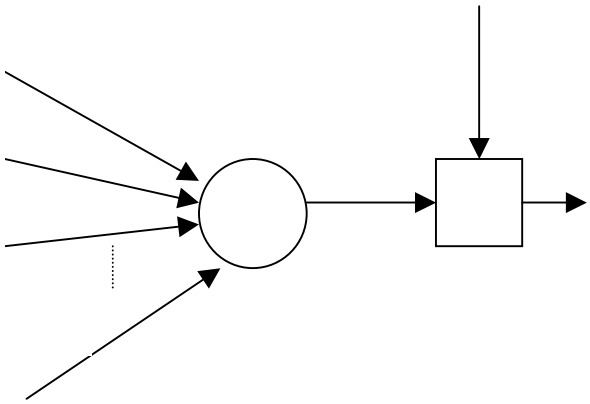
Le neurone (perceptron)



Les équations qui caractérise le neurone j sont:

$$\mathbf{net}_j = \sum_{i=1}^{n} \mathbf{w}_{ji} . \mathbf{x}_i \qquad \textbf{(2.1)}$$

$$\mathbf{o}_j = \mathbf{f}(\sum_{i=1}^{n} \mathbf{w}_{ji}\mathbf{x}_i - \theta_j) \qquad \textbf{(2.2)}$$

où:

- $\mathbf{w_{ji}}$ qui multiple l'entrée $\mathbf{x_i}$ s'appelle **poids** ;
- $\mathbf{x_i}$ est l'entrée i;

Les limites du perceptron (neurone)

On peut montrer que le perceptron n'apprend que des catégories linéairement séparables.

Deux catégories sont linéairement séparables si elles sont separables par une droite (Fig.3.5). Les fonctions logiques AND, OR and NOT impliquent des classifications séparables par une droite

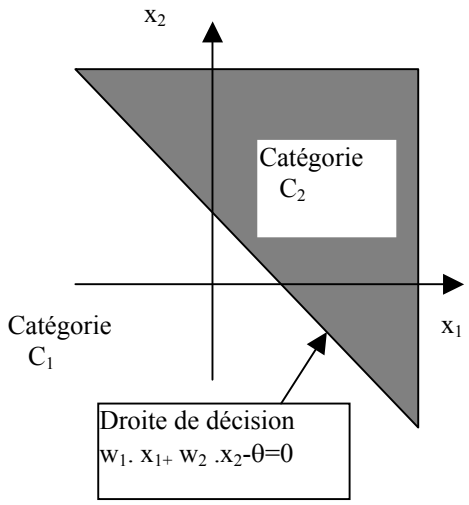


Fig.3.5 Deux catégories linéairement séparables

Le cas particulier plus connu de catégories qui ne possèdent pas la propriété de séparablilité linéaire est la fonction XOR (OU EXCLUSIVE). La fonction de sortie est donnée par la relation : $\mathbf{x_1 . \bar{x}_2 + \bar{x}_1 . x_2}$ Le tableau de vérité suit:

Tableau de vérité pour la fonction XOR

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

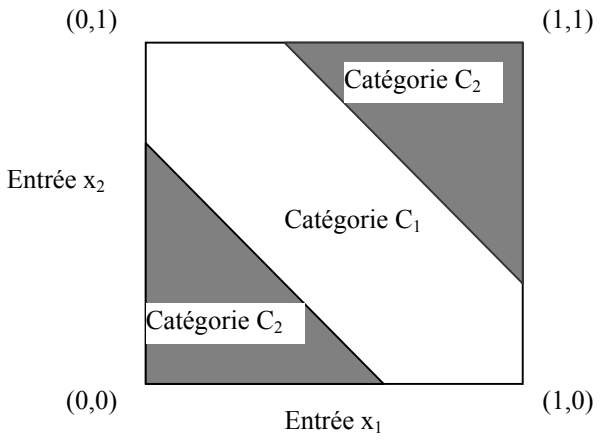


Fig.3.6 L'espace des entrées dans le cas du problème XOR

Pour des limites de décision plus complexe sont nécessaires plusieurs couches. Pour réaliser la fonction XOR sont nécessaires deux couches de perceptrons. Dans la Fig.3.7 sont présentés deux réseaux possibles d'apprendre la fonction XOR.

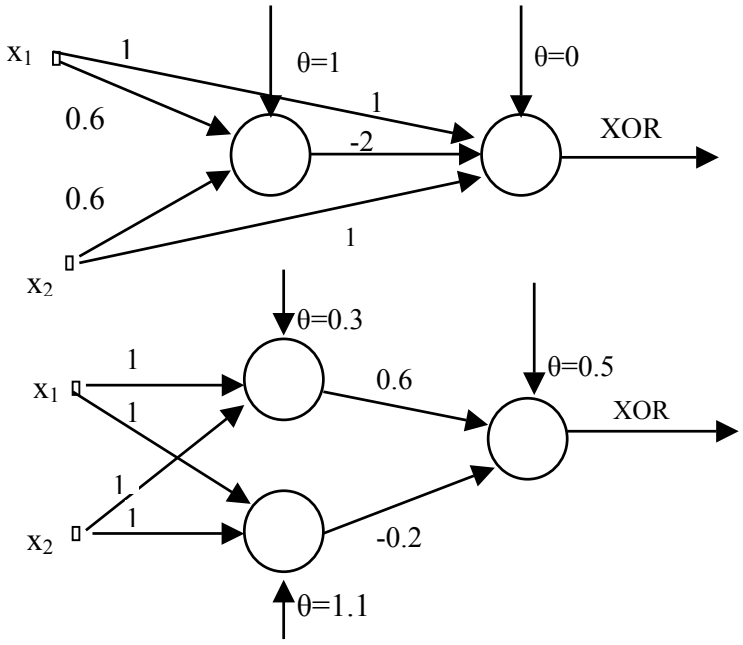


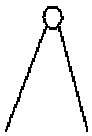
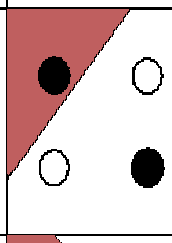
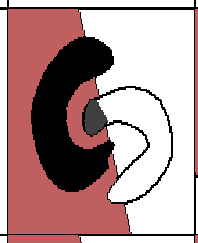
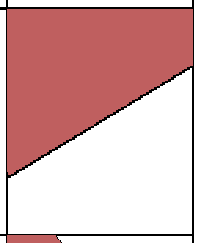
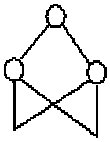
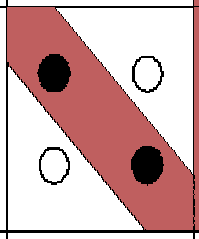
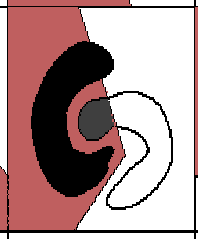
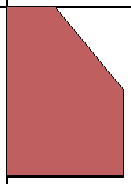
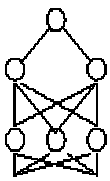
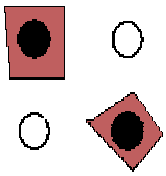
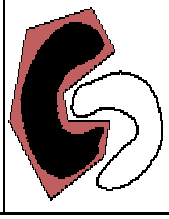
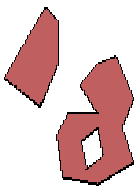
Fig.3.7 Deux réseaux pour apprendre XOR

Un réseau de neurones avec deux couches peut former des régions de décisions convexes obtenues par l'intersection des semiplanes générée de chaque neurone de la premier couche, comme dans l'exemple de Fig.3.8a.

Un réseau de neurones avec deux couches est capable d'identifier n'importe quelle région convexe, a condition que la couche cachée contient un nombre suffisant de neurones et que les poids sont proprement adaptés.

Un réseau de neurones avec trois couches peut séparer des régions de décision arbitraires, la complexité étant limitée seulement par le nombre de neurones. De plus, on a démontré que la précision d'une telle classification nonlinéaire peut être arbitraire .

C'est à dire qu'un réseau avec trois couches est capable de réaliser n'importe quelle transformation non linéaire avec une n'importe quelle précision, en utilisant un nombre suffisant de neurones.

Structure	Régions décision-nelles	Pb du XOR	Régions pénét-rantes	Forme générale
 1 couche	Demi Plan			
 2 couches	Arbitraire dépend du nombre de couches cachées			
 3 couches	Arbitraire dépend du nombre de couches cachées			

Les réseaux de neurones multicouches

4.1 L'architecture

L'incapacité des réseaux de neurones avec deux couches de transformer des models arbitraires d'entrée en models arbitraires de sortie a été dépassée avec les réseaux multicouches qui ont une ou plusieurs couches intermédiaires entre la couche d'entrée et cellui de la sortie.

- **La couche d'entrée** ne joue généralement que le rôle d'interface avec l'extérieur, c'est à dire qu'elle n'effectue pas de traitement sur l'information.
- **Les couches intermédiaires (couches cachées)** n'ont aucune interaction directe avec l'environnement. Chaque neurone qui compose une de ces couches est généralement connectée à chaque unité de la couche précédente dont il reçoit l'information..
- **La couche de sortie** traite également l'information qu'elle reçoit de la couche précédant mais elle ne communique pas son résultat aux autres neurones. L'information fournie par chacune de ses unités constitue le vecteur de sortie du réseau. Celui-ci est donc le résultat du traitement d'un vecteur d'entrée par l'ensemble du réseau.

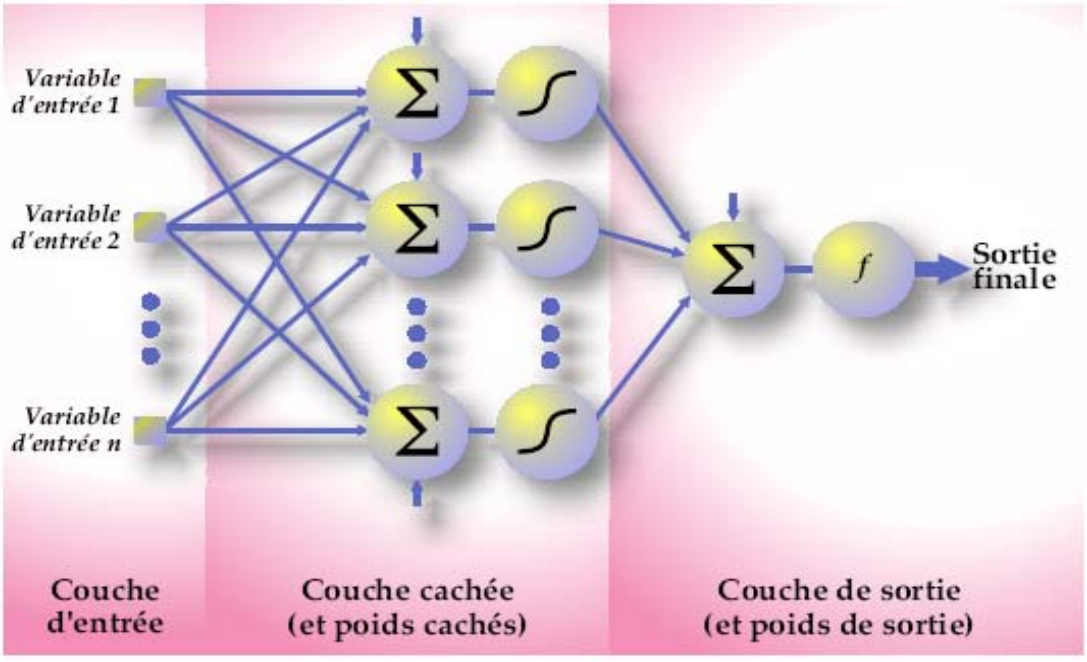


Fig.1 L'architecture d'un réseau de neurones multicouche

Parce que l'information se propage en avant, de l'entrée à la sortie, ces types de **réseau** sont nommés **"en avant"** . Chaque unité de la couche intermédiaire traite l'information reçue avec une fonction d'activation simple. La composition de ces fonctions simples donne pour résultat la fonction globale réalisée par le réseau entier. Un réseau de neurones génère donc une certaine application de l'espace des entrées possibles dans celui des sorties.

Suivant le but recherché (régression non linéaire ou classification) la fonction d'activation de la couche de sortie pourra être linéaire ou non.

L'expérience a prouvé la nécessité d'utiliser une fonction d'activation au moins semilinéaire dans les couches cachées pour dépasser les performances des réseaux avec deux couches (entrée- sortie).

$$a_i(t+1) = f(net) = \begin{cases} 0, & \text{pentru } net(t) < -\theta \\ \dfrac{net(t)+\theta}{2\theta}, & \text{pentru } -\theta < net < \theta \\ 1, & \text{pentru } net(t) > \theta \end{cases} \qquad (4.1)$$

Usuellement elle est une fonction logistique, nommée aussi sigmoïde:

$$o_i(t) = \frac{1}{1 + e^{-\beta . net_i(t)}} \qquad (4.2)$$

où β est un facteur de proportionnalité. L'avantage de la fonction sigmoïde est sa dérivée qui est simple:

$$f'(x) = f(x)[1 - f(x)] \qquad (4.3)$$

La fonction tangente hyperbolique est aussi souvent utilisée lorsque le domaine de la réponse est l'intervalle [-1, +1].Elle se calcule comme :

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (4.4)$$

Sa dérivée est aussi relativement facile à calculer :

$$\frac{d\tanh(x)}{dx} = [\sec h(x)]^2 = \frac{4}{(e^x + e^{-x})^2} \qquad (4.5)$$

4.2 L'algorithme de la rétro-propagation de l'erreur

L'algorithme de rétro-propagation de l'erreur a été mis au point indépendamment par des nombreux auteurs dont les compétences vont de l'analyse numérique (Bryson & Ho, 1969) et de la statistique (Werbos 1974) aux réseaux de neurones (Parker 1982 , Le Cun 1986, Rumelhart, Hinton & Wiliam 1986). L'algorithme est supervisé et comprenne deux étapes. Il est connu aussi sur le nom de "règle delta généralisé", nomme introduit de groupe autour Rumelhart & Mc Clelland dans leur livre "Parallel Distributed Processing".

4.2.1 La première étape

Dans une première étape, le réseau propage l'information, de la couche d'entrée à la première couche cachée et ensuite à la couche suivante jusqu'à la couche de la sortie. Fixons les notations suivantes:

$\mathbf{N}$: nombre d'entrées du réseau .

$\mathbf{N_h}$: nombre d'unités sur la couche cachée du réseau.

$\mathbf{N_{output}}$: nombre d'unités sur la couche de sortie.

L'entrée nette de chaque unité cachée est donnée pour chaque p stimulus par la relation:

$$\mathbf{net_{pj} = \sum_i w_{ji} . x_i + \theta_j} \qquad \mathbf{(4.6)}$$

La sortie de chaque unité cachée est une fonction de la somme pondérée des composantes du vecteur d 'entrée x et du seuil θ :

$$\mathbf{o_j = f(\sum_{i=1}^{N_{input}} w_{ji} x_i + \theta_j)} \qquad \mathbf{(4.7)}$$

où f est en général soit une fonction sigmoïde soit la fonction tangente hyperbolique. Dans le cas d'un réseau muni d'une seule couche cachée, la sortie o_i du neurone k de la couche de sortie du réseau s'exprime en fonction de l'information venant de la couche cachée par :

$$\mathbf{o_k = f(\sum_{j=1}^{N_h} w_{kj} o_j + \theta_k)} \quad \mathbf{ou} \quad \mathbf{o_k = \sum_{j=1}^{N_h} w_{ij} o_j + \theta_k} \qquad k=1,.., N_{output} \qquad \mathbf{(4.8)}$$

La fonction totale réalisée par le réseau sur la sortie i s'écrit ainsi :

$$\mathbf{o_k = f(\sum_{k=1}^{N_h} w_{kj} f(\sum_{i=1}^{N_{input}} w_{ji} x_i + \theta_j) + \theta_k)} \qquad \mathbf{(4.9)}$$

Les fonctions réalisées par les neurones d'une même couche sont du même type, seuls les vecteurs poids et les seuils changent d'une unité à l'autre.

En comparant la réponse courante du chaque neurone de la sortie o_k avec la réponse désirée d_{pk} et en tenant compte de l'état d'activation on évalue une erreur δ_{pk}:

$$\delta_{pk} = (d_{pk} - o_{pk}).f'(net_{pk}) \quad (4.10)$$

où f' este la dérivée de la fonction d'activation du neurone.

4.2.2 La deuxième étape

Pendant la deuxième étape l'erreur entre la sortie courante et la réponse désirée se propage de la sortie à l'entrée en générant la modification de l'intensité des connexions. La règle d'apprentissage pour les connexions entre la dernière couche cachée et la sortie est :

$$\Delta_p w_{kj} = \eta.\delta_{pk}.o_{pj} \quad (4.11)$$

où η est la constante d'apprentissage, qui peut avoir une valeure appartenante à l'intervalle (0,1).

Pour les neurones cachés (indexés par j) l'erreur δ_{pj} se calcule avec la relation suivante:

$$\delta_{pj} = (\sum_k \delta_{pk}.w_{kj}).f`(net_{pj}) \quad (4.12)$$

Suivant, on modifie l'intensité des connexions entre l'entrée est la couche cachée avec une règle d'apprentissage (qui peut être la même) (4.11):

$$\Delta_p w_{ji} = \eta.\delta_j.o_{pi} \quad (4.13)$$

S'il y a plusieurs couches cachées, pour chaque unité de la couche cachée l'erreur s'évalue avec la relation (4.12). Les erreurs se propagent successivement, de couche à couche, de la sortie vers l'entrée.

Dans un réseau de neurones peuvent coexister aussi des connexions avec une intensité fixe. S'il y a des unités de sortie aussi dans les couches cachées, celles ci ont deux types des erreurs. Il y a une erreur obtenue en comparant la sortie courante avec la réponse désirée et une erreur déterminée par la propagation des erreurs des unités de la sortie couplées avec l'unité cachée.

L'algorithme de rétro-propagation de l'erreur minimise le carré de l'erreur dans chaque itération . On peut démontrer que **l'algorithme de rétro-propagation de**

l'erreur implemente un gradient d'erreur carrée descendent dans l'espace des poids:

$$w(t+1) = w(t) - \eta . \frac{dE}{dw} \qquad (4.14)$$

où dE/dw est le gradient aléatoire inconnu d'erreur de paire (x_i, t_i).

Applications des réseaux de neurones multicouches :

Les RN sont des approximateurs universels.

Jusqu'au présent la rétro- propagation de l'erreur a été utilisée dans un très grand nombre des applications, comme par exemple :

- la détection et la classification des signaux radar (Haykin 1992) ;
- filtrage adaptatif de chenal de communication [182], [67] (Widrow 1960, Gibson, Cowan 1991).
- analyse de courbes, classification de formes (EEG, diagrammes) ;
- la reconnaissance de chiffres et autres symboles écrits (Le Cun1990) ;
- la reconnaissance de la parole (des phonèmes) (Botttou,1988, Cohen 1993);
- la lecture des textes (Guyon et al.) Le succès de la méthode a permis de fournir une des premières applications industrielles des réseaux de neurones ;
- machine a prononcer l'anglais. Sejnovski et Rosenberg [82] ont entraîné un réseau de neurones, baptisé NET-TALK à prononcer l'anglais à partir d'un texte écrit. Le réseau a été composé de 203 unités d'entrée, d'une couche cachée de 80 neurones et d'une couche de sortie de 30 unités. Ils ont montré que l'histoire d'apprentissage d'un réseau de neurones ressemble à l'apprentissage humain. Le réseau a commencé par être incapable de prononcer de sons cohérents, puis il a babillé et finalement, après un temps assez long, il a prononcé les mots correctement, intelligiblement
- le modelage d'un système inconnu et le contrôle d'un système dynamique [127] (Narendra et Partharsarathy) ;
- le contrôle d'un bras mobile de robot [102] (Josin 1988, Nguyen et Widrow 1990) ;
- le contrôle d'un véhicule autonome [84];
- analyse financière
- aide à la prise de décision.

4.2.3 Exemple

Soit un réseau avec I=3 neurones d'entrée, L=2 neurones cachés et J=3 neurones de sortie. On considère une fonction d'activation sigmoïde donne par la

relation (4.2) avec β=1, la vitesse d'apprentissage η=1. Le but du réseau est d'apprendre à associer un ensemble de stimuli à un ensemble de réponses. Pour cet exemple, il doit apprendre l'association suivante:

le stimulus $\mathbf{x}=\begin{bmatrix}\mathbf{1}\\ \mathbf{2}\\ \mathbf{3}\end{bmatrix}$ doit donner la réponse $\mathbf{t}=\begin{bmatrix}\mathbf{0.1}\\ \mathbf{0.3}\\ \mathbf{0.7}\end{bmatrix}$ **(4.21)**

Solution

La matrice de connexions W relie les neurones de la couche d'entrée aux neurones de la couche cachée. Elle est d'ordre $L \times I=2 \times 3$.

$$\mathbf{W}=\begin{bmatrix}\mathbf{w}_{11} & \mathbf{w}_{12} & \mathbf{w}_{13}\\ \mathbf{w}_{21} & \mathbf{w}_{22} & \mathbf{w}_{23}\end{bmatrix}=\begin{bmatrix}\mathbf{.5} & \mathbf{.3} & \mathbf{.1}\\ \mathbf{.3} & \mathbf{.2} & \mathbf{.1}\end{bmatrix} \quad \textbf{(4.22)}$$

La matrice de connexions Z relie les neurones de la couche cachée aux neurones de la couche de sortie (la notation Z est utilisée plutôt que $Z_{[t]}$ pour alléger l'écriture. Elle est d'ordre $J \times L=3 \times 2$.

$$\mathbf{Z}=\begin{bmatrix}\mathbf{z}_{11} & \mathbf{z}_{12}\\ \mathbf{z}_{21} & \mathbf{z}_{22}\\ \mathbf{z}_{31} & \mathbf{z}_{32}\end{bmatrix}=\begin{bmatrix}\mathbf{.1} & \mathbf{.2}\\ \mathbf{.3} & \mathbf{.4}\\ \mathbf{.5} & \mathbf{.6}\end{bmatrix} \quad \textbf{(4.23)}$$

Dans une première étape, on transmet l'information dans le sens direct. On commence par calculer l'activation des neurones de la couche cachée notée **b**:

$$\mathbf{b}=\mathbf{W}.\mathbf{x}=\begin{bmatrix}\mathbf{0.5+2.0.3+0.3}\\ \mathbf{0.3+0.4+0.3}\end{bmatrix}=\begin{bmatrix}\mathbf{1.4}\\ \mathbf{1.0}\end{bmatrix} \quad \textbf{(4.24)}$$

Cette activation est ensuite convertie en réponse. En utilisant la fonction logistique on obtienne:

$$\mathrm{h}=f(\mathbf{b})=\frac{\mathbf{1}}{\mathbf{1+e^{-b}}}=\begin{bmatrix}\mathbf{0.8022}\\ \mathbf{0.7311}\end{bmatrix} \quad \textbf{(4.25)}$$

Cette activation est ensuite transmise aux neurones de la couche de sortie. Elles calculent leur activation:

$$\mathbf{a} = \mathbf{Z} \cdot \mathbf{h} = \begin{bmatrix} \mathbf{0.1 \cdot 0.8022 + 0.2 \cdot 0.7311} \\ \mathbf{0.3 \cdot 0.8022 + 0.4 \cdot 0.7311} \\ \mathbf{0.5 \cdot 0.8022 + 0.6 \cdot 0.7311} \end{bmatrix} = \begin{bmatrix} \mathbf{0.2264} \\ \mathbf{0.5331} \\ \mathbf{0.8397} \end{bmatrix} \tag{4.26}$$

Les neurones de sortie transforment leur activation en réponse en utilisant la fonction logistique:

$$\mathbf{o} = \mathbf{f(a)} = \begin{bmatrix} \mathbf{0.5564} \\ \mathbf{0.6302} \\ \mathbf{0.6984} \end{bmatrix} \tag{4.27}$$

La première étape qui fournit la réponse est maintenant terminée. L'apprentissage peut commencer. Tout d'abord, un signal d'erreur est calculé en comparant la réponse donnée **o** avec la réponse attendue **d**:

$$\mathbf{e} = \mathbf{d} - \mathbf{o} = \begin{bmatrix} \mathbf{-0.4564} \\ \mathbf{-0.3302} \\ \mathbf{0.0016} \end{bmatrix} \tag{4.28}$$

Pour calculer le signal d'erreur de la sortie δ_{sortie} il faut évaluer la dérivée du signal de sortie:

$$\mathbf{f©(a)} = \mathbf{o} * \mathbf{(1-o)} = \begin{bmatrix} \mathbf{0.5564} \\ \mathbf{0.6302} \\ \mathbf{0.6984} \end{bmatrix} * \begin{bmatrix} \mathbf{0.4436} \\ \mathbf{0.3698} \\ \mathbf{0.3016} \end{bmatrix} = \begin{bmatrix} \mathbf{0.2468} \\ \mathbf{0.2330} \\ \mathbf{0.2106} \end{bmatrix} \tag{4.29}$$

Les neurones de sortie peuvent maintenant évaluer leur signal d'erreur:

$$\boldsymbol{\delta}_{\mathbf{sortie}} = \mathbf{f©(a)} \otimes \mathbf{e} = \mathbf{o} \otimes \mathbf{(1-o)} \otimes \mathbf{(t-0)} = \begin{bmatrix} \mathbf{-0.1126} \\ \mathbf{-0.0770} \\ \mathbf{0.0003} \end{bmatrix} \tag{4.30}$$

Le signal d'erreur est transmis en sens inverse vers les neurones de la couche cachée en utilisant les connexions qui relient la couche de sortie de la couche cachée. On calcule une matrice **R** (pour retour), dont chaque terme $r_{j,l}$ est obtenu en multipliant le signal d'erreur du neurone de sortie j par l'intensité de la connexion qui connecte le neurone j

avec le neurone l de la couche cachée: $\mathbf{r}_{j,l} = \delta_j^{\text{sortie}} \times \mathbf{z}_{j,l}$ (avec δ_j^{sortie} est dénotée la j-ème composante du vecteur δ_{sortie}). En notation matricielle la relation devienne :

$$\mathbf{R} = \mathbf{Z} \otimes \left(\mathbf{1}^{\mathrm{T}} \otimes \delta_{\text{sortie}}\right) = \mathbf{Z} \otimes \left[\delta_{\text{sortie}} \delta_{\text{sortie}}\right]$$

$$= \begin{bmatrix} .1 & .2 \\ .3 & .4 \\ .5 & .6 \end{bmatrix} \otimes \begin{bmatrix} -0.1126 & -0.1126 \\ -0.0770 & -0.0770 \\ 0.0003 & 0.0003 \end{bmatrix} = \begin{bmatrix} -0.0113 & -0.0225 \\ -0.0231 & -0.0308 \\ 0.0002 & 0.0002 \end{bmatrix} \qquad \textbf{(4.31)}$$

où 1^{T} est un vecteur ligne rempli de valeurs 1.

Une fois l'erreur propagée vers la couche cachée, les neurones de la couche de sortie peuvent corriger leurs synapses. Elles calculent donc $\Delta\mathbf{Z}$ et corrigent $\mathbf{Z}$ qui devient $\mathbf{Z}_{[t+1]}$ (avec η=1 pour simplicité):

$$\mathbf{Z}_{[t+1]} = \mathbf{Z} + \Delta\mathbf{Z} = \mathbf{Z} + \eta \delta_{\text{sortie}} \mathbf{h}^{\mathrm{T}}$$

où:

$$\eta \delta_{\text{sortie}} \mathbf{h}^{\mathrm{T}} = \begin{bmatrix} -0.1126 \\ -0.0770 \\ 0.0003 \end{bmatrix} \begin{bmatrix} 0.8022 & 0.7311 \end{bmatrix} = \begin{bmatrix} -0.0904 & -0.0823 \\ -0.0617 & -0.0563 \\ 0.0003 & 0.0002 \end{bmatrix} \qquad \textbf{(4.32)}$$

résulte que:

$$\mathbf{Z}_{[t+1]} = \begin{bmatrix} .1 & .2 \\ .3 & .4 \\ .5 & .6 \end{bmatrix} + \begin{bmatrix} -0.0904 & -0.0823 \\ -0.0617 & -0.0563 \\ 0.0003 & 0.0002 \end{bmatrix} = \begin{bmatrix} 0.0096 & 0.1177 \\ 0.2383 & 0.3437 \\ 0.5003 & 0.6002 \end{bmatrix} \qquad \textbf{(4.33)}$$

Une autre quantité intermédiaire correspond à l'estimation d'une erreur f dans les neurones de la couche cachée. Elle est obtenue en faisant la somme des signaux d'erreur des neurones de la sortie propagés en retour par les connexions:

$$\mathbf{f} = \mathbf{R}^{\mathrm{T}} \mathbf{1} = \mathbf{Z}^{\mathrm{T}} \delta_{\text{sortie}} = \begin{bmatrix} -0.0342 \\ -0.0531 \end{bmatrix} \qquad \textbf{(4.34)}$$

Le signal d'erreur $\delta_{\text{cachée}}$ pour les neurones de la couche cachée se calcule d'une manière similaire à celui de la couche de sortie:

$$\delta_{cachee} = f\copyright(b) \otimes f = h \otimes (1 \otimes h) \otimes (Z^T \delta_{sortie}) = \begin{bmatrix} -0.0054 \\ -0.0104 \end{bmatrix} \quad (4.35)$$

Les cellules de la couche cachée peuvent maintenant corriger **W** en $\mathbf{W}_{[t+1]}$:

$$\mathbf{W}_{[t+1]} = \mathbf{W} + \Delta\mathbf{W} = \mathbf{W} + \eta\delta_{cachée}\mathbf{x}^T = \begin{bmatrix} .5 & .3 & .1 \\ .3 & .2 & .1 \end{bmatrix} + \left(\begin{bmatrix} -0.0054 \\ -0.0104 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \right)$$

$$= \begin{bmatrix} .5 & .3 & .1 \\ .3 & .2 & .1 \end{bmatrix} + \begin{bmatrix} -0.0054 & -0.0108 & -0.0163 \\ -0.0104 & -0.0209 & -0.0313 \end{bmatrix} = \begin{bmatrix} .4946 & .2892 & .0837 \\ .2896 & .1791 & .0687 \end{bmatrix} \quad (4.36)$$

L'étapes du traînement se succèdent jusque le moment que le critère d'arrêt est satisfait. L'information se propage en sens directe, l'erreur en sens inverse et les poids sont modifiés pour minimiser l'erreur. (Pour les critères d'arrêt du traînement voyez le paragraphe suivant).

CARACTERISTIQUES DES RESEAUX MULTICOUCHES

- **Approximateur universel (classifieur universel)**
- **Rapidité d'exécution (++ si multi-processeurs ou chip)**
- **Robustesse des solutions, résistance au bruit des données**
- **Facilité de développement**
- **Le choix de l'architecture est critique**
- **Le temps d'apprentissage peut être long**
- **Présence de minima locaux de la fonction de coût**

4.3 Des aspects importants pour l'algorithme de rétro-propagation de l'erreur

La technique de rétro-propagation de l'erreur ne peut pas s'appliquer que pour des problèmes de classification dont on connaît les bonnes réponses.

Il y a des **nombreuses variantes** qui poursuivent spécialement **l'élimination de ses deux désavantages majors** :

- **la vitesse de convergence réduite ;**
- **le blocage dans des minimums locales de la fonction d'erreur, qui ne sont pas des solutions optimales ;**

À la suite on traite quelques facteurs qui influencent la performance de l'algorithme. On présente les causes des possibles résultats non satisfaisants et on offre des indications utiles en pratique.

4.3.1 La création de la base de données comprend :

- le rassemblement des données;
- le choix des variables ;
- l'analyse des données;
- le traitement des données ainsi qu'il peut être apprises avec efficience ;

Il y a des expertes qui affirme que 9/10 du temps du développement d'un réseau de neurones, comme solution d'un problème, est affecté pour trouver et assembler les dates adéquates.

Choisir les dates d'entrée comporte souvent le choix entre nombreux variables, les plus signifiantes. Par exemple, un procès industriel permet la surveillance de la pression, de la température, du flux technologique en quelques cent points. On doit sélectionner entre ces variables les plus importantes**. L'expérience dans le domaine d'application est très importante**. Par exemple, c'est une nécessité de collaborer avec des experts dans le système bancaire pour réaliser un réseau qui fonctionne dans ce domaine ci. Après choisir les plus importantes variables, on pose la question de leur distribution usuelle. On élimine les valeurs dehors celles typiques.

L'analyse des données d'entrée doit être faite de la perpective des techniques statistiques :

-la valeur de la fonction de corrélation entre une certaine entrée et une certaine sortie peut suggérer l'inclusion ou l'exclusion de la variabl

- une corrélation prononcée entre deux variables peut conduire à l'élimination d'une des ces deux.

L'analyse des données peut conduire à l'identification des tendances, des cycles ou des autres relations qui peut être extraites par un pré traitement.

Le pré traitement des données implique le calcul des sommes, des inverses, des différences, des dérivées, des exponentiels, des radicales, des moyennes, des transformées de Fourrier ou extractions des caractéristiques. Un réseau peut préparer les données d'entrée pour un autre, par exemple les grouper en avant de les classifier.

La quantité suffisante des données est affecté de considérations pratique, comme le prix de rassemblement des dates. Les données doivent assurer un échantillon représentatif mais, en même temps suffisant pour un traînement correct.

Des grands sets de données réduisent le risque de sous-échantillonnage de la fonction à représenter. De plusieurs données, de meilleure est l'estimation de la fonction, mais l'apprentissage peut durer trop long temps.

Des sets réduits de données conduisent au traînement rapide, mais le réseau peut échouer dans l'application finale.

Il y a une règle empirique qui dit que pour un apprentissage correct sont nécessaires approximativement dix models de traînement pour un poids.

Pratiquement, la quantité "suffisante" de dates dépend de:

-la dimension du réseau ;

-les conditions de test ;
-la distribution des dates d'entrée et de la sortie.

4.3.2 Les composantes de la base de données

L'ensemble des valeurs connues de la fonction à approximer est appelée une **tâche.** Elle a des couples **vecteur d'entrée - vecteur de sortie désiré**.

a. L'ensemble d'apprentissage

b. L'ensemble de validation

Cet ensemble sert à mesurer l'ajustement du modèle en cours d'apprentissage. La taille de l'ensemble de validation correspond, en général, à un tiers du nombre de données de l'ensemble d'apprentissage.

c. L'ensemble de test

Pour pouvoir juger si le réseau a bien codé les caractéristiques de la fonction sous-jacente, il est nécessaire de mesurer les performances du réseau. Ces performances sont appréciées, en présentant au réseau des valeurs inconnues. Ce sont des associations qui n'ont pas été utilisées pour entraîner le réseau.

d. Découpage de la tâche

Le découpage judicieux de la tâche est décisive pour obtenir des performances satisfaisantes. En général **l'ensemble d'apprentissage et de validation représente 90% de la tâche** et **l'ensemble de test 10%.** Il y a aussi des résultats théorétiques qui justifie ce découpage. Les performances du réseau sont essentiellement dépendantes du partage de la base de dates.

On recommande que le nombre d'associations utilisé soit dépendent du nombre de poids et de l'erreur carrée finale :

$$\mathbf{B = \frac{W}{\varepsilon}} \qquad \textbf{(4.37)}$$

où W est nombre de poids et ε l'erreur carrée finale.

4.3.3 L'apprentissage

L'apprentissage consiste à ajuster les paramètres du réseau de façon à obtenir la meilleure approximation possible de la fonction désirée.

À la fin de l'apprentissage, il faut tester les capacités de prédiction du réseau sur l'ensemble d'associations inconnues des données de test. Ceci permet d'évaluer la capacité de généralisation du réseau. Le réseau de neurones généralise bien si l'erreur sur les données inconnues est bas. Au contraire, un réseau généralise mal lorsqu'il obtient de bons résultats sur les données de traînement, mais il présente une erreur de prédiction élevée sur les données inconnues de l'ensemble de test.

4.3.3.1 Les types d'apprentissage

Toutes les associations de l'ensemble d'apprentissage sont disponibles pour le réseau dès le départ. Ce type d'apprentissage est appelé **Off-line**

L'apprentissage **On-line** (en temps réel) est plus proche de conditions réelles d'utilisation du réseau. Ce procédé présente un défaut. Si la mise à jour des paramètres du réseau est trop radicale suite à la présentation d'une nouvelle association, alors la fonction de sortie du réseau risque de perdre ses capacités interpolatrices par rapport aux autres données. De même, si la nouvelle association est proche de la précédente, alors le changement des paramètres sera trop faible.

Pour l'algorithme de la rétro-propagation de l'erreur il y a deux modalités de traînement off-line :

- **Modèle après modèle** (en anglais "pattern mode") Les poids changent après chaque présentation d'un couple modèle d'entrée – modèle de sortie. Les résultats empiriques endiguent que la convergence d'algorithme est plus rapide en ce cas ci. On recommande que les couples sont sélectionnés aléatoirement de l'ensemble d'apprentissage. **Une présentation complète de l'ensemble d'apprentissage est nommée "époque''**. Pour traîner un réseau sont nécessaires plusieurs époques. On recommande de changer l'ordre de modèles de l'époque à l'époque pour éviter que le réseau apprenne une périodicité fausse des données.
- **Après tous les modèles de l'ensemble d'apprentissage** (en anglais "batch mode") Les poids sont changés après l'application de l'entier ensemble d'apprentissage.

La meilleure méthode d'apprentissage est dépendante d'application. Les mieux résultats sont obtenus quand on utilise des données couvertes de bruit. En ce cas, la capacité de généralisation est meilleure.

4.3.3.2 Critères d'arrêt de l'apprentissage

Il y a plusieurs critères pour arrêter l'apprentissage :

- après un nombre prédéfini d'itérations (mais parfois l'erreur ne descend suffisamment);
- à une erreur établie (le nombre d'itérations pourrait être trop grand);
- à une variation minimum de la valeur d'erreur entre deux itérations consécutives (ce mode n'est pas utile quand l'erreur a une valeur encore élevée, mais des palières où les variations sont insignifiantes);
- "validation croisée" Les données se partagent en deux ensembles, l'ensemble d'apprentissage dont la performance du réseau s'améliore en cours du traînement et l'ensemble de test dont la performance du réseau s'améliore en cours d'entraînement jusqu'à un point, après ce moment l'erreur est de plus en plus élevée. Cette intervalle du temps est nommée période de "supra-concordance" (en anglais "over fitting"). Le réseau n'apprend pas le processus qui a généré les données, mais même les données. L'apprentissage s'arrête au moment du commencement de la "supra-concordance".

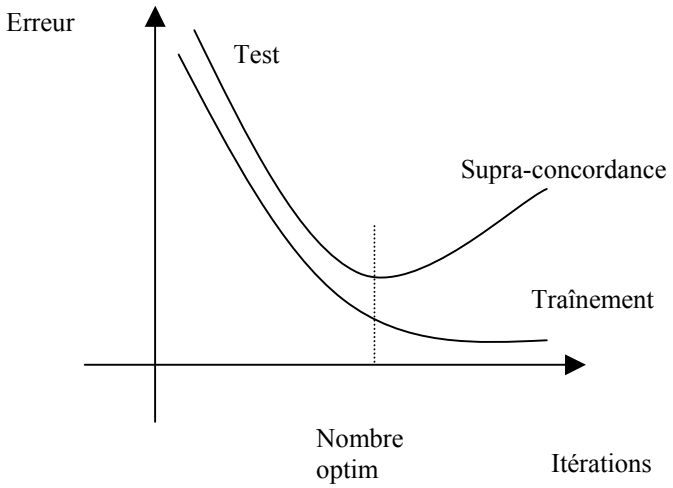


Fig.4.2 L'erreur d'apprentissage en fonction de nombre des itérations

Suivant l'étape du traînement on pose la question **de la généralisation.** Le terme généraliser signifie extraire des principes ou des conclusions par expérience. L'habilité des réseaux neuraux d'identifier des règles les donne la capacité de faire des prédictions sur des donnés nouvelles. La généralisation est vérifiée sur l'ensemble de test.
Les facteurs qui influencent la généralisation sont:

- le nombre d'échantillons de données;
- la complexité du problème;
- la dimension du réseau;

La mesure Vapnik-Chervonenkis donne une limite inférieure et une limite supérieure du nombre de données, pour une généralisation correcte en fonction de la dimension du réseau et du nombre d'échantillons de traînement .

4.3.4 La fonction d'activation

Usuellement la fonction d'activation est une fonction sigmoïde bipolaire ou unipolaire. Il y a des résultats théorétiques qui indiquent une vitesse plus grand de convergence dans le cas d'une fonction bipolaire. L'ajustage du paramètre β (sa décroissance en temps avec le gradient de l'erreur) dans la relation (4.2) est bénéfique en spécial dans la première phase d'apprentissage.

Un avantage majeur de la fonction sigmoïde est la relation simple avec sa dérivée (relation 4.3) qui conduise à une évaluation rapide des erreurs.

L'observation que les variations des poids sont négligeables pour les signaux grands, pour lesquels la dérivée s'annule, a déterminé la croissance artificielle de la dérivée. Par l'addition d'une constante de 0.1 à la dérivée, on a réduit le temps d'apprentissage à demie.

Dans les applications de classifications est utile la fonction "softmax", qui permet l'interprétation des sorties comme probabilités, donnée par la relation suivante :

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \qquad (4.38)$$

4.3.5 La vitesse d'apprentissage

Le choix de la constante d'apprentissage, dans l'intervalle (0,1) est très importante pour l'évolution de l'apprentissage du réseau :

- une η grande assure une convergence rapide de l'algorithme, mais elle peut déterminer des oscillations du réseau;
- une constante réduite (0.05,0.25) a comme effet la croissance du temps de convergence et peut conduire plus souvent à blocage dans un minimum local. Donc c'est justifié d'essayer d'élever la constante d'apprentissage et grandir la vitesse de convergence.

  Il y a plusieurs méthodes pour satisfaire cette condition :

4.3.5.1 La méthode de l'impulsion

La méthode consiste en introduire une relation entre le changement d'un poids courante et celui précédente. Ainsi on peut réaliser des larges pas d'itération, qui assure une convergence rapide, sans utiliser une grande constante d'apprentissage :

$$\Delta_p w_{ij}(t+1) = \eta.\delta_{pj}.o_{pi} + \alpha.\Delta_p w_{ij}(t) \qquad (4.39)$$

où α est une constante, baptisée "implusion", qui détermine l'effet du poids antérieur sur celui momentané. Cette relation pratiquement filtre les grandes variations des erreurs dans l'espace des poids des connexions. Elle évite les oscillations du réseau aux grandes valeurs de la vitesse d'apprentissage.

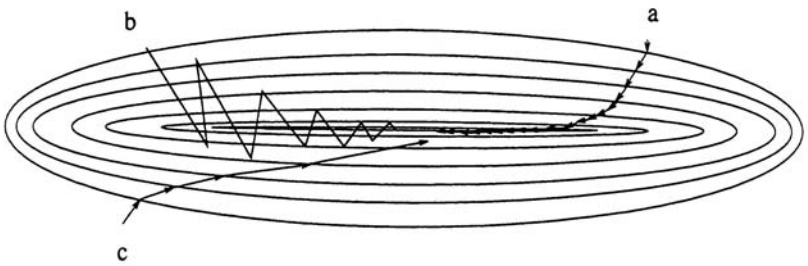


Fig.4.3 La convergence dans l'espace de poids
a) avec une constante d'apprentissage faible ; b) avec une constante d'apprentissage élevée ; c) avec une constante d'apprentissage élevée et un terme d'impulsion

La figure 4.3 présente la convergence d'un réseau dans l'espace de l'erreur en trois cas possible :

a) η réduite, sans momentum, quand il est nécessaire un temps long pour attendre le minimum ;

b) η grande, sans momentum α, quand le minimum n'est pas touché à cause des oscillations ;

c) η grande , avec impulsion α, quand le minimum est touché rapidement;

4.3.5.2 La méthode du "lissage "

C'est une généralisation de la méthode de l'impulsion, introduise par Sejnowski et Rosenberg (1987)

$$\Delta_p w_{ij}(t+1) = \eta.[b.\Delta_p w_{ij}(t) + (1-b).\delta_{pj}.o_{pi}] \qquad (4.40)$$

Si :

- b=0 alors on peut reconnaître la forme standard du RPE ;
- b=1, le poids reste celui ci du moment antérieur ;
- b$\in$ (0,1) alors la variation du poids est lissée avec b ;

4.3.5.3 La constante d'apprentissage variable

Pour éviter les oscillations du réseau, qui peut apparaître autour de l'optimum, on choisit une constante descendante en temps. Pour assurer une convergence rapide de l'algorithme cette descendance doit être rapide. La condition de la décroissance rapide du η est satisfaite quand :

$$\lim_{k\to\infty} \sum_{k=1} (\eta[k])^2 < \infty \qquad (4.41)$$

où $\eta[k]$ est la constante d'apprentissage au moment k

Si la constante d'apprentissage descende trop rapide il y a le risque que les poids oublient les models déjà appris. Ainsi que la constante d'apprentissage doit décroiser suffisamment de lent. Cette condition de la décroissance lente du η est satisfaite quand :

$$\lim_{k\to\infty} \sum_{k=1} (\eta[k])^2 = \infty \qquad (4.42)$$

Si $\eta[k]=1/k$ les conditions (4.41) et (4.42) sont simultanément satisfaites.

Les deux conditions représentent une formulation mathématique du dilemme plasticité stabilité découverte par Grossberg.

Un réseau de neurones doit satisfaire deux conditions contradictoires :

1. il doit être suffisamment stable pour se rappeler les modèles antérieurement appris ;

2. il doit être suffisamment plastique pour apprendre des modèles neufs;

4.3.6 L'initialisation des poids et des seuils

On doit attentivement choisir les valeurs des poids et des seuils pour éviter la saturation prématurée de l'état des neurones, et en conséquence la descendance de la vitesse de convergence. C'est improbable que les neurones se saturent si le nombre d'entrées est petit. Usuellement on choit aléatoirement et éventuellement uniformes distributifs les valeurs des poids dans l'intervalle $(-2{,}4/F_i, 2{,}4/F_i)$ où F_i représente le nombre des entrées dans le neurone .

4.3.7 La fonction d'erreur

Dans une forme générale la fonction d'erreur s'écrit:

$$\mathbf{E = \sum_{p}\sum_{j=1}^{n}\left|o_j(X_p;W) - d_{jp}\right|^R} \qquad \mathbf{(4.43)}$$

où

- d_{jp} est la réponse désirée pour le modèle p d'entrée;
- o_j est la sortie;
- W la totalité des paramètres du réseau (poids et seuils);
- X_p le modèle d'entrée p;

Pour R=2 on obtient les moindres carrés.

Observations

- L'erreur des moindres carrés est extrêmement sensible à la présence des erreurs individuelles grandes. On obtient des performances supérieures pour des autres types de distances métriques avec R<1 (par exemple la distance Manhattan pour R=1).
- En observant la nature statistique de l'algorithme de la rétro-propagation de l'erreur, White proposait des techniques de la statistique robuste pour élever l'insensibilité du réseau aux perturbations. L'erreur se remplace par des fonctions d'erreur comme : $\mathbf{\tanh(e_i/2),\quad e_i/(1+e_i)^2,\quad max[-e_i, min(e_i, x_i)]}$. Pour choisir la fonction d'erreur on utilise la méthode empirique "essayer et rater" .
- La fonction d'erreur d'un réseau multicouche est une fonction nonlinéaire

parametrisée par les valeurs des interconnexions et des seuils. L'algorithme de traînement doit minimiser cette fonction. La représentation géométrique de la fonction d'erreur mette en évidence l'existence d'un minimum global et des plusieurs minimums locaux. La surface d'erreur dans le cas d'une fonction d'activation nonlinéaire n'est pas concave, ainsi qu'existe la possibilité que l'algorithme s'arrête dans des minimums locaux. C'est possible que dans un minimum local l'erreur soit encore élevée. Pour une certaine catégorie de problèmes le blocage dans des minimums locaux est irelevante, comme, par exemple l'implémentation de la fonction XOR. Aussi un minimum local peut donner une solution satisfaisante. L'élaboration d'un algorithme adéquat est basée sur des techniques d'optimisation nonlinéaires.

- Il y a plusieurs méthodes utilisées pour minimiser la fonction d'erreur dans un réseau multicouche :

1. Méthodes avec une évolution de l'erreur décroissante ou constante d'une itération à l'autre, qui ne permettent pas la croissance temporaire de l'erreur. Le désavantage de ces méthodes est l'impossibilité de sortir dehors de minimums locaux. Des exemples sont les algorithmes de type "paquet de données " , du "gradient conjugué" , "quasi Newton" .
2. Méthodes avec une évolution en moyenne vers un minimum, qui permettent des croissances temporaires de l'erreur. Dans cette catégorie sont les algorithmes de type gradient "modèle avec modèle" avec de l'impulsion.

Plusieurs variantes performantes font appel aux approximations locales carrées de la fonction de l'erreur et utilisent la matrices Jacobienne des dérivées partielles de premier ordre aussi que les matrices Hessienne des dérivées partielles de deuxième ordre : l'algorithme du "gradient conjugué" et l'algorithme Levenberg Marquardt.

4.3.8. Le problème des minimums locaux

Il y a plusieurs méthodes pour éviter les minimums locaux :

- changer la constante d'apprentissage ;
- changer les poids initiaux ;
- changer le nombre des neurones cachés ;
- par addition des petites valeurs aléatoires aux poids ;

Si le nouvel état est suffisamment éloigné du minimum local l'apprentissage peut évoluer dans une nouvelle direction sans y revenir. Si l'erreur est satisfaisante l'arrêt du traitement dans un minimum local n'a plus d'importance, parce qu'il est une solution acceptable.

Pour les réseaux multicouches les algorithmes génétiques évitent les minimums locaux a prix d'une capacité de mémoire et un temps de calcul très grand. La méthode "simulated annealing" permit aussi éviter les minimums locaux aussi que sortir de minimums.

4.3.9 Choix d'une architecture

Le réseau est dit "à architecture fixe" si le nombre de neurones qu'il contient est fixe et ne varie pas au cours de l'apprentissage. Dans ce cas, le nombre de neurones dans les couches intermédiaires doit être choisi judicieusement afin que le réseau ne soit pas surdimensionné ou sousdimensionné. Un réseau petit n'est pas capable de fournir une bonne sortie. Il ne peut pas apprendre l'information entière contenue dans les données d'entrée. D'autre partie, c'est difficile de manipuler un réseau trop large. Le réseau de neurones pourrait être "trop capable". Il peut aussi générer du bruit et il ne peut pas construire une représentation compacte de la relation entre les modèles d'entrée. Pour obtenir l'architecture optimale pour une tâche donnée, plusieurs architectures fixes pourraient être testées.

Certains réseaux, dits à l'architecture variable, ont la capacité d'adapter le nombre de neurones de leur couche intermédiaire pendant l'apprentissage. Ces algorithmes sont coûteux en temps. Il y a plusieurs procédures pour créer un tel réseau:

- on commence avec un neurone et au parcours on additionne des nouveaux neurones s'ils sont nécessaires. On teste la performance du réseau et si elle est insuffisante alors il grandit par additioner des nouveaux neurones;
- on commence avec un grand réseau et puis on efface des neurones inutiles, en testant la performance du réseau;

Pour choisir le nombre des couches cachées et des neurones sur une couche il n y a pas des critères ou des règles quantitatives, seulement le bon sens. Toujours on essaye, pour trouver la plus meilleure structure.

Soit n la dimensionne du modèle d'entrée. Le nombre des neurones de la première couche cachée doit être plus grand que un quand il y a plusieurs régions de décisions convexes ou avec des creux. On apprécie que le nombre des neurones cachés de la première couche doive être trois fois plus grand que le nombre des neurones de la deuxième couche cachée pour fournir trois ou plusieurs côtés pour chaque région convexe de la deuxième couche. Si dans la première couche cachée il y a n_1 neurones, les régions de décision de la deuxième couche cachée peuvent avoir maximum n_1 côtés, parce que chaque région de décision est formée par l'intersection des semiplanes formés par la première couche. Si à la sortie sont nécessaires m classes on a besoin d'un même nombre de neurones. En général la structure d'un réseau multicouches doit être de type compression.

4.3.10 Trouver le nombre et les valeurs optimales des poids

On a démontré que trouver les poids optimaux pour un réseau de dimension fixe, pour n'importe quel ensemble de traînement est un problème combinatoire (de type np complet).

Il y a plusieurs méthodes pour réduire le nombre des connexions, en maintenant les possibilités du réseau inaltérées:

- éliminer les poids jusqu'au moment quand la performance du réseau est affectée;

- utiliser des connexions locales. Les neurones reçoivent entrées seulement d'une région locale des entrées possibles ; Comme exemple dans la reconnaissance des caractères d'une dimensionne de nxn, les neurones cachés traitent seulement une fractionne de mxm (n>m) des modèles d'entrée. C'est possible que plusieurs neurones soient utilisés pour couvrir la même région et que les régions adjacentes se superposent.

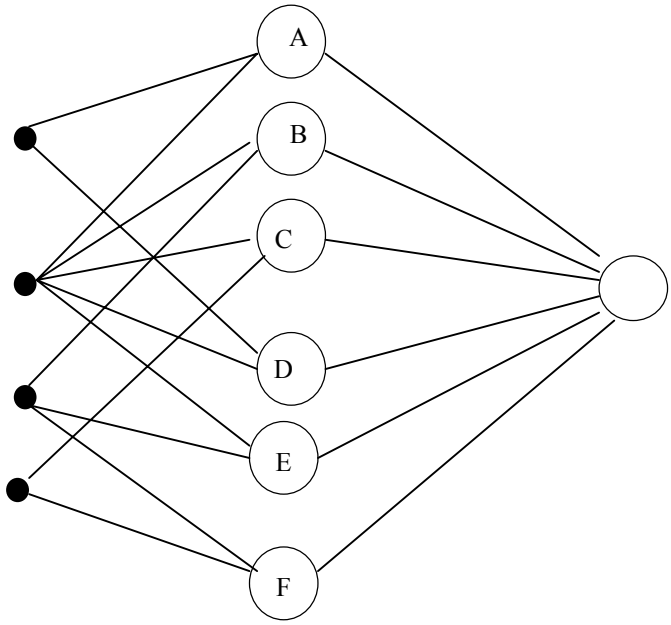


Fig.4.4 Un exemple pour des poids locales

La Fig.4.4 présente un réseau obtenu par partage. Chaque neurone caché est connecté dans une région locale de l'entrée. Les régions adjacentes se superposent avec une position. Les neurones A, B, C et D, E, F couvrent les mêmes régions. En comparaison avec le réseau complet connecté qui aurait 37 connexions celui ci a seulement 13 connexions parce que les neurones A,B,C ont les mêmes poids que D,E,F.

4.4 Le problème d'égalisation

Les voies de télécommunication ont une réponse en fréquence variable comme niveau et non linéaire comme phase, ainsi que la transmission des données est affectée par l'interférence intersymboles. Pour compenser la caractéristique non idéale de la voie de communication on utilise un circuit, nommé égalisateur. Par l'égalisation on élimine les perturbations introduites par la voie de communication, comme par exemple: le bruit, les distorsions non linéaires, la variance en temps des caractéristiques de la voie, l'interférence intersymboles et l'interférence avec les voies adjacents. Parce qu'en général les voies de communications sont variantes en temps les égalisateurs doivent être adaptatifs, pour suivre la variation en temps de la réponse en fréquence de la voie

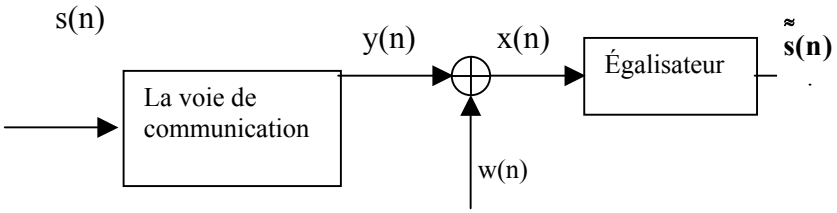


Fig. 4.5 La position d'égalisateur dans la transmission du signal

Fig. 4.5 présente la position relative d'un égalisateur dans un système de transmission des données. La sortie de la voie de communication s(n) est affectée par un bruit additif w(n), dans le plus simple cas, modelé comme un procès gaussien additif.

On doit considérer les facteurs suivants pour développer un égalisateur :

- l'interférence intersymboles;
- le bruit introduit par le récepteur;
- le type de la voie ;

Il y a des voies fixes et variants en temps. Des distorsions linéaires et non linéaires peuvent affecter une voie de communication. L'interférence intersymboles est spécialement responsable pour les distorsions linéaires. Les distorsions non linéaires sont introduites par les amplificateurs, convertisseurs, par l'environnement de la propagation.. Si la voie est linéaire alors la sortie y est donnée par la convolution entre s et h, c'est à dire $y = s \otimes h$. Si la voie est non linéaire, alors c'est plus compliqué.

La phase de la voie peut être minimum ou non. Une voie a une phase minimum si les zéros de la fonction de transfert H(z) sont dans l'intérieure du cercle unité dans le plan z. En absence du bruit, dans le cas de la voie avec une phase minimum, l'égalisateur travaille comme un système inverse de la voie pour transmettre sans distorsions. Quand le bruit est présent et la voie n'a pas une phase minimum, alors l'utilisation du modèle inverse n'est pas suffisante [67].

L'égalisateur doit réaliser le meilleur compromis entre l'élimination d'interférence intersymboles et l'amplification du bruit [82].

Dans une forme conventionnelle l'égalisateur est construit sur la base de la théorie du filtre linéaire dont les poids sont adaptés avec un algorithme qui minimise l'erreur entre le signal de la sortie et le signal reçu. Usuellement on utilise l'algorithme des moindres carrés. Ces égalisateurs n'ont pas des bonnes performances dans le cas des voies avec des nulles spectrales profondes parce qu'ils ont un gain élevé à ces fréquences, et ils amplifient le bruit additif. L'amplification du bruit est évitée par l'utilisation des techniques non linéaires. Un model classique d'égalisateur non linéaire est celui avec une décision sur la base de la réaction inverse. Un autre exemple est l'égalisateur qui estime la séquence d'entrée sur la base de la ressemblance maximum. À la même complexité qu'un égalisateur linéaire, un égalisateur avec la décision sur la base de la réaction inverse a des performances significativement améliorées quand l'interférence intersymboles est grande. Si de plus, la structure d'égalisateur avec une décision sur la base de la réaction inverse inclue un réseau de neurones les performances sont encore améliorées [196] .

La différence essentielle entre l'égalisateur linéaire et celui non linéaire est que le dernier utilise le signal reconstruit de la sortie pour s'adapter.

4.4.1 Egalisation comme un problème de classification

Comme une alternative à l'idée du filtrage inverse on peut aborder le problème d'égalisation comme un problème de classification des modèles d'entrée. Dans le cas d'une transmission bipolaire de données on doit faire une classification, comme +1 ou –1 entre les échantillonnes reçus affectés par l'interférence intersymboles et par du bruit. Dans ce cas la fonction d'égalisateur est d'associer au chaque échantillon reçu la région correcte de décision. Ainsi la fonction d'égalisateur devient équivalente au problème de classification. Dans les situations réelles, quand le bruit est présent dans le signal reçu, quand la voie de communication varie pendant temps et n'a pas une phase minimum, la classification optimale est non linéaire. Dans tous ces cas, quand les distorsions de la voie sont sévères, les égalisateurs linéaires ne répondent pas aux nécessités, ainsi qu'on fait appel aux égalisateurs non linéaires. Les RN sont capables de générer des courbes non linéaires et classifier des signaux dans l'espace multidimensionnel ainsi qu'ils sont des bons candidats pour l'égalisation.

4.4.2 Le modèle

Soit le signal s[n] appliqué a l'entrée d'une voie linéaire dispersif avec une réponse finie au impulse (FIR), comme dans la Fig.4.6. On modèle la voie dans ce cas avec un filtre FIR avec des valeurs réelles et différentes de zéros des coefficients a_0, a_1, ..., a_k. Le signal aléatoire d'entrée génère la sortie y[n] donnée par la relation :

$$y[n] = \sum_{i=0}^{k} a_i .s[n-i] \qquad (4.44)$$

Soit w[n] un bruit additif appliqué pour générer le signal observé x[n].Dans le plus simple cas ce bruit est modelé comme un bruit blanc. Par l'égalisation on reconstruit le signal d'entrée s[n] avec l'information représentée par le signal reçu x[n]. Un peut implémenter un tel circuit égalisateur avec un filtre transversal.

Soit M l'ordre du filtre, le vecteur poids $\mathbf{c=[c_0\, c_1\, c_2\, \ldots c_M]}$, d le retard et le signal réceptionné **x=[x[n] x[n-1]...x[n-M]].** Le problème d'égalisation est de déterminer les coefficients c_i ainsi que la fonction d'erreur soit minimisée:

$$J[n] = \frac{1}{2} E\left[d[n] - \sum_{K=0}^{M} c_k\, x[n-k] \right]^2 \qquad (4.45)$$

où E est l'espérance mathématique.

La solution optimal pour les coefficients du filtre transversal est:

$$\mathbf{c_{opt}[n] = R^{-1}[n]\xi[n]} \qquad (4.46)$$

où:

- R est la matrice symétrique de corrélation des entrées ;

$$\mathbf{R = E[x^T x]} \qquad (4.47)$$

- ξ est le vecteur de corrélation entre
  x=[x[n] x[n-1] ….x[n-M]] et **d=[d[n] d[n-1] ….d[n-M]]**

$$\xi = \mathbf{E[x^T d]} \qquad (4.48)$$

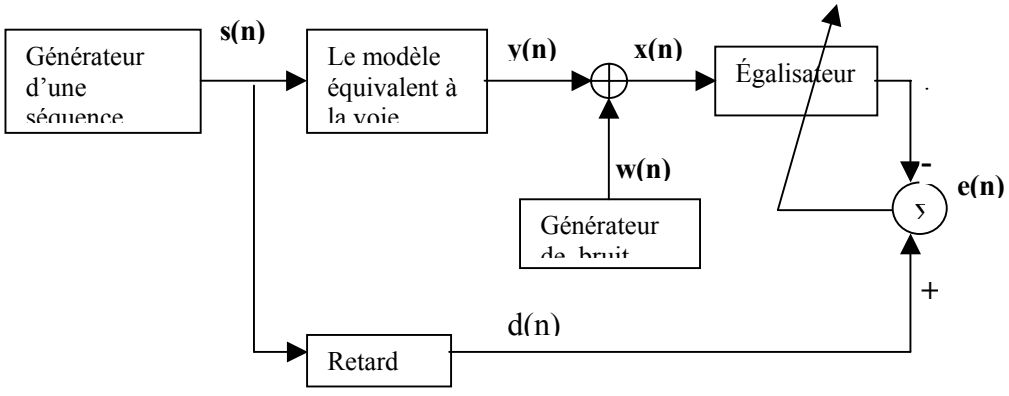


Fig.4.6 Le modèle du problème d'égalisation

On développe la relation (4.45) et on ignore les termes constants ainsi qu'on obtient :

$$J[n] = \frac{1}{2} c^T R[n] c - \xi^T[n].c \qquad (4.49)$$

Toutes les méthodes de type gradient sont adéquates pour trouver le minimum de cette fonction .

Pour modeler une voie non linéaire, variante , comme par exemple une voie avec des coefficients variants en temps on peut utiliser la fonction de transfert suivante:

$$H(z) = a_o(k) + a_1(k).z^{-1} + a_2(k).z^{-2} + ... + a_n(k).z^{-n} \qquad (4.50)$$

où les coefficients $a_i(k)$ varient avec le temps k .On peut générer ce type de coefficients par le passage d'un bruit gaussien a travers un filtre Butterworth [196]. La bande du filtre détermine la bande relative de la voie (c'est à dire la rate de la variation) . Dans le cas d'une voie avec une bande de 2 kHz, pour une vitesse de transmission de 2400 symboles per secondes et un filtre Butterworth d'ordre 2 avec une bande de 0.5 Hz au 3 dB, les coefficients obtenus sont représentés dans la Fig. 4.7

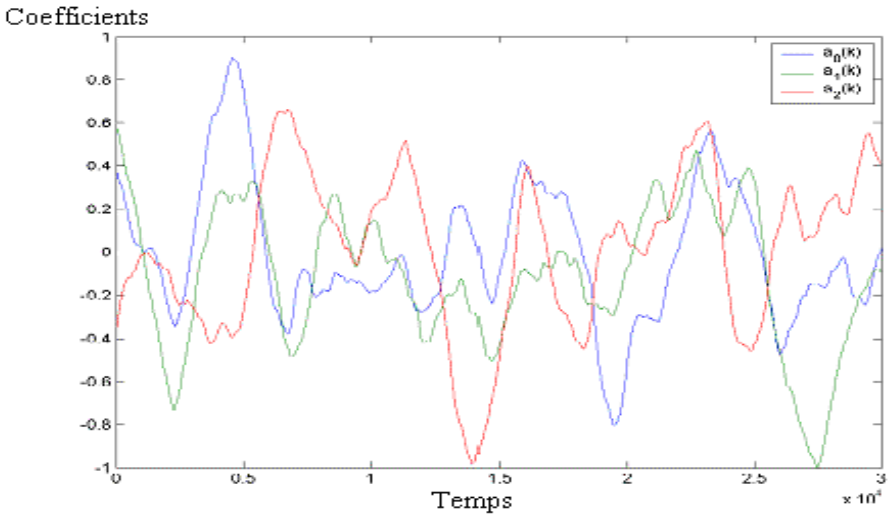


Fig.4.7 Les coefficients d'une voie variante en temps

4.4.3 Egalisateurs avec des réseaux de neurones

Toutes les caractéristiques des réseaux de neurones sont des arguments pour leur application dans l'égalisation des signaux, surtout : la capacité d'adaptation, la possibilité d'aborder des problèmes non linéaires, le parallélisme massive qui donne une vitesse exceptionnelle dans le traitement des signaux , l'implémentation matérielle efficiente.

Il y a plusieurs architectures de RN pour l'égalisation adaptative :

- RN multicouches traînés avec un algorithme compétitif de type gradient descendent, comme par exemple l'algorithme de la rétro-propagation de l'erreur [32]
- RN avec des liaisons fonctionnels [88]
- RN avec des fonctions radiales [33],[34]
- RN récurrents [133],[196]
- RN cellulaires [19],[134]
- Structures combinées , qui contiennent des égalisateurs traditionnels (linéaires ou non linéaires) et des RN [91].

Égalisateur avec un réseau de neurones multicouches

La structure d'un égalisateur de neurones implementé par un réseau avec une couche adaptative est représentée dans la Fig.4 .

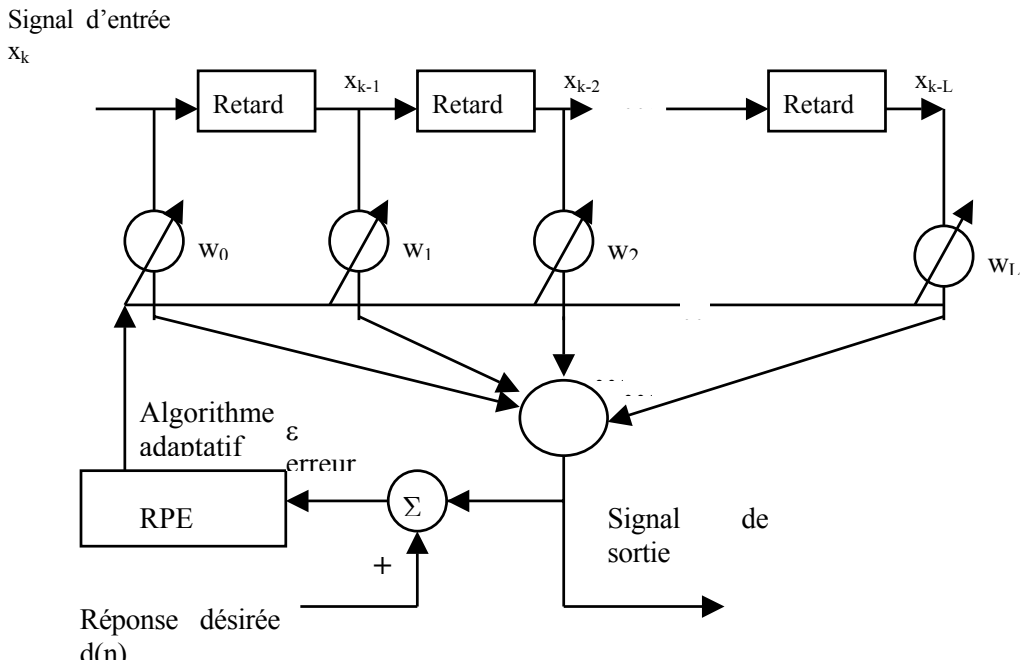


Fig.4.8 La structure d'un égalisateur de neurones implementée par un réseau avec une couche adaptative

Les entrées dans le réseau de neurones sont les échantillons retardés du signal d'entrée. L'information est passée de l'entrée à la sortie à travers les couches cachées, de couche en couche. C'est suffisant d'utiliser deux couches de neurones cachées , parce qu'on sait que ce RN peut implementer n'importe quel fonction de transformation, avec une précision désirée si le nombre des neurones est adéquatement choisi. La couche de la sortie a un seul neurone. On compare la sortie avec la réponse désirée qui est le signal retardée d(n). La fonction d'activation est une fonctionne sigmoïde, par exemple :

$$\mathbf{f(net_{pk}) = \frac{1}{1 + e^{-net_{pk}}}} \qquad \mathbf{(4.51)}$$

L'information est stockée dans les poids et seuils du RN, qui se mettent à jour périodiquement pour s'adapter aux caractéristiques de la voie de communication. Pour adapter l'égalisateur on utilise un algorithme du type gradient descendent, usuellement la rétro-propagation de l'erreur. L'algorithme de la rétro-propagation de l'erreur minimise les moindres carrés entre les signal d'entrée et la réponse désirée.

Les égalisateurs avec RN sont efficaces pour les voies et schèmes de communication réels (pour les signaux binaires, bipolaires, PAM) [112]. Des RN complexes sont nécessaires pour les canaux et schèmes complexes (par exemple pour les signaux QAM avec une modulation d'amplitude en quadrature, PSK avec une modulation en phase). Les extensions complexes sont obtenues directement à partir des schèmes réelles (les paramètres relevants sont remplacées par des valeurs complexes), comme dans la Fig.4.9.

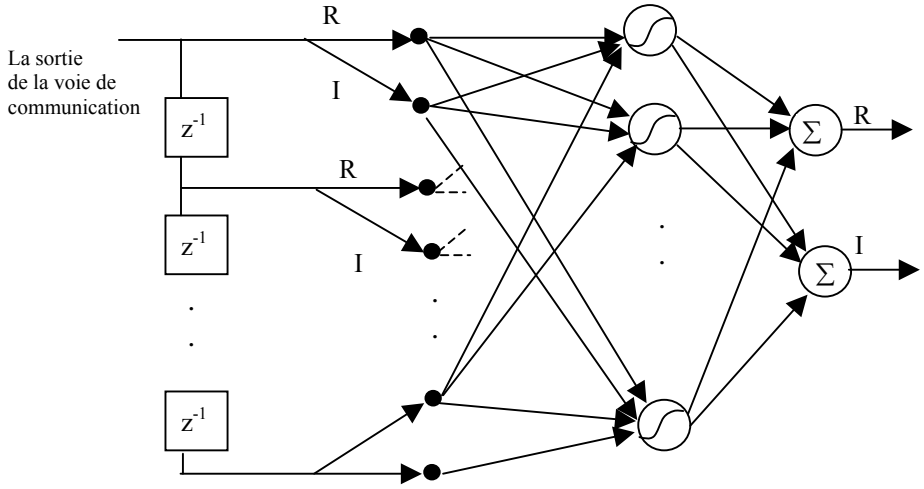


Fig.4.9 Un égalisateur avec une structure multicouche pour des signaux complexes

Dans tous les études rapportés dans la littérature de spécialité les performances obtenus par les égalisateurs avec des RN sont supérieures aux celles obtenues par les égalisateurs traditionnels , avec la même complexité. Surtout, les égalisateurs sont capables d'offrir des solutions dans tous les cas quand les égalisateurs traditionnels ont faillé, dans le cas des distorsions nonlinéaires et schèmes compliqués de modulation (par exemple modulation M-QAM, pour M>4). Dans la Fig. 4.10 est représentée l'erreur des symboles pour l'égalisation d'une voie mobile de satellite pour différentes structures [77] .

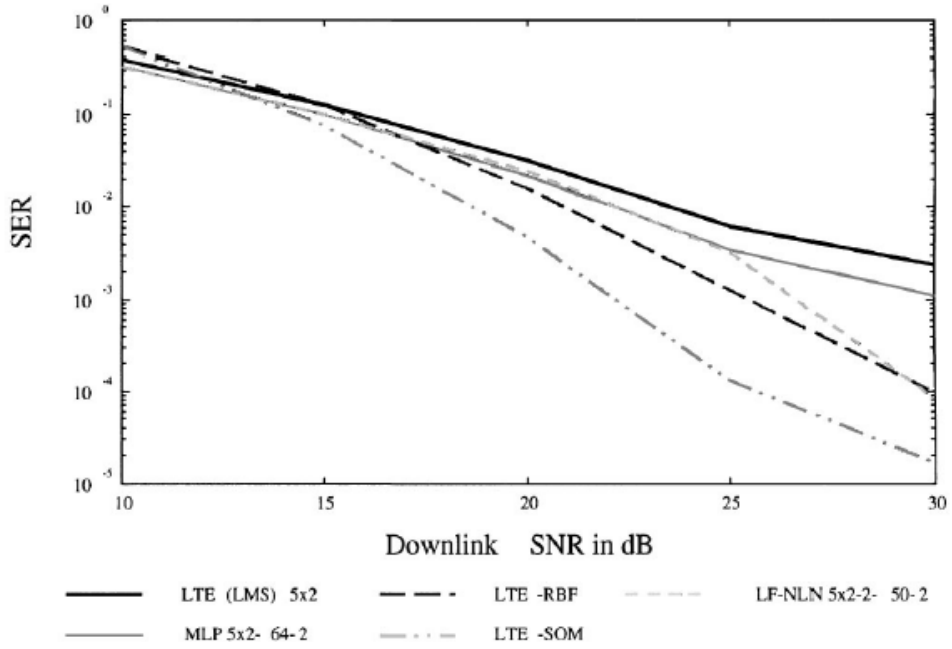


Fig. 4.10 L'erreur des symboles pour l'égalisateur d'une voie mobile de satellite

La performance d'égalisateur linéaire transversal d'ordre cinq (LTE) traîné avec l'algorithme des moindres carrés est représentée par une ligne grosse continue. La performance d'égalisateur de neurones multicouche (MLP) traîné avec l'algorithme de la rétro-propagation de l'erreur est représentée par une ligne fine continue. Le réseau de neurones a cinq entrées, soixante six neurones cachées et une sortie. Les autres égalisateurs ont une structure qui contienne un filtre transversal aussi qu'un réseau de neurones (de types différents). On peut observer que dans tous les cas les performances d'égalisateurs avec des RN ont été supérieures.