminer les valeurs adéquates.

Le fichier d'apprentissage d'une année est présenté au simulateur de réseaux de neurones. Celui-ci va alors réaliser un nombre déterminé d'itérations. Une iteration consiste en la présentation des 5 ratios de chacune des entreprises à l'entrée du réseau, à la propagation de ces informations jusqu'à la sortie et à la rétropropagation de l'erreur vers l'entrée avec modification des poids. L'apprentissage s'arrête quand il n'est plus possible de faire diminuer l'erreur ou dès que la diminution n'est plus significative.

On présente ensuite au simulateur le fichier de test de la même année. Les 5 ratios de chaque société sont propagés vers la sortie et on peut comparer les résultats avec ceux donnés par l'expert financier.

En répétant ces opérations pour différents nombres de neurones en couche cachée et pour différentes valeurs de η, on constate en étudiant les pourcentages de classification correcte, qu'il n'est pas nécessaire d'utiliser plus de 2 neurones en couche cachée et qu'une valeur adéquate de η est 0.01. Utiliser plus de neurones n'améliore pas les résultats, mais allonge la phase d'apprentissage. Des valeurs supérieures de η donnent des résultats moins précis et un temps d'apprentissage plus long (égarement). De plus l'évolution de la fonction d'erreur au cours du temps est assez chaotique. Quelques simulations pour les différentes années sont synthétisées dans les tableaux 3 et 4.

Configuration 1992	Apprentissage		Test
	Correct	Nbre iter	Correct
2 neurones			
η =0.2	83.82%	120	83.08%
η =0.01	83.97%	442	83.52%
η =0.001	84.04%	2 000	83.31%
10 neurones			
η =0.2	83.22%	860	82.83%
η =0.01	84.23%	942	83.76%
η =0.001	83.99%	2 000	83.20%

Table 3: Résultats des simulations pour 1992

Configuration 1990	Apprentissage Correct	Nbre iter	Test Correct
2 neurones			
η =0.2	70.60%	767	70.5%
η =0.01	70.78%	500	70.63%
η =0.001	70.89%	3 000	70.21%
10 neurones			
η =0.2	70.82%	852	69.45%
η =0.01	70.78%	600	70.5%
η =0.001	70.47%	2 000	70.60%

Table 4: Résultats des simulations pour 1990

7.4 Phase 3: Analyse des résultats

D'après les tableaux 3 et 4, les réseaux de neurones sont aptes à traiter ce genre de problème. 84% de classification correcte pour les données d'apprentissage de 1992 est un résultat honnête. De plus, le réseau a bien généralisé car pour un fichier test, le réseau s'en tire presque aussi bien. Une légère baisse est normale; il est plus facile d'analyser des informations qui ont déjà été présentées au réseau durant l'apprentissage qu'un ensemble inconnu. Pour les autres années, les scores restent bons. Bien entendu, ils se dégradent plus on remonte dans le temps. Il est plus difficile de faire des prévisions 3 ans à l'avance que l'année précédente.

Cette première analyse n'est pas suffisante. On sait que l'on a obtenu un réseau idéal pour les données d'apprentissage. On sait aussi qu'il donne de bons résultats pour un fichier test différent. Par contre, nous ne savons pas encore si ce réseau est le meilleur pour ce fichier test ou un autre. En effet, il est possible qu'après un certain nombre d'itérations le réseau ait trop étudié les caractéristiques de l'ensemble d'apprentissage et se soit spécialisé dans celui-ci. Pour s'en assurer, on utilise la technique de la validation croisée. Après chaque itération de l'apprentissage, on obtient un réseau intermédiaire auquel on présente un fichier test (ou fichier de validation). La fonction d'erreur pour le fichier d'apprentissage va décroître au cours du temps. Par contre, la fonction d'erreur calculée pour le fichier de validation recommencera à augmenter s'il y a surapprentissage. A ce moment, il vaut mieux s'arrêter.

Cette technique a été utilisée dans notre exemple. Un troisième fichier a été constitué et présenté au réseau durant l'apprentissage. La figure 9 montre l'évolution de la fonction d'erreur au cours du temps pour le fichier d'apprentissage et le fichier de validation de 1992. On constate que l'erreur diminue continuellement pour les deux fichiers et tend vers une asymptote horizontale.

Le fichier d'apprentissage était ici important et contenait un échantillon suffisamment représentatif des entreprises que pour ne pas avoir de problèmes de surapprentissage. Remarquons aussi en passant qu'après environ 440 itérations, le meilleur réseau est atteint; l'erreur ne diminue pratiquement plus.

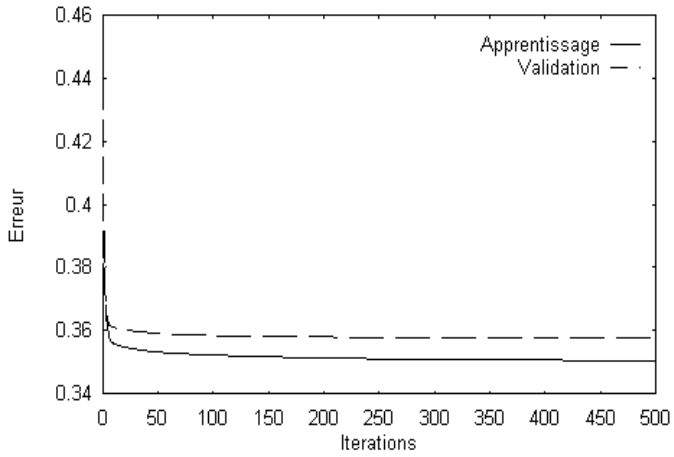

Figure 9: Evolution de l'erreur

Maintenant que nous avons une certaine confiance dans les performances de ce réseau, essayons de mieux comprendre les résultats. On sait que pour 1992, le réseau diagnostiquera correctement la situation 84 fois sur 100. Inversément, il ne se trompera que 16 fois sur 100. Pour un financier, cette erreur n'aura pas toujours le même prix. Si une société est jugée à risques alors qu'elle ne l'est pas, il y aura juste un coût d'opportunité. Par contre, si une société à risques est jugée saine, les dégats pourraient être beaucoup plus considérables. Il paraît donc important de considérer les deux cas.

La table 5 montre des résultats plus détaillés obtenus pour les fichiers tests. Pour chaque année, le réseau sous-estime le nombre de sociétés saines et surestime le nombre de sociétés à risques. Par exemple en 1991, 17.71% des entreprises sont considérées à tort malsaines et 7.22% seulement sont annoncées erronément saines. L'investisseur qui ferait totale confiance au réseau subirait donc essentiellement des coûts d'opportunités.

Si une erreur de classification est vraiment très néfaste, on peut encore pousser plus loin l'analyse. Il est possible en étudiant les résultats de diminuer le pourcentage de classification incorrecte. On a représenté une société parfaitement saine par une sortie valant 1 et une société à risques par une sortie nulle. Le réseau fournit des résultats continus entre 0 et 1. La valeur pivot 0.5 représentait jusqu'à présent la scission entre les deux catégories. Cependant, on ne devrait pas avoir la même confiance en une valeur de 0.499 qu'en une valeur 0. Le risque d'erreur est beaucoup plus grand pour la première. La solution envisagée ici est de revoir le partitionnement de l'espace de sortie. Par exemple, en choi-

Obtenu	Désiré		Total
	Sain	Risqué	
1992			
Sain	30.24%	7.87%	38.11%
Risqué	8.61%	53.28%	61.89%
Total	38.85%	61.15%	100%
1991			
Sain	20.82%	7.22%	28.04%
Risqué	17.71%	54.25%	71.96%
Total	38.53%	61.47%	100%
1990			
Sain	17.22%	7.74%	24.96%
Risqué	21.63%	53.41%	75.04%
Total	38.85%	61.15%	100%

Table 5: Résultats détaillés des simulations pour l'ensemble test

Obtenu	Désiré		Total
	Sain	Risqué	
1992			
Sain	28.54%	6.80%	35.34%
Incertitude	2.83%	2.57%	5.40%
Risqué	7.48%	51.78%	59.26%
Total	38.85%	61.15%	100%
1991			
Sain	15.73%	3.94%	19.67%
Incertitude	9.20%	8.68%	17.88%
Risqué	13.60%	48.85%	62.45%
Total	38.53%	61.47%	100%
1990			
Sain	13.16%	5.12%	18.28%
Incertitude	11.47%	10.71%	22.18%
Risqué	14.22%	45.32%	59.54%
Total	38.85%	61.15%	100%

Table 6: Résultats affinés des simulations pour l'ensemble test

sissant comme borne supérieure des sociétés à risques la valeur 0.4 et comme borne inférieure des sociétés saines la valeur 0.6, le risque d'erreur chute. Il en découle également une diminution des pourcentages de classification correcte et la création d'une zone d'indécision. Pour les sociétés dont le score se situe entre 0.4 et 0.6, il est préférable de laisser trancher un expert financier disposant d'une plus grande panoplie d'outils de décision. Suivant le degré d'aversion pour le risque, d'autres bornes peuvent être envisagées. Signalons aussi que d'autres méthodes sont envisageables, en particulier si l'on avait choisi de coder chacune des classes par un neurone en sortie.

Les résultats présentés à la table 5 sont revus avec cette méthode dans la table 6. On constate comme annoncé une diminution de l'erreur avec comme prix la mise sur le coté d'une fraction des entreprises. Pour 1990, le pourcentage d'erreur chute de 10%.

8 Mise en oeuvre

Les réseaux de neurones sont actuellement réalisés de nombreuses manières différentes. En général, on utilise un simulateur sur un ordinateur séquentiel conventionnel. Cependant, pour obtenir les meilleurs résultats, il est nécessaire de les implémenter physiquement en tenant compte qu'ils sont massivement parallèles: chaque neurone peut être vu comme un processeur indépendant, aux fonctions très simples. Le problème auquel on est confronté actuellement est celui des connexions. Il est très coûteux de réaliser un circuit électronique à très forte connectivité en VLSI (Very Large Scale Integration). Cependant, des études montrent que des implémentations électroniques des réseaux sont réalisables et prometteuses.

En attendant les résultats de ce développement technique, on utilise des simulateurs sur des ordinateurs digitaux. Ces derniers sont généralement dénommés neuro-ordinateurs programmables. Les plus performants sont, bien entendu, ceux qui peuvent travailler en parallèle. Malgré cela et d'autres améliorations comme des cartes accélératrices, ces neuro-ordinateurs ne peuvent rivaliser avec une implémentation spécifique.

9 Conclusions

9.1 Des propriétés

L'intérêt porté aujourd'hui aux réseaux de neurones tient sa justification dans les quelques propriétés fascinantes qu'ils possèdent. Citons les plus importantes: le parallélisme, la capacité d'adaptation, la mémoire distribuée, la capacité de généralisation et la facilité de construction.

Le parallélisme se situe à la base de l'architecture des réseaux de neurones considérés comme ensembles d'entités élémentaires qui travaillent simultanément. Le parallélisme permet une rapidité de calcul supérieure, mais exige de penser et de poser différemment les problèmes à résoudre.

La capacité d'adaptation permet au réseau de tenir compte de nouvelles contraintes ou de nouvelles données du monde extérieur. Cette capacité présente un intérêt déterminant pour tous les problèmes évolutifs. Il faut, pour les résoudre, pouvoir tenir compte de situations non encore connues.

Dans les réseaux de neurones, la "mémoire" d'un fait correspond à une carte d'activation de l'ensemble des neurones. Cela permet une meilleure résistance au bruit. La perte d'un élément ne correspond pas à la perte d'un fait mémorisé, mais à une dégradation, d'autant plus faible qu'il y a de synapses. De plus, la recherche d'un fait ne nécessite pas la connaissance de l'endroit de stockage, le réseau entier se chargeant de le restituer.

La capacité de généralisation est essentielle. Elle assure que le réseau donnera une bonne solution pour une entrée ne figurant pas dans l'ensemble d'apprentissage. Nombre de problèmes résolus par des experts le sont de façon plus ou moins intuitive, ce qui rend difficile l'exposé explicite des connaissances et des règles nécessaires à leur solution. Le réseau adopte une démarche semblable à celle de l'expert.

Le principe des réseaux de neurones et leur structure sont assez simples. La simulation informatique ne nécessite qu'un temps de développement assez court.

9.2 Des limites

Un des principaux reproches fait aux réseaux de neurones est l'impossibilité d'expliquer les résultats qu'ils fournissent. Les réseaux se présentent comme des boîtes noires dont les règles de fonctionnement sont inconnues. Ils créent eux-mêmes leur représentation lors de l'apprentissage. La qualité de leurs performances ne peut être mesurée que par des méthodes statistiques, ce qui amène parfois une certaine méfiance de la part des utilisateurs potentiels.
Le second problème a déjà été invoqué: la mise en oeuvre physique. Les réseaux de neurones seront optimums quand ils auront leur propre support. Différentes solutions ont été envisagées pour atténuer ce problème; notamment diminuer le nombre de neurones (par complexification de leur structure, par prétraitements).

References

[1] Altman E.I., *The Prediction of Corporate Bankruptcy: A Discriminant Analysis*, Garland, New-York, 1988.

[2] Blayo F., *Réseaux neuronaux*, laboratoire de Microinformatique, avril 1992.

[3] Cottrell M., *Introduction aux réseaux de neurones artificiels et spécificité des méthodes neuronales*, Les réseaux de Neurones en Finance: Conception et Applications, D facto publications, pp. 23-58, 1995.

[4] Davalo E. - Naïm P., *Des réseaux de neurones*, Eyrolles, 2ème édition, 1990.

[5] De Bodt E., Cottrell M. et Levasseur M., *Les réseaux de neurones en finance. Principes et revue de la littérature*, Finance, Vol 16, pp 25-91, 1/1995.

[6] Fombellida M., *Architectures connexionistes évolutives: algorithmes d'apprentissage et heuristiques*, Thèse doctorale, Faculté des Sciences Appliquées, Université de Liège, 1993.

[7] Press W.H. - Flannery B.P. - Teukolsky S.A. - Vetterling W.T., *Numerical Recipes*, Cambridge University Press.

[8] Schyns M., *Prétraitement de données en reconnaissance de formes par réseaux de neurones artificiels*, travail de fin d'études, Faculté des Sciences Appliquées, Université de Liège, 1993.

[9] Zirilli J.S., *Financial Prediction using Neural Networks*, International Thomson Computer Press, 1997.

[10] Zurada J.M., *Introduction to Artificial Neural Systems*, West Publishing Company, 1992.