REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE
FACULTE DES MATHEMATIQUES

MEMOIRE
Présenté pour l'obtention du diplôme de MAGISTER
EN : MATHEMATIQUES

Spécialité : Recherche Opérationnelle : Méthodes Stochastiques

Par : MOUSSAOUI Fatma Zohra

THEME

PREVISION DES SERIES TEMPORELLES PAR LES RESEAUX DE NEURONES

Soutenu le 01/ 12 / 2008, devant le jury composé de :

Mr A. BERRACHEDI	Professeur	U.S.T.H.B.	Président
Mr O. ANES	Maître de Conférences	INPS	Directeur de thèse
Mme H. GUERBIENNE	Maître de Conférences	U.S.T.H.B.	Examinateur
Mr A.REBOUH	Maître de Conférences	U.S.T.H.B.	Invité
Mr M. CHOUIK	Maître de Conférences	ESC	Invité

Ma première pensée

Se dirige vers ma mère décédée, qui a tant fait pour moi, qui est une partie de moi et qui est à jamais dans mon cœur. A elle je dédie ce travail.

Je tiens en premier lieu à exprimer ma profonde gratitude et mes plus vifs remerciements à Monsieur Ouali ANES, mon directeur de thèse pour ses conseils avisés tout au long de ce travail, toute la confiance qu'il m'a témoignée, sa disponibilité, son aide et sa compréhension.

J'exprime aussi ma gratitude à Monsieur A. BERRACHEDI, Professeur à l'U.S.T.H.B pour m'avoir fait l'honneur de présider mon jury de thèse.

Je remercie vivement Madame H.GUERBIENNE Maître de Conférences à l'U.S.T.H.B ainsi que Messieurs A.REBOUH chargé de cours à l'U.S.T.H.B et M. CHOUIK, maître de conférences à l'Ecole Supérieure de Commerce qui ont bien voulu faire partie du jury.

Je remercie Melle Merzougui Mouna enseignante à l'université de Boumerdes pour son aide.

Mes derniers mots de remerciements vont tout naturellement à ma famille, en particulier mon père, mes frères et soeurs qui m'aiment et qui m'encouragent ainsi que tous les autres membres de ma famille comme mon beau frère Zoubir et mes neveux Amine et Tamara pour leur soutien.

Mes amies Fadhila et Samia méritent aussi mes remerciements pour leurs encouragements.

Mes sincères remerciements aussi à tous ceux qui ont contribué de prés ou de loin à l'aboutissement de ce travail.

TABLE DES MATIERES

1. INTRODUCTION

1.1. Généralités

Les dernières décennies ont connu le développement de modèles paramétrique linéaires connus par modèles autorégressifs à moyenne mobile (ARMA : Auto-Regressive-Moving-Average), introduits par Box et Jenkins (1976) et qui ont été très bien développés, étudiés et expérimentés. Malgré la puissance de la modélisation linéaire, elle s'avère insuffisante pour maîtriser certaines dynamiques pour lesquelles la relation entre la valeur à un instant donné et les valeurs passées de la série est de nature non linéaire. Ces phénomènes sont très nombreux et variés et ce sont eux qui apparaissent dans la pratique et dans différents domaines, tels que la médecine, la finance, l'économie. Nous nous sommes donc intéressés à des modèles faisant ressortir ce type de phénomènes : ce sont les réseaux de neurones.

Le neurone formel introduit par McCullogh et Pitts (1943) est un automate qui reproduit la composée de plusieurs fonctions très simples. Le perceptron a été introduit par Rosenblatt (1962) qui fut le premier à y associer un algorithme « d'apprentissage » et désigne un ensemble de neurones formels connectés. Les deux mathématiciens Minsky et Papert (1969) ont montré les limites théoriques du perceptron simple, ce qui a rendu le modèle inutile. Les chercheurs après avoir montré un manque d'intérêt pour les réseaux de neurones, se tournèrent vers l'approche symbolique. Dans les années 1980, deux équipes de chercheurs, l'une française [Lecun et al. (1989)] et l'autre américaine [Rumelhart et al. (1986)], ont mis au point l'algorithme de rétro propagation du gradient qui contribua au succès des réseaux de neurones revenus ainsi sur le devant de la scène avec de nouveaux algorithmes. Les chercheurs s'intéressèrent alors à ce thème dans des cadres généraux (approximation fonctionnelle, prédiction, processus de Markov, …) et à partir des années 80, les modèles neuronaux ont connu un grand succès à cause de leur capacité à résoudre certains problèmes par apprentissage. On a pu réaliser certaines applications basées sur les réseaux de neurones mais quoiqu'elles soient reconnues efficaces, nous devons être prudents dans la conclusion, parce que nous ne sommes pas encore surs de surpasser les méthodes classiques, même si nous obtenons rapidement des résultats cohérents. En effet, même si les réseaux de neurones nous ont éclairé sur l'utilisation de modèles non linéaires et qu'il est simple d'utiliser des principes connexionnistes, de concevoir des modèles complexes, nous ne sommes pas encore

capables de les maîtriser et les employer. Certains types de modèles de neurones peuvent être inappropriés pour modéliser des phénomènes où par exemple on veut résoudre un problème pour lequel nous n'avons que trop peu de données et dans ce cas, le modèle est inadéquat avec le phénomène réel .Ce problème a montré que pour avoir une bonne modélisation, nous devons bien choisir une structure de modèle adéquat plutôt que de s'intéresser seulement à la performance d'une méthode d'apprentissage. Les réseaux de neurones, cependant, ont apporté une nouveauté dans les domaines comme la classification, la modélisation et la prévision de séries temporelles, la régression simple ou la reconnaissance de forme.

L'idée est d'extraire des combinaisons linéaires des entrées comme des caractéristiques dérivées et alors modéliser la cible ou le résultat espéré, comme une fonction non linéaire de ces caractéristiques. L'approche neuronale est plutôt basée sur une manipulation visuelle des variables en modifiant les structures des modèles, ce qui rend la méthode plus aisée et plus accessible aux non mathématiciens et aux personnes fuyant les calculs. Un autre avantage théorique qui a contribué à la réussite des réseaux de neurones et en particulier à un réseau de neurones à propagation directe connu sous le nom de perceptron multicouches : est la propriété d'approximation universelle. En effet, l'un des thèmes qui a intéressé les chercheurs est l'étude de la capacité d'approximation de fonctions par des perceptrons multicouches.

1.2. Exemples

De nombreux événements ne peuvent être décrits de manière précise et sont difficiles à quantifier. Ils sont aussi soumis à des perturbations et des impondérables appelés bruits.
Dans ce mémoire, nous nous intéressons à des séries d'événements quantifiables observés à des intervalles réguliers comme par exemple la consommation annuelle de gaz, d'électricité. Ces séries sont des séries temporelles à temps discret. Dans les deux exemples cités, les deux séries sont à valeurs réelles et .sont observées à des intervalles de temps différents. Aussi, les phénomènes ne sont pas indépendants : en effet, la consommation de gaz et d'électricité durant l'année 2002 par exemple est fortement corrélée avec celle des années précédentes. Mais pour un grand nombre de phénomènes, l'hypothèse d'indépendance n'a pas de sens et nous considérons en général les techniques qui prennent en considération la corrélation du phénomène avec son passé. Pour analyser ces données, la modélisation qui consiste à mettre en équation la série et à déterminer ses caractéristiques statistiques est l'une des solutions les plus employées. Pour des séries de données assez régulières et quand la

durée d'observations est assez longue, nous pouvons dégager certaines propriétés asymptotiques qui concernent le processus associé ou les méthodes statistiques de modélisation. Après avoir modélisé et analysé la série, nous pouvons alors essayer et c'est un but très recherché, de prévoir les évolutions futures du phénomène, en calculant les valeurs futures de la série : nous utilisons alors le passé pour prévoir l'avenir.

Avant de présenter la problématique et le plan de travail, nous introduisons quelques notations et rappels.

1.3. Notations et rappels

Nous utiliserons la notation suivante pour définir un modèle. Etant donné une suite finie de réels $(X_t)_{t=1,2....T}$, que nous voulons modéliser pour analyser ces données et en comprendre la structure sous-jacente associée. Nous voudrions aussi prévoir l'évolution future du phénomène, en calculant les valeurs $(X_{T+k})_{k\geq 1}$, nous sommes alors confrontés à un problème de prévision de séries temporelles. Différentes possibilités peuvent être envisagées quant à la suite $(X_t)_t$. Nous allons supposer les hypothèses suivantes :

- Les T observations $X_1, X_2, \ldots X_T$ sont extraites d'une suite infinie aléatoire $(X_t)_{t\in Z}$ appelée processus à temps discret.
- Au processus (X t) t∈Z est associée une suite de variables aléatoires indépendantes et identiquement distribuées (εt) t∈Z appelée bruit, de moyenne 0 et de variance σ^2 finie, avec $\forall t, s \in Z, s \leq t$ ε_t indépendant de X_s.
- X_t peut s'écrire sous la forme d'une fonction d'un certain nombre p de retards endogènes ($X_{t-1}, X_{t-2}, \ldots X_{t-p}$), de ε_t et d'un certain nombre q de retards du bruit ($\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$).

La perturbation $(\varepsilon_t)_{t\in Z}$ est vue comme la somme des erreurs de relevés des données dont on dispose et/ou des impondérables liés au phénomène.

$$\forall t \in / Z,\ X_t = f(X_{t-1}, X_{t-2}, \ldots . X_{t-p}, \varepsilon_{t-1}, \varepsilon_{t-2}, \ldots . \varepsilon_{t-q}) + \varepsilon_t \qquad (1.1)$$

(Nous avons supposé que $(\varepsilon_t)_{t\in Z}$ entre dans la formulation de X_t de façon additive).
Cette équation définit un modèle fonctionnel auto régressif avec moyenne mobile. Par la suite, nous noterons $X_t^{(p)}$ la suite de retards de longueur p associée à X_t et $\varepsilon_t^{(q)}$ la suite de retards de longueur q associée à ε_t :

$$\begin{cases} X_t^{(p)} = (X_t, X_{t-1}....X_{t-p+1}) \\ \varepsilon_t^{(q)} = (\varepsilon_t, \varepsilon_{t-1}....\varepsilon_{t-q+1}) \end{cases}$$

Dans l'écriture (1.1), la fonction f est inconnue. Effectuer une modélisation paramétrique consiste alors à approximer f par une fonction f_θ paramétrée par un vecteur de paramètres θ Nous écrivons alors (1.1) en utilisant l'approximation paramétrique :

$$X_t = f_\theta(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) + \varepsilon_t + \eta_t. \qquad (1.2)$$

avec $\eta_t = f(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) - f_\theta(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)})$. Ici f_θ peut être de la forme d'un modèle linéaire du type AR, d'un modèle bilinéaire du type polynomial, ou d'un réseau de neurones basé sur des combinaisons de fonctions sigmoïdes (nous en verrons la définition ultérieurement). Dans notre travail, nous nous intéressons à ce dernier modèle. Le terme η_t correspond alors à l'erreur due à l'approximation de f par f_θ .Nous considérons en général qu'il existe un vecteur de paramètres θ_0 tel que l'erreur η_t soit nulle ou négligeable devant ε_t.

(1.2) s'écrit alors : $$X_t = f_{\theta_0}(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) + \varepsilon_t. \qquad (1.3)$$

C'est cette équation qui définit le vrai modèle, et θ_0 est la vraie valeur du paramètre. Modéliser, c'est supposer alors que la structure du modèle est correcte et tenter d'ajuster le vecteur des paramètres θ . Enfin, le modèle s'écrit :

$$X_t = f_\theta(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) + \varepsilon_t. \qquad (1.4)$$

avec θ à estimer.

Dans le cas où les retards des bruits n'interviennent pas, nous parlons d'un modèle auto régressif fonctionnel (ARF) dont certains auteurs ont étudié les propriétés probabilistiques comme Doukhan et Ghines (1992); Robinson (1997); Jones (1978) et Duflo (1996).Ce modèle s'écrit : $$X_t = f_\theta(X_{t-1}^{(p)}) + \varepsilon_t. \qquad (1.5)$$

Une fois le cadre général défini, nous devons résoudre certains problèmes, tels que :

- Choisir la structure générale du modèle paramétrique (c'est-à-dire déterminer la famille de fonctions f_θ la plus adaptée aux données).
- Identifier un modèle au sein de cette famille (choisir une structure définie et fixe au sein de la famille de modèles).
- Estimer efficacement le vecteur des paramètres θ du modèle identifié (découvrir la valeur de θ_0).

- Evaluer la fiabilité du modèle à l'aide de tests statistiques sur les paramètres et sur les prévisions.

Ces quatre questions sont traitées en général de manière séquentielle, en utilisant des méthodes statistiques ou des heuristiques plus ou moins fiables, suivant la famille de modèle choisie et la quantité de données dont nous disposons. Nous pouvons même dire que si ces quatre étapes sont correctement spécifiées et mises en œuvre, la recherche aboutit, sauf rarement, à un modèle ayant les mêmes propriétés statistiques que le processus $(X_t)_{t \in Z}$ étudié et reproduisant son comportement à court ou à moyen terme.

Nous allons en premier lieu rappeler des définitions utiles pour la suite. Toutes les connaissances générales sur les modèles de prévision peuvent être trouvées dans plusieurs ouvrages classiques de base tels que « time series analysis » de Box et Jenkins F. M. (1976) et « time series theory and methods » de Brockwell et Davis (1991).

1.3.1. Processus stochastiques.

Définition 1.1. Un processus aléatoire est une famille de variables aléatoires indicées par le temps noté $\{X_t, t \in T\}$, définies sur le même espace probabilisable (Ω, F)

$$X_t : (\Omega, F) \to (IR, B_{IR})$$
$$\omega \to X_t(\omega)$$

$\mathbf{X_t}$ est mesurable, $\forall t \in T$ t est l'instant d'observation**,** T est l'espace des instants. Si T=IR**,** le processus est dit à temps continu. Si T= Z ou N ou N^*, il est dit à temps discret ou discret.

1.3.1.1. Auto covariance Soit X_t (notation simplifiée pour $\{X_t, t \in Z\}$), un processus, r et s deux instants. L'auto covariance (ou fonction d'auto covariance) de X_t pour les deux instants r et s est, par définition, la covariance des variables X_r et X_s. L'auto covariance s'écrit γ_X et :

$$\gamma_X (r, s) = Cov (X_r , X_s)$$

La fonction d'auto covariance est une notion importante, qui est aux séries temporelles ce que la matrice variance covariance est aux variables aléatoires classiques.

1.3.1.2. Stationnarité Un processus X_t discret est dit stationnaire (du second ordre) si :

- $E (|X_t|^2)$ est finie pour tout t entier ;
- $E (X_t) = \mu$ quel que soit t ;
- $\gamma_X (r + u , s + u) = \gamma_X (r , s)$ quels que soient r, s et u, entiers.

La première condition dit que les moments du second ordre de X_t et donc que les variables X_t

ne doivent pas prendre des valeurs infiniment grandes. La seconde dit que les espérances (les moyennes) des variables X_t sont égales. La troisième dit que la corrélation entre deux variables ne dépend que de l'écart entre les instants respectifs de ces variables. Ceci veut dire aussi que lorsqu'on se déplace sur l'axe du temps, la corrélation entre les variables séparées d'un certain délai est toujours la même. Cette condition ne dépend donc que du délai et peut s'écrire en fonction de ce dernier :

$$\gamma_X(h) = Cov(X_t, X_{t+h})$$

quels que soient t et h.

1.3.1.3. Auto corrélation Soit X_t un processus stationnaire, la notion d'auto corrélation découle de la notion d'auto covariance comme la corrélation de la covariance. Ainsi, l'auto corrélation (ou le coefficient d'auto corrélation) du processus X_t est

$$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = Cov(X_{t+h}, X_t)$$

pour tout t et h.

1.3.1.4. Processus Bruit Blanc (white noise) Un bruit blanc est un processus stochastique $\{\varepsilon_t, t \in T\}$ tel que :

- $E(\varepsilon_t) = 0 \quad \forall t \in Z$
- $var(\varepsilon_t) = \sigma^2 \quad \forall t \in Z$
- $Cov(\varepsilon_t, \varepsilon_{t+h}) = \begin{cases} \sigma_\varepsilon^2 & si\ h = 0 \\ 0 & si\ non \end{cases}$

1.3.1.5. Les modèles linéaires/non linéaires Il y a des modèles linéaires par rapport aux paramètres et des modèles linéaires par rapport aux variables.

Par exemple, le processus $(Y_t)_{t \in Z}$ défini par :

$$Y_t = \alpha L_n(Y_{t-1}) + \varepsilon_t \qquad \alpha \in IR$$

est un modèle non linéaire par rapport à Y_t mais linéaire par rapport à α. La méthode d'estimation classique et les tests appliqués aux paramètres dans le cas linéaire restent dans ce cas valides mais l'identification pour ces modèles est limitée et donc peu utilisée. Par la suite, par modèle linéaire et non linéaire nous associerons toujours par rapport aux paramètres.

Les modèles linéaires Le modèle est dit linéaire si la fonction paramétrée f_θ est linéaire en θ. Si nous réécrivons l'équation 1-4 avec une telle fonction, X_t s'écrit alors sous la forme d'une combinaison linéaire des retards du processus et du bruit :

$$X_t = \mu + \sum_{i=1}^{p} a_i X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t \qquad t \in Z$$

avec $a_i \in IR, \quad \forall i \in \{1,2,\cdots p\} \quad et \quad b_j \in IR, \quad \forall j \in \{1,2,\cdots q\}$.

Ces modèles appelés modèles ARMA pour autoregressive moving average, peuvent modéliser une large classe de processus stationnaires utilisés dans la pratique. Beaucoup de chercheurs ont étudié leurs propriétés statistiques et nous savons choisir le modèle le plus performant en ce qui concerne l'identification et l'estimation des paramètres. Mais malheureusement, ces modèles sont limités et ne sont plus utilisés dés que les relations entre les variables à des pas différents ne sont plus linéaires.

Les modèles non linéaires Par modèle non linéaire, nous entendons modèle non linéaire par rapport aux paramètres, et ayant une description qui ne correspond pas à celle d'un modèle linéaire. Il y a dans cette classe des modèles de structures variées, ayant des propriétés statistiques et des qualités d'approximation diverses. Nous devons donc définir des sous classes de modèles ou familles ayant les mêmes propriétés statistiques et structures bien définies pour faciliter l'identification. Il existe en général des combinaisons de fonctions particulières, non linéaires par rapport aux paramètres, appliquées aux retards du processus et du bruit blanc associé. Dans une telle modélisation, les comportements sont de natures différentes, il y a des non linéarités à cause d'une discontinuité due à un changement brusque d'évolution, c'est le cas des processus stationnaires par morceaux, une non linéarité quadratique, exponentielle ou autre. Nous cherchons une famille de modèles ayant des qualités d'approximation universelle, permettant de modéliser une grande classe de processus (par exemple les processus associés à l'équation (1.1) avec f continue sur un compact).

1.3.1.6. Modèles neuronaux Par réseaux de neurones nous entendons plusieurs modèles, de structures variées utilisés dans différents domaines. Nous distinguons ici les modèles à propagation directe qui sont des modèles entrée-sortie sans boucles et les modèles récurrents à boucles de la sortie vers l'entrée ou au sein même de sa structure. Ces derniers ne font pas partie de notre étude. Nous pouvons trouver difficile la mise en équation des modèles neuronaux mais ils ont l'avantage de se représenter et se manipuler sous la forme de graphes orientés qui décrivent les structures des modèles de manière tout aussi précise que la formalisation mathématique. Comme pour le modèle auto régressif linéaire AR (p), dans un modèle à propagation directe, X_t est fonction des retards (X_{t-1}, X_{t-2},…X_{t-p}) uniquement.

Le modèle est alors constitué d'un réseau de filtres non linéaires ordonnés selon un certain ordre et appliqués à une combinaison linéaire des retards. Ces filtres sont appelés « fonctions d'activation » ou « fonctions de transfert » et peuvent être de plusieurs natures.
Pour la prévision de séries temporelles, l'un des modèles les plus employés est basé sur le modèle du perceptron simple défini par Rosemblatt (1962).

Ce modèle, ainsi que le modèle dérivé le plus connu, le perceptron multicouches, sont décrits ainsi que leurs propriétés dans un prochain chapitre. Ils constituent une extension naturelle dans le domaine non linéaire des modèles autorégressifs linéaires. Comme nous allons le voir, les perceptrons multicouches jouissent de la qualité d'approximateur universel et peuvent reproduire n'importe quel type de comportement. Dans la suite, nous donnons la définition d'un perceptron multicouches à une seule couche cachée.

Définition1.2. Soient (p, n) $\in N^2$ non nuls, nous appelons modèle neuronal à une couche cachée un modèle de la forme :

$$X_t = \sum_{j=1}^{n} \alpha_j \psi\left(\sum_{i=1}^{p} \beta_{ij} X_{t-i} + \beta_{0j}\right) + \alpha_0 + \varepsilon_t$$

où

- n est le nombre de neurones de la couche cachée.
- p est le nombre d'entrées.
- $\theta = \left\{(\alpha_j)_{0\le i\le n}, (\beta_{ij})_{0\le j\le n}\right\} \in IR^{n\times(p+2)+1}$ est le vecteur des paramètres, l'ensemble des poids, ou encore connexions en langage connexionniste.
- ψ est une fonction de IR dans IR généralement non linéaire, et non polynomiale appelée fonction d'activation.

La fonction d'activation la plus employée est la fonction logistique $f(x) = \frac{1}{1+e^{-x}}$ ou la fonction tangente hyperbolique : $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

Un exemple d'une série engendrée par un modèle de type perceptron multicouches muni de fonction d'activation tangente hyperbolique et ayant deux entrées est donné par:
(α_0 = -0.4 ; α_1 = -1.2 ; α_2 =0.8, β_{11} =0.2 ; β_{21} =0 ; β_{01} = 0.2 ; β_{12} =0 ; β_{22} =0.8 ; β_{02} =0.3)

$$\begin{cases} X_t = -1.2\tanh(0.2X_{t-1} + 0.2) + 0.8\tanh(0.8X_{t-2} + 0.3) - 0.4 + \varepsilon_t \\ \varepsilon_t \to N(0,1) \end{cases}$$

Nous reviendrons dans un chapitre ultérieur à ce type de modèles.

1.3.2. Martingales

Une tribu sur Ω est une famille F d'événements, telle que :

$\Omega \in F$ et si $A \in F \Rightarrow A^C \in F$ et $\forall n \in IN, \quad A_n \in F \Rightarrow \bigcup A_n \in F$

Si C est une classe de parties de Ω, la plus petite tribu contenant C est dite tribu engendrée par C.

Si T : Ω→ $!R^n$ est un vecteur aléatoire, la tribu engendrée par T : notée σ(T); est la tribu σ (C) où C est l'image réciproque des boréliens de R^n par T.

Un espace probabilisé est un triplet (Ω; F; P) où F est une tribu sur Ω et P: F→[0; 1] une probabilité.

Soit L^2 l'espace des classes de variables aléatoires X telles que $E(X^2) < \infty$ muni du produit scalaire < X, Y> = E(X, Y) et de la norme $||X||^2 = \sqrt{\langle X, X \rangle} = \sqrt{E(X^2)}$, alors$L^2$ est un espace de Hilbert.

Filtration Soit (Ω; F; P) un espace probabilisé. Une filtration F_n est une suite croissante de sous tribus F_n de F. $\Im_n$= tribu des évènements antérieurs à n-1.

Un processus (X_n) est adapté si ∀ n, $X_n \in \Im_n$ .Il est dit prévisible si ∀ n, $X_n \in \Im_{n-1}$.

Si $\Im_n{}^X = \sigma (X_0; X_1; :::X_n)$, X_n est adapté à sa filtration naturelle.

Définition1.3. Soit (X_n) un processus $\Im_n$ -adapté, et intégrable. X_n est une martingale si $E(X_{n+1} | \Im_n) = X_n$; presque sûrement.

La notion de martingale est liée à celle de filtration F_n et parler de F_n –martingale serait plus correct ; c'est ce qu'on fera s'il y a la moindre ambiguïté.

Si aucune filtration n'est précisée, on dit qu'un processus réel $X = (\Omega; F; (X_n)_{n\geq 0}, P)$ est une martingale si c'est une $\Im_n{}^X$ -martingale où $\Im_n{}^X = \sigma (X_0; X_1; \ldots, X_n)$

La propriété fondamentale (et constamment utilisée en finance) d'une martingale, est que pour tout horizon T > 0, l'ensemble du processus $\{X_t, t \leq T\}$ est complètement déterminé par la valeur terminale X_T au sens où $X_t = E[X_T | F_t]$ pour tout $t \leq T$ et que c'est un objet qui a des propriétés très proches des sommes de variables aléatoires indépendantes.

1.3.2.1. Somme de variables aléatoires indépendantes et centrées Il est clair que le processus $(X_n)_{n\geq 0}$ est une martingale si et seulement si $E(X_{n+1} - X_n \mid \Im_n) = 0$. Donc si $(Y_n)_{n\geq 0}$ est un processus adapté)(intégrable ou positif), alors $X_n = Y_0 + Y_1 + ... + Y_n$ définit une martingale si et seulement si $E(Y_{n+1} \mid \Im_n) = 0$.

En particulier si $(Y_n)_{n\geq 0}$ est une suite de variables aléatoires indépendantes intégrables ou positives et si on pose $S_0 = 0$, $S_n = Y_0 + Y_1 + ... + Y_n$, $n\geq 0$, alors $(S_n)_{n\geq 0}$ est une martingale, pour la filtration $\sigma(Y_0; Y_1; :::Y_n)$.si, pour tout $n\geq 0$, $E(Y_n) = 0$.

1.3.2.2. Martingales de carré intégrable L'importance de la notion de martingale se justifie par certains résultats de convergence. Dans ce qui suit, il y a le résultat le plus fort :

Théorème 1.4. Soit M_n une martingale telle que $\sup_n E(Mn^2) < \infty$. Alors M_n converge p.s. et dans L^2.

Théorème 1.5. (Martingale et loi des grands nombres) Soit $Mn = \sum_{k=1}^{n} Y_k$ une martingale telle que $\sum_{n\geq 1} \frac{1}{n^2} E(Y_n^2) \prec 0$. Alors $\lim_n \frac{M_n}{n} = 0$ presque sûrement et dans L^2.

Soit $1 \leq p \leq \infty$. On dit que le processus X_n est borné dans L^p, s i $\sup_n E(\mid X_n \mid^p) < \infty$

1.3.2.3. Crochet d'une martingale bornée.

Proposition 1.1. Si $(M_t)_{t\geq 0}$ est une martingale continue bornée, il existe un unique processus croissant $(< M >_t)_{t\geq 0}$ tel que $< M >_0 = 0$ et $\left(M_t^2 - \langle M_t \rangle \right)_{t\geq 0}$ est une martingale. $(< M >_t)$ est appelé crochet de la martingale (M_t)

1.3.3. Rappels sur les séries temporelles

1.3.3.1. Outils d'analyse d'une série chronologique L'analyse des séries chronologiques fait l'objet de développements nombreux et récents notamment en économie. Dans ce qui suit, nous allons introduire les outils utilisés dans l'analyse chronologique.

a) Représentation graphique Lorsque l'on dispose d'un historique, on doit en premier lieu tracer le graphique de la série chronologique. Dans l'examen de l'allure générale de la courbe représentative de la série, on distingue quatre composantes fondamentales :

- **La composante tendancielle (trend) T_t** C'est un mouvement de longue durée, à la hausse ou à la baisse, qui représente l'évolution générale d'un phénomène

économique, appelée aussi mouvement de fond ou structurel. C'est une fonction du temps qui constitue la composante la plus importante dans une série chronologique.

- **La composante cyclique C_t** C'est un mouvement d'allure quasi-périodique comportant une phase croissante et une phase décroissante ; la tendance et le cycle sont souvent regroupés en une composante appelée l'extra saisonnier.
- **La composante saisonnière S_t** C'est une composante cyclique relativement régulière de période intra annuelle (semaines, mois, trimestre,…etc.). il y a le mouvement saisonnier rigide bien marqué et répétitif et celui souple pour lequel la saison est répétitive mais moins marquée.
- **La composante résiduelle R_t** Elle peut correspondre à un mouvement erratique. Elle rassemble tout ce que les autres composantes n'ont pu expliquer du phénomène observé (grève, guerre, catastrophes…etc.). Le résidu présente en général une allure aléatoire plus au moins stable autour de sa moyenne.

b) Calcul des statistiques de base Avant le traitement d'une série chronologique, on doit étudier certaines de ses caractéristiques stochastiques comme la moyenne et la variance.

- **La moyenne** la moyenne d'une série chronologique de terme X_t pour laquelle nous disposons de n observations est : $\overline{X} = \frac{1}{n}\sum_{t=1}^{n} X_t$.
- **La variance** la variance d'une chronique stationnaire permet d'évaluer la dispersion autour de sa moyenne. La variance empirique est donnée par : $\text{var}(X) = \frac{1}{n-1}\sum_{t=1}^{n}(X_t - \overline{X})^2$

c) Auto corrélation, corrélogramme

Si (r_t) est la suite (fonction) des autocovariances" (ACV) de X_t définie en 1.3.1.3 . La suite (ρ_h) telle que $\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = Cov(X_{t+h}, X_t)$ est appelée suite (fonction) des autocorrélations" (ACF en Anglais) de (X_t) (avec$(\rho_{_h}) = \rho_h$ et $|\rho_t| \le \rho_0 = 1$). Le corrélogramme est une représentation des coefficients d'auto corrélation (les ρ_h) d'une série chronologique.

Auto corrélation empirique et auto corrélation empirique

Les auto covariances empiriques $\hat{r}_h$ qui sont des estimateurs de r_h. sont définies par

$$\hat{r}_h = \frac{1}{T}\sum_{k=1}^{T-h}(X_k - \overline{X}_T)(X_{k+h} - \overline{X}_T) \quad 0 \le h \le T\text{-}1 \quad \text{et} \quad \hat{r}_h = \hat{r}_{-h} \quad \text{(symétrie)}$$

où $\overline{X}_T = \frac{1}{T}\sum_{t=1}^{T} X_t$. De même, l'auto corrélation empirique est définie par $\hat{\rho}_t = \frac{\hat{r}_t}{\hat{r}_o}$

d) Fonction d'autocorrélation partielle notée FAP (PACF en Anglais) c'est la corrélation entre X_t et X_{t-h}, l'influence des variables X_{t-h-i} $(i < h)$ ayant été retirée. Etant donnée la matrice des corrélations symétrique formée des (h-1) premières auto corrélations P_k ,la fonction d'autocorrélation partielle est donnée par : $\rho_{hh} = \frac{|P_h^*|}{|P_h|}$ où $|P_h^*|$ est le déterminant

de la matrice P_h en remplaçant la dernière colonne par le vecteur $(\rho_1,.....,\rho_h)$

$$P_h = \begin{bmatrix} 1 & \rho_1 & \ldots & \rho_{h-1} \\ \rho_1 & 1 & \ldots & \rho_{h-2} \\ . & & 1 & \\ . & & & \\ \rho_{h-1} & \rho_{h-2} & \ldots & 1 \end{bmatrix} \quad h \in N \qquad \text{Ainsi} \quad P_h^* = \begin{bmatrix} 1 & \rho_1 & \ldots & \rho_1 \\ \rho_1 & 1 & \ldots & \rho_2 \\ . & & & \\ . & & & \\ \rho_{h-1} & & \ldots & \rho_h \end{bmatrix}$$

Vu la complexité des calculs, on utilise l'écriture récurrente de ρ_{ii} où

$$\rho_{ij} = \rho_{i-1,j} - \rho_{ii}\ \rho_{i-1,i-j}, \quad j = 1,.....,i-1.$$

$$\rho_{ii} = \begin{cases} \rho_1 \quad si\ i = 1 \\ \dfrac{\rho_i - \sum_{j=1}^{i-1} \rho_{i-1,j}\ \rho_{i-j}}{1 - \sum_{j=1}^{i-1} \rho_{i-1,j}\ \rho_j} \quad i = 2,.....,h \end{cases}$$

Cet algorithme utilisant les équations de Yule-Walker est appelé algorithme de Durbin.

c) Les opérateurs linéaires

- **Opérateur de retard B (Backward)** Soit le processus $\{ X_t, t \in Z \}$; l'opérateur de retard B transforme une observation en sa valeur passée, il est défini par $B X_t = X_{t-1}$ Mais $(1- B)^2 X_t \neq (1-B^2) X_t$ car :$(1-B)^2 X_t = (1-2B+B^2) X_t = X_t-2X_{t-1}+X_{t-2} \neq X_t-X_{t-2}$
- **Opérateur d'avance F (Forward)** L'opérateur d'avance F associé à un processus $\{ X_t, t \in Z \}$ transforme une observation en sa valeur future : $F X_t = X_{t+1}$, il a les mêmes propriétés que le précédent.
- **Opérateur de différence ∇** Appliqué à une série pour lui omettre l'effet tendance**,** il est défini par : $\nabla X_t = (1-B) X_t = X_t-X_{t-1}$ pour $n \geq 1$, $\nabla^n X_t = (1-B)^n X_t$
- **Opérateur de différence saisonnière ∇_S** Appliqué à une série afin de lui ôter la saisonnalité, il est défini par :$\nabla_S X_t = (1-B^s) X_t = X_t-X_{t-S}$ On définit le $D^{éme}$ opérateur de différence saisonnière par : $\nabla_S^D X_t = (1-B^s)^D X_t$

1.3.3.2. Processus stationnaires **Les modèles ARMA et dérivés** Ce sont des modèles très utilisés dans la pratique; on peut atteindre avec cette classe de modèles des modélisations très variées avec un nombre limité de paramètres. Ils furent étudiés dans les années 70 par [(Box et Jenkins), 1976] qui donnèrent des techniques efficaces d'estimation du vecteur des paramètres et d'identification de ce modèle. [(Brockwell et Davis), 1991] ont réalisé une étude statistique complète de tels modèles. Les processus ARMA forment une famille de processus stationnaires [Wold, 1954].

Modèles ARMA Nous appelons modèle « autoregressive moving average » (ARMA (p, q)) pour $p \geq 1 \; et \; q \geq 1$ un modèle de la forme : $\phi(B)\; X_t = \theta(B)\; \varepsilon_t$ avec

ϕ (B)= 1-ϕ_1B-ϕ_2B^2-…….-ϕ_PB^p le polynôme autorégressif d'ordre p

θ (B)= 1-θ_1B-θ_2B^2-…….-θ_q B^q le polynôme moyenne mobile d'ordre q et $\varepsilon_t, t \in Z$ est processus bruit blanc $(0, \sigma_\varepsilon^2)$ et $(\phi_i, \theta_j) \in IR^2, \forall i \in \{1, \cdots p\} \; et \; \forall j \in \{1,2, \cdots q\}$.

Si q=0, on obtient le processus autorégressif AR d'ordre p, AR (p) donné par

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots\cdots + \phi_p X_{t-p} + \varepsilon_t$$

Si p=0, on obtient le processus moyenne mobile MA d'ordre q, MA (q) donné par :

$$X_t = \varepsilon_t - \theta_1 \; \varepsilon_{t-1} - \theta_2 \; \varepsilon_{t-2} \; \ldots\ldots\ldots - \theta_q \varepsilon_{t-q}$$

Chacun de ces modèles est caractérisé par ses fonctions d'auto corrélation simple (FAC) et d'auto corrélation partielle (FAP).

Akaike (1974) a étudié la stabilité de ces modèles. L'estimation est effectuée à l'aide de la méthode des moindres carrés, ou en résolvant le système de Yule - Walker. Dans cet exposé, nous nous limitons aux conditions assurant la stationnarité.

Proposition1.2. Soient $\psi(z)$ et $\theta(z)$ les polynômes relatifs au modèle ARMA de la définition précédente définis par :

$$\psi(z) = 1 - \phi_1 z - \phi_2 z^2 + \ldots\ldots - \phi_p z^p. \qquad \text{et} \qquad \theta(z) = 1 - \theta_1 z - \theta_2 z^2 + \ldots\ldots - \theta_q z^q.$$

Alors le modèle est stationnaire si et seulement si :

- Les polynômes $\psi(z)$ et $\theta(z)$ n'ont pas de racines communes. (1.6)
- Le polynôme $\psi(z)$ a toutes ses racines contenues dans le cercle unité.

Ainsi, un processus AR est toujours inversible, un processus MA est toujours stationnaire, il est inversible si les racines de θ (B)=0 sont à l'extérieur du cercle unité du plan complexe

Le tableau de l'annexe A.1 représente certaines des caractéristiques des modèles ARMA, dans lequel E désigne l'enveloppe représentée par le graphe de l'annexe A.2.

1.3.3.3. Processus non stationnaires ils sont représentatifs de la plupart des phénomènes aléatoires quand d'autres facteurs ne sont pas pris en compte en dépit de leur importance ou de leur imprévisibilité (grève, catastrophe,…). Les non stationnarité sont représentés par :

a) Les Processus TS Ce sont des processus qui présentent une non stationnarité de type déterministe (Trend Stationnary). Ils sont formés de deux composantes : $X_t = f_t + \varepsilon_t$

où f_t est une fonction polynomiale du temps, linéaire ou non, ε_t est un processus stationnaire.

b) Les Processus DS Ce sont des processus qui présentent une non stationnarité de type aléatoire (Differency Stationnary) donnés par l'équation : $X_t = X_{t-1} + \beta + \varepsilon_t$ où ε_t peut être un processus stationnaire .On distingue deux types de tels processus :

Pour $\beta=0$: X_t s'écrit : $X_t = X_{t-1} + \varepsilon_t$ .Le processus DS est dit sans dérive, il est appelé aussi marche aléatoire . Il s'agit d'un processus autorégressif d'ordre 1, explosif.

Pour $\beta\neq0$: X_t s'écrit $X_t = X_{t-1} + \beta + \varepsilon_t$ .Le processus DS est dit avec dérive, il s'agit d'un processus autorégressif explosif d'ordre 1 avec constante.

Pour stationnariser ces deux processus, on utilise le filtre aux différences :

Pour $\beta = 0$: $(1-B)^d X_t = \varepsilon_t$ et pour $\beta \neq 0$: $(1-B)^d X_t = \beta + \varepsilon_t$ où d est l'ordre de différenciation ou d'intégration

Remarque Nelson et Kang (1981) ont montré sur la base de simulation que la stationnarisation d'une série dépend du type de la tendance qu'elle exhibe. Du point de vue statistique et économique, e tromper sur le type de tendance aboutit à des résultats erronés, ce qui explique de ce fait sa grande importance ; en d'autres termes, si on traite un processus TS comme un processus DS, on introduit une perturbation artificielle ; à l'opposition, si on traite un processus DS comme TS, on introduit dans la série un mouvement cyclique long.

c) Les Processus ARIMA Ce sont des modèles ARMA intégrés notés ARIMA. Ils sont issus des séries stationnarisées par l'application du filtre aux différences. Le processus X_t suit un ARIMA (p, d, q), c'est-à-dire qu'il est solution d'une équation aux différences stochastiques du type : $\phi(B)(1-B)^d X_t = \theta(B)\varepsilon_t$.

d) Les Processus SARIMA Ce sont des extensions des processus ARMA et ARIMA. Ils représentent généralement des séries marquées par une saisonnalité comme c'est le plus

souvent le cas pour des séries économiques voire financières. Ces séries peuvent mieux s'ajuster par des modèles saisonniers. Ce sont les processus SARIMA (p, d, q) (P, D, Q) qui répondent au modèle: $\phi(B)\phi_s(B^s)(1-B)^d(1-B^s)^D X_t = \theta(B)\theta_s(B^s)\varepsilon_t$ où

$\phi(B)= 1-\phi_1 B-\phi_2 B^2-\ldots\ldots\ldots-\phi_p B^p$: polynôme autorégressif non saisonnier d'ordre p.

$\phi_s(B)= 1-\phi_{1s} B^s-\phi_{2s} B^{2s}-\ldots\ldots\ldots-\phi_{Ps} B^{Ps}$: polynôme autorégressif saisonnier d'ordre P.

$\theta(B)= 1-\theta_1 B-\theta_2 B^2-\ldots\ldots\ldots-\theta_q B^q$: polynôme moyenne mobile non saisonnier d'ordre q.

$\theta_s(B)= 1-\theta_s B^s-\theta_{2s} B^{2s}-\ldots\ldots\ldots-\theta_{Qs} B^{Qs}$: polynôme moyenne mobile saisonnier d'ordre Q

$(1-B)^d$: opérateur de différence d'ordre d . $(1-B^s)^D$ opérateur de différence saisonnière d'ordre D. $\varepsilon_t \rightarrow BB(0,\sigma_\varepsilon^2)$ et s la saisonnalité.

Les paramètres (p, d, q) de la partie non saisonnière, sont identifiés à l'aide de la coupure des fonctions d'auto corrélation simple et partielle. Pour la partie saisonnière (P, D, Q), les paramètres Pet Q sont identifiés en examinant la coupure des fonctions d'autocorrélation simple et partielle à des ordres multiples de s respectivement, D représente le nombre de fois où l'opérateur de différence saisonnière d'ordre s a été appliqué.

1.4. Problématique

Quand un statisticien observe un phénomène chronologique quantifiable sous la forme d'une suite finie de réels $(X_t)_{t=1,2,\ldots,T}$, il est appelé à modéliser ces données pour les analyser et en comprendre la structure. Il peut aussi prévoir l'évolution du phénomène, en calculant les valeurs futures $(X_{T+k})_{k\geq 1}$. Il est alors confronté à un problème de prévision de séries temporelles. Parmi les nombreux modèles non linéaires qui existent, les modèles neuronaux forment une classe de méthodes dites d'apprentissage qui ont été séparément développées en statistique et intelligence artificielle. Cette modélisation comporte deux phases :

- une phase d'apprentissage.
- une phase de validation du modèle.

Dans ce travail, nous essayons de montrer comment cette méthode paramétrique non linéaire fonctionne pour la prévision des séries temporelles. La modélisation consiste dans ce cas à fixer la structure générale du modèle et à trouver le vecteur de paramètres qui s'adapte le mieux aux données. Nous tentons en particulier d'énumérer les propriétés statistiques de ce modèle, pour évaluer et améliorer à travers différentes techniques sa capacité dans la prévision de séries temporelles pour ainsi relever le défaut d'une modélisation linéaire.

Il est organisé de manière à avoir comme idée principale la prévision de séries temporelles par les modèles paramétriques non linéaires, en particulier par des modèles de réseaux de neurones à propagation directe.
Dans un premier temps, nous développons le complément nécessaire à la présentation de notre document. Ainsi, nous distinguons quatre parties :

La partie1 est didactique : on y trouve une introduction et des rappels sur les modèles paramétriques utilisés dans la suite et sur les théorèmes de convergence.

La partie 2 est consacrée au perceptron multicouches et à ses propriétés, on y introduit le modèle NAR (neural autoregression) basé le perceptron multicouches après avoir détaillé les modèles auto régressifs fonctionnels linéaires ou non, et donné les conditions qui en assurent la stabilité. On termine par l'estimation du vecteur des paramètres associés à un modèle NAR, ainsi que les conditions de la consistance forte et la normalité asymptotique de l'estimateur des moindres carrés du vecteur des paramètres. Comme dernier paragraphe, on y présente les méthodes d'optimisation où on va détailler les algorithmes qui permettent de choisir les paramètres d'un réseau de neurones afin de minimiser la fonction erreur.

La partie 3 commence par une application pratique, où on passe à l'apprentissage du réseau de neurones en comparaison avec l'approche de Box et Jenkins et ceci dans une application, celle de la prévision de la consommation d'électricité et de gaz dans l'une des villes de l'Algérie. Enfin, on va donner dans les parties annexes certains rappels ainsi que quelques résultats de calcul obtenus dans chacune des deux phases d'estimation, celle de SARIMA et celle des réseaux de neurones respectivement. Ensuite, on expliquera les fonctionnalités du logiciel utilisé pour la partie neuronale. Nous terminerons par une conclusion (la partie 4) et enfin une bibliographie.

Dans le chapitre suivant, nous allons après un bref historique, définir le perception multicouches ainsi que toute la théorie qui lui est liée (propriétés, modèle NAR,…etc).

2. PERCEPTRONS MULTICOUCHES ET MODELE NEURONAL

2.1. Introduction

C'est dans les années 40, que McCulloch W. et Pitts S. (1943), Turing A. (1947), Von Neumann J. (1945) et d'autres chercheurs ont tenté de mettre à profit les connaissances nouvelles apportées par les biologistes et les sciences cognitives sur le cerveau pour concevoir des systèmes censés reproduire certaines de ses fonctionnalités telles que l'apprentissage de taches complexes, la capacité de raisonnement et de déduction et enfin la possibilité d'évaluation, d'estimation et de résolution de problèmes. Deux écoles ont alors vu le jour, l'une ayant une vision « connexionniste » [McCullogh, Minsky, Pitts] et une autre adoptant une démarche « symbolique » [Von Neumann, et Turing]. Cette approche aussi appelée IA (intelligence artificielle) forte est basée sur une modélisation symbolique de l'environnement dans lequel nous évoluons en construisant des structures d'entités ordonnées, codifiées par des symboles, et en définissant les propriétés de ces entités, ainsi que leurs liens et relations. On lui doit en particulier l'ordinateur et les bases de l'intelligence artificielle. L'approche connexionniste, appelée IA faible, est inspirée de la description biologique, en essayant de construire des systèmes proches du cerveau dans leur organisation, pour reproduire certaines de ses particularités telles que :

- une mémoire distribuée et non localisée.
- un apprentissage adaptatif par modifications locales successives.
- une robustesse à la détérioration en cas de destruction partielle.

Dés 1943, McCullogh et Pitts ont formalisé le modèle de neurone formel basé sur les observations neurophysiologiques des neurones du système nerveux. Ce neurone est jusqu'à ce jour un élément de base de la plupart des modèles connexionnistes. Plusieurs variantes ont été proposées plus ou moins plausibles mais revenant toujours au concept présenté à cette époque. Nous savons néanmoins que ce modèle n'est qu'une approximation du neurone biologique mais que nous ne pouvons pas s'en servir pour comprendre profondément le fonctionnement du système nerveux. Dans les années 40, le neurone formel dit de McCullogh et Pitts suscita un intérêt parmi les pionniers du connexionnisme, mais la première application n'apparut qu'au début des années 60, Rosenblatt (1962) introduisit Le perceptron simple qui

désigne un ensemble de neurones formels connectés et fut le premier à y associer un algorithme d'apprentissage. Dans ce chapitre, nous détaillons le perceptron multicouches, ses propriétés statistiques, ainsi que les techniques d'estimation. Deux modèles classiques vont être décrits dans la première partie de ce chapitre, le perceptron proposé par Rosenblatt (1959) à la fin des années 50 et le modèle Adaline présenté par Widrow et Ho (1960) au début des années 60.

2.2. Perceptrons multicouches

2.2.1. Définitions et notations

2.2.1.1. Point de vue biologique

Neurone = unité simple, capable de faire des calculs élémentaires.

Les neurones sont connectés entre eux par des dendrites (entrées) et des axones (sorties).
La notion de synapse explique la transmission des signaux entre un axone et une dendrite

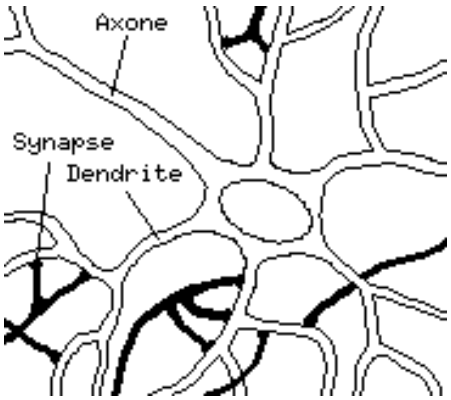


Fig 2.1 Le neurone biologique

Les neurones communiquent entre eux en émettant des signaux électriques. Chaque neurone effectue une « sommation » des signaux électriques qui lui parviennent par les dendrites. Si cette sommation dépasse un seuil, le neurone émet à son tour un signal sur son axone. Entre les axones et les dendrites, la transmission du signal est assurée par les synapses .Il existe

deux types de synapses : les synapses excitatrices, qui renvoient un signal de sortie proportionnel au signal d'entrée, et les synapses inhibitrices qui renvoient un signal inversement proportionnel au signal d'entrée.

2.2.1.2. Le neurone formel

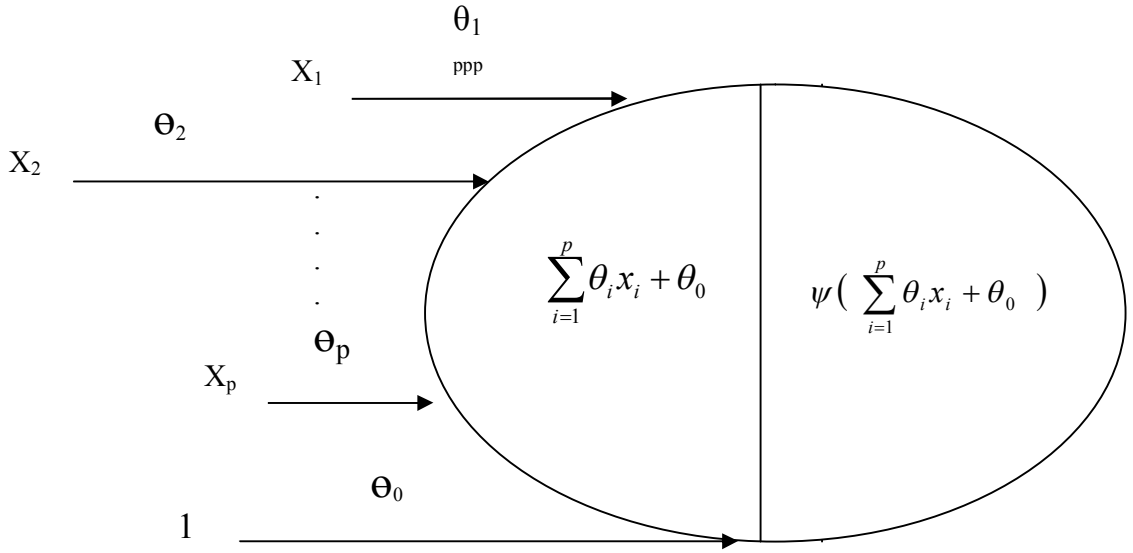


Fig. 2.2 le neurone : unité de base du traitement neuronal

Le neurone formel introduit par Mc Cullogh et Pitts (1943) est un automate reproduisant la composée de plusieurs fonctions très simples (voir Fig.2.2).Chacune des p liaisons synaptiques entrantes est affectée d'un poids θ_i, $i \in \{1,2,\ldots.p\}$.Par convention, on ajoute aussi une entrée constante égale à 1, pondérée par un poids θ_0.appelé biais L'opposé de θ_0 peut alors être vu comme une valeur seuil, au-delà de laquelle le neurone est activé. Le neurone effectue les deux opérations suivantes en calculant :

- Son potentiel, c'est-à-dire la somme pondérée des entrées $\sum_{i=1}^{p}\theta_i x_i$.
- Son activation à travers le filtre d'une fonction d'activation ou fonction de transfert ψ, en calculant $\psi(\sum_{i=1}^{p}\theta_i x_i + \theta_0)$.

La fonction d'activation peut être linéaire et on parle de réseau linéaire ou non linéaire (on a alors un réseau non linéaire). La fonction d'activation ψ la plus simple est la fonction signe S (voir Fig.2.5) appelée aussi fonction seuil ou fonction de Heaviside (qui n'est pas linéaire)

$$\text{S : IR} \rightarrow \text{IR définie par : } \begin{cases} S(x) = 1 & si\ x \geq 0 \\ S(x) = -1 & \sin on \end{cases} \qquad (2.1)$$

La sortie du réseau ainsi est soit 1 soit -1, en fonction des entrées. Le réseau peut alors être utilisé pour une classification où étant données deux classes, il peut décider si un échantillon d'entrées appartient à l'une de deux classes .Si l'entrée totale est positive, l'échantillon sera

assigné à la classe +1, si l'entrée totale est négative, l'échantillon sera assigné à la classe -1. Un réseau à une couche représente une fonction discriminante linéaire.

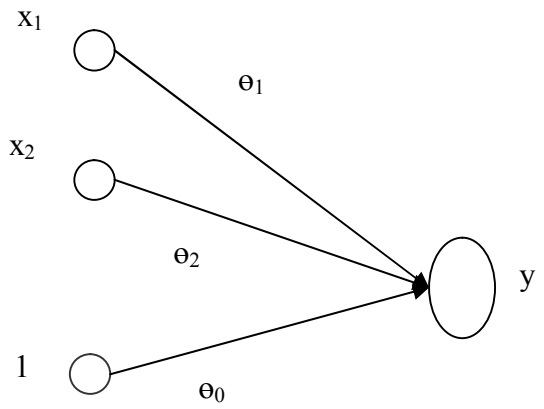


Fig 2.3 Un réseau à une couche avec deux entrées et une sortie

La séparation entre les deux classes dans ce cas est une ligne droite donnée par l'équation:

$$\theta_1 x_1 + \theta_2 x_2 + \theta_0 = 0 \qquad (2.2)$$

Une représentation géométrique du réseau de neurones linéaire à seuil est donnée en Fig 2.3. L'équation (2.2) peut être écrite comme:

$$x_2 = -\frac{\theta_1}{\theta_2} x_1 - \frac{\theta_0}{\theta_2} \qquad (2.3)$$

Et on voit que les poids déterminent la pente de la ligne et le biais détermine "la compensation" i.e. de combien la ligne est loin de l'origine. Remarquer aussi que les poids peuvent être représentés dans l'espace des entrées: le vecteur poids est toujours perpendiculaire à la fonction discriminante.

ψ peut aussi être définie à valeurs dans [0,1].On peut aussi utiliser d'autres fonctions plus régulières, la fonction de Gauss par exemple (voir Fig.2.6) ; cependant, la famille de fonctions la plus utilisée est celle des fonctions sigmoïdes :

$$x \to \sigma_{c,k,r}(x) = c\frac{e^{kx}-1}{e^{kx}+1} + r; c, k, r \in IR : c, k \succ 0 \qquad (2.4)$$

En faisant varier le paramètre k, on retrouve dans cette famille des fonctions qui approximent la fonction signe ; pour {c=1 ; r=0} on a :

$$x \to \sigma_{1,k,0}(x) = \frac{e^{kx}-1}{e^{kx}+1} = \tanh(\frac{kx}{2}) \qquad (2.5)$$

Sur IR*, la fonction $x \to \sigma_{1,k,0}(x)$ tend vers la fonction signe S quand k$\to+\infty$.La fonction : x $\to$ tanh (x)= $\frac{e^x - e^{-x}}{e^x + e^{-x}}$ (voir Fig.2.8) qui est la fonction sigmoïde la plus utilisée est alors

obtenue en prenant {c=1; k=2; r=0}, et la fonction logistique : $x \rightarrow 1 / (1+e^{-x})$ (voir Fig.2.7) est obtenue en prenant {c=1/2; k=1; r=1/2}.

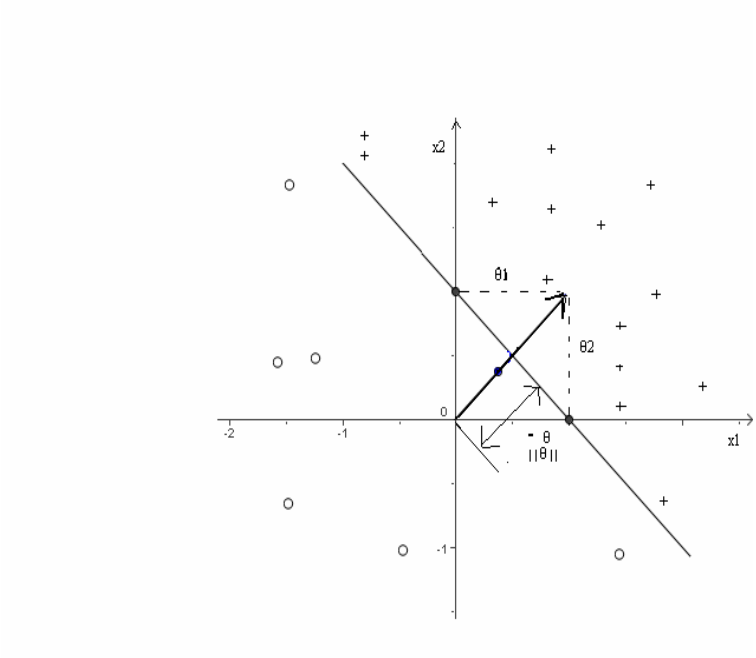


Fig 2.4 : Représentation géométrique de la fonction discriminante et des poids

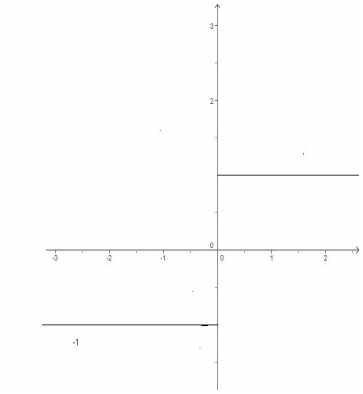


Fig. 2.5 Fonction signe

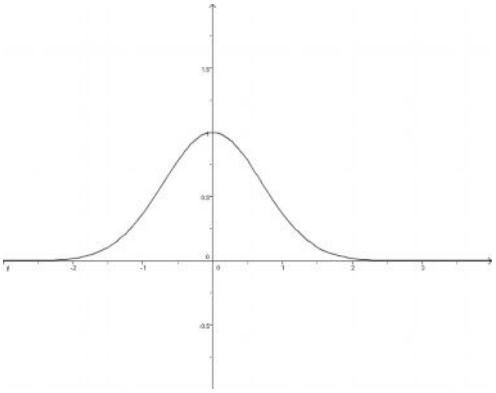


Fig. 2.6 Fonction de Gauss

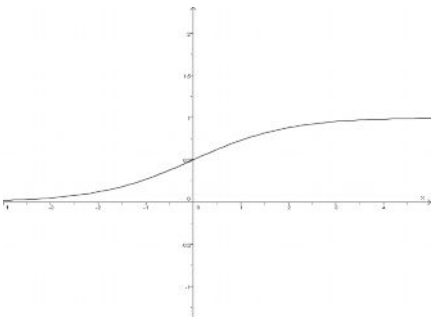


Fig. 2.7 Fonction logistique

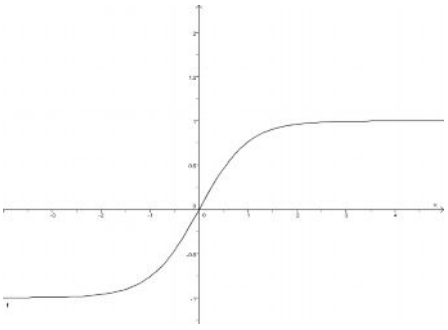


Fig. 2.8 Fonction tangente hyperbolique

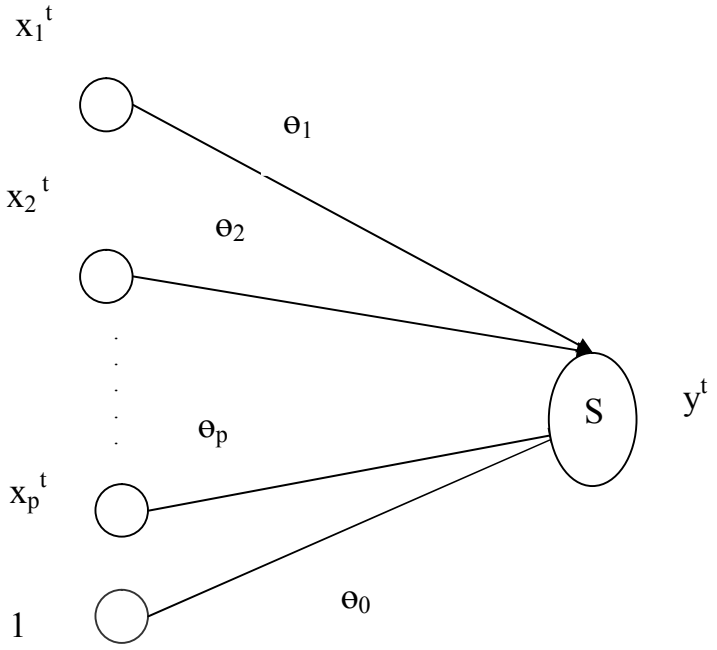


Fig.2.9 Le perceptron simple

2.2.2. Le perceptron simple

Dans un perceptron simple (voir Fig. 2.9), les unités de la couche d'entrées sont directement reliées à l'unité de la couche de sortie. Les poids des connexions reliant l'entrée de dimension p (à laquelle on ajoute par convention une entrée constante égale à 1) à la sortie scalaire sont notés $\theta = \{\theta_0, \theta_1, \ldots. \theta_p\}$.Le but de ce modèle est d'apprendre progressivement à séparer deux parties(classes) finies A et B de IR^p. Après l'apprentissage, on veut que le réseau réponde '1' sur présentation de tout élément de A et '-1'sur présentation de tout élément de B. Donc on cherche à calculer le vecteur des paramètres ou poids des connexions

θ tels que $\forall x=\{x_1, x_2, \ldots. x_p\} \in A \quad \sum_{i=1}^{p} \theta_i x_i + \theta_0 \succ 0 \quad et \; \forall x' = (x'_1, x'_2, \ldots..x'_p) \in B,$

$\sum_{i=1}^{p} \theta_i x'_i + \theta_0 \prec 0$. Pour cela, on présente successivement des éléments d'une base d'exemples de taille T. Cette base est formée d'un ensemble de vecteurs $(x^t)_{t=1,2\ldots..T} \in A \cup B$,$x^t=(x^t_1, \ldots x^t_p)$,et de valeurs binaires associées $(d^t)_{t=1,2\ldots T} \in \{-1,1\}$ associées à'1'si $x^t \in A$ et '-1' si $x^t \in B$ (les $(d^t)_{t=1,2\ldots T}$ sont appelées valeurs désirées). Lorsqu'on présente x^t, on ajuste alors le vecteur des paramètres $\{ \theta'_0, \theta'_1 \ldots. \theta'_p \}$ à partir du vecteur des paramètres précédent θ^{t-1}, de sorte que la sortie y^t du réseau s'approche de la sortie désirée d^t.

Rosenblatt (1962) a proposé la règle suivante d'apprentissage :

$\theta^0 = 0 \in IR^{p+1}$, $\forall t, \theta^{t+1} = \theta^t + \frac{1}{2}\varepsilon(d^t - y^t)x^t$ où ε est un pas unidimensionnel de déplacement dans l'espace des paramètres, petit, positif, et dont la valeur est à régler.

2.2.2.1. Ensemble linéairement séparable Soit S un ensemble d'exemples dans $R^n \times \{-1,1\}$.On note $S_0 = \{s \in R^n \mid (s, -1) \in S\}$ et $S_1 = \{s \in R^n \mid (s, 1) \in S\}$. On dit que S est linéairement séparable s'il existe un hyperplan H de R^n tel que les ensembles S_0 et S_1 soient situés de part et d'autre de cet hyperplan.

Théorème 2.1. Un perceptron linéaire à seuil à n entrées divise l'espace des entrées R^n en deux sous-espaces délimités par un hyperplan. Réciproquement, tout ensemble linéairement séparable peut être discriminé par un perceptron.

Démonstration Il suffit pour s'en convaincre de se rappeler que l'équation d'un hyperplan dans un espace de dimension n est de la forme $a_1 x_1 + ... + a_n x_n = b$. Un perceptron est donc un discriminant linéaire. On montre facilement qu'un échantillon de R^n est séparable par un hyperplan si et seulement si l'échantillon de R^{n+1} obtenu en rajoutant une entrée toujours égale à 1 est séparable par un hyperplan passant par l'origine.

Dans les années 60, Rosenblatt (1962) a démontré le théorème de convergence suivant qui suscita beaucoup d'espoir :

Théorème 2.2. [Rosenblatt (1962)] Si les parties A et B sont strictement linéairement séparables, la suite $(\theta^t)_{t\geq 0}$ converge en un nombre fini d'étapes vers un vecteur θ^∞ vérifiant :

$$\forall x = (x_1, x_2 \ldots .. x_p) \in A \quad \sum_{i=1}^{p} \theta_i^{\infty} x_i + \theta_0^{\infty} > 0 \quad . \text{et} \tag{2.6}$$

$$\forall x = (x_1, x_2 \ldots .. x_p) \in B \quad \sum_{i=1}^{p} \theta_i^{\infty} x_i + \theta_0^{\infty} < 0 \quad . \tag{2.7}$$

D'un point de vue géométrique, à l'instant t, le vecteur des paramètres θ^t définit un hyperplan $H = \left\{ x / \sum_{i=1}^{p} \theta_i^t x_i^t + \theta_0^t = 0 \right\}$ qui divise l'espace des entrées IR^p en deux demi-espaces

$H^+ = \left\{ x / \sum_{i=1}^{p} \theta_i^t x_i^t + \theta_0 \geq 0 \right\}$ et $H^- = \left\{ x / \sum_{i=1}^{p} \theta_i^t x_i^t + \theta_0 \leq 0 \right\}$.

Notons que lorsque θ_0^t est nul, l'hyperplan trouvé passe par l'origine.

Plus tard, les deux mathématiciens Minsky et Papert (1969) ont montré l'incapacité théorique du perceptron simple incapable de séparer deux ensembles non linéairement séparables, ce qui a constitué un grave handicap. Leur démonstration est illustrée par le célèbre exemple du « ou exclusif », (XOR), incapable d'être modélisé par le perceptron simple décrit ci-dessous.

2.2.2.2. La fonction XOR La fonction XOR appelée aussi « ou exclusif » est une fonction utilisée en logique .Elle revient à associer une valeur de sortie pour deux cellules d'entrée de la manière suivante :

$$\begin{array}{ccc} 0 & 0 & \rightarrow 0 \\ 1 & 0 & \rightarrow 1 \\ 0 & 1 & \rightarrow 1 \\ 1 & 1 & \rightarrow 0 \end{array}$$

Théorème 2.3. Le XOR ne peut pas être calculé par un perceptron linéaire à seuil.

Démonstration algébrique Si nous avons un perceptron avec deux cellules d'entrée et une cellule de sortie qui doit apprendre cette fonction (Fig.2.10), Il possède des poids Θ_1 et Θ_2 qui relient les cellules d'entrée à celle de la sortie et le fait d'associer les valeurs (1.0) d'entrée à 1 en sortie implique que $\Theta_1 > 0$.Et de même, le fait d'associer les valeurs (0.1) d'entrée à 1 en sortie implique que $\Theta_2 > 0$.De ce fait, lorsque les valeurs (1,1) sont présentées en entrée, le perceptron donnera toujours la réponse 1 en sortie , ce qui est contraire à ce qu'on veut lui faire apprendre. Il est donc impossible de trouver un ensemble de valeurs Θ_i permettant à un perceptron d'apprendre la fonction XOR.

Démonstration géométrique La fonction XOR n'est pas linéairement séparable, puisqu'il est impossible de tracer une ligne telle que les sommets (0,0) et (1,1) soient d'un même coté de la ligne et les sommets (0,1) et (1,0) de l'autre(Fig.2.11). En revanche, la fonction « ET » avec la table de vérité suivante est linéairement séparable :

$$\begin{array}{ccc} 0 & 0 & \rightarrow 0 \\ 1 & 0 & \rightarrow 0 \\ 0 & 1 & \rightarrow 0 \\ 1 & 1 & \rightarrow 1 \end{array}$$

Puisque l'on peut tracer une droite qui sépare les objets devant avoir une valeur de sortie 1 des objets devant avoir une valeur de sortie 0 (voir Fig.2.12).

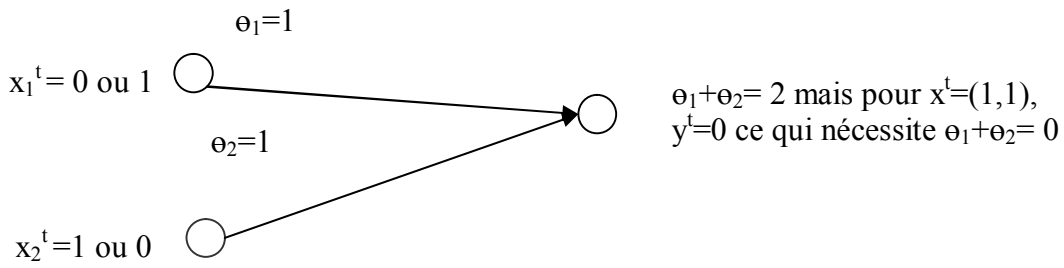


Fig 2.10 Impossibilité d'apprentissage de la fonction XOR par un perceptron

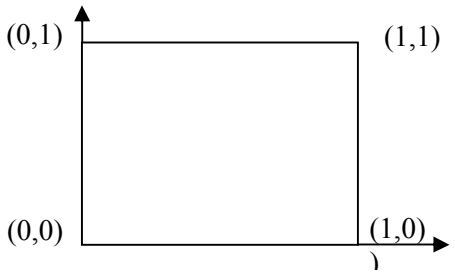


Figure 2.11 : Comment trouver une droite séparant les points (0,0) et (1,1) des points (0,1) et (1,0) ?

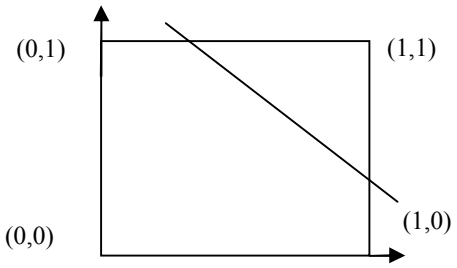


Figure 2.12 : la fonction « ET » est linéairement séparable

La solution naturelle pour dépasser les limites imposées au perceptron simple est très vite apparue en structurant le réseau en couches, de sorte à voir une composition de plusieurs perceptrons simples : on obtient alors un réseau appelé perceptron multicouches. Mais l'algorithme d'apprentissage de Rosenblatt ne fonctionnait pas, ce qui a rendu le modèle inutilisable et les chercheurs immédiatement se désintéressèrent des réseaux de neurones et se tournèrent vers l'approche symbolique plus intéressante.

Dans les années 1980, il y a eu un retour vers cet outil avec l'élaboration de l'algorithme de rétro propagation du gradient par [Rumelhart et al (1986)] et [Lecun et al (1985).]. Cet algorithme utilise un calcul de dérivées de fonctions composées permettant un apprentissage du perceptron multicouches en minimisant une fonction d'erreur.

2.2.3. Le perceptron multicouches

Comme il est montré à la figure 2.13, on peut résoudre le problème du XOR qui n'était pas soluble par un perceptron simple (voir annexes A.1 et Tab A.3).
La possibilité de séparer des ensembles non linéairement séparables justifia alors l'intérêt suscité par le perceptron multicouches. Celui décrit à la figure 2.14, adapté au problème de la régression comporte p unités en entrée recevant p variables (X_1, X_2,…..X_p), et une seule unité de sortie qui produit la variable Y. Si le réseau a n neurones sur sa couche cachée, il est noté PM (p, n, 1).Un neurone seuil est aussi défini, correspondant à une entrée égale à 1.

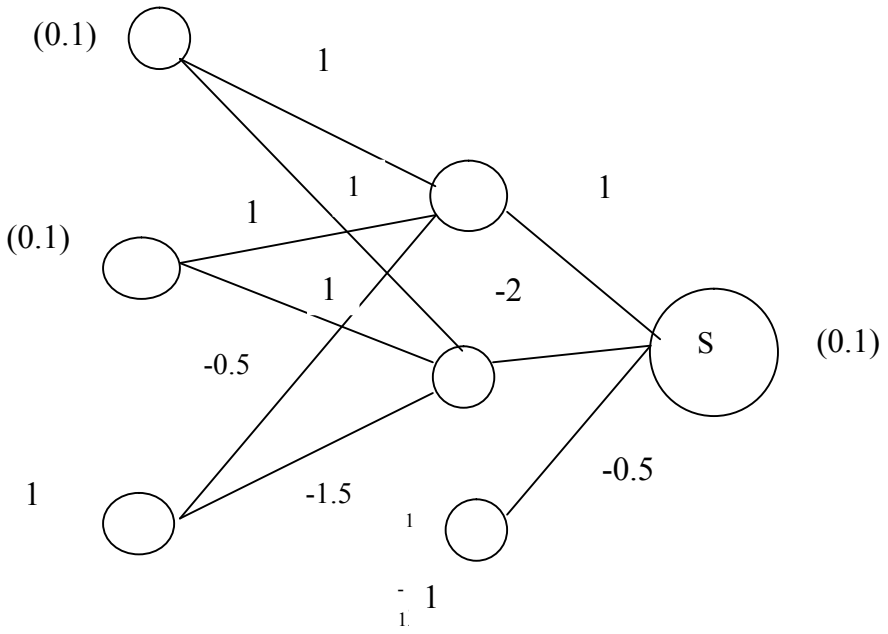


Fig 2.13 Le réseau de neurones qui résout le problème du XOR

Soit alors un réseau avec une sortie scalaire déterminée par l'équation (voir fig 2.14) :

$$Y = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ji} X_i + \beta_{j0}) + \alpha_0 \qquad (2.8)$$

où

- n est le nombre de neurones de la couche cachée.
- p est celui des neurones d'entrées.
- $\theta = \left\{(\alpha_j)_{0\le j\le n} (\beta_{ji})_{0\le i\le p, 1\le j\le n}\right\} \in IR^{n\times(p+2)+1}$ est le vecteur des paramètres ou poids, ou encore connexions en langage connexionniste.
- $\{\alpha_0, \beta_{10}, ..., \beta_{n0}\}$ (des réels) sont les poids reliés aux neurones de seuil.
- ψ est une fonction de IR dans IR généralement ni linéaire, ni polynomiale.

β_{ji} est le poids de la connexion qui relie le neurone j de la couche cachée à l'entrée i. Par convention, les neurones d'entrée ont toujours une fonction d'activation « identité », laissant passer l'information sans la modifier. Pour le neurone de sortie, nous pouvons lui associer une fonction d'activation ou non, selon la nature du problème à résoudre. Pour la fonction d'activation associée aux neurones de la couche cachée, nous pouvons utiliser toutes celles citées section 2.1, mais pour notre étude, une fonction d'activation sigmoïde sera utilisée. L'équation 2.8 définit alors un modèle de régression non linéaire paramétrée par le vecteur θ. Celle-ci s'exprime sous une forme graphique comme le montre la fig.2.14, ce qui permet une manipulation visuelle et simple des variables utilisées. Cette particularité permet aux utilisateurs des réseaux de neurones de redécouvrir visuellement la régression ou la classification, qui sont en général plutôt abordées par le biais d'équations. La fig 2.15 représente un perceptron multicouches à l couches, N_i la couche d'entrées, $N_{h,k}$ k=1…l-2 les (l-2) couches cachées et N_o la couche des sorties : la sortie des unités cachées est distribuée à travers la couche suivante des $N_{h,2}$ unités cachées, jusqu à la dernière couche des unités cachées pour laquelle les sorties sont à l'intérieur d'une couche de N_o unités de sortie. En effet, on peut avoir un vecteur sortie mais pour notre étude on n'a qu'une seule sortie.

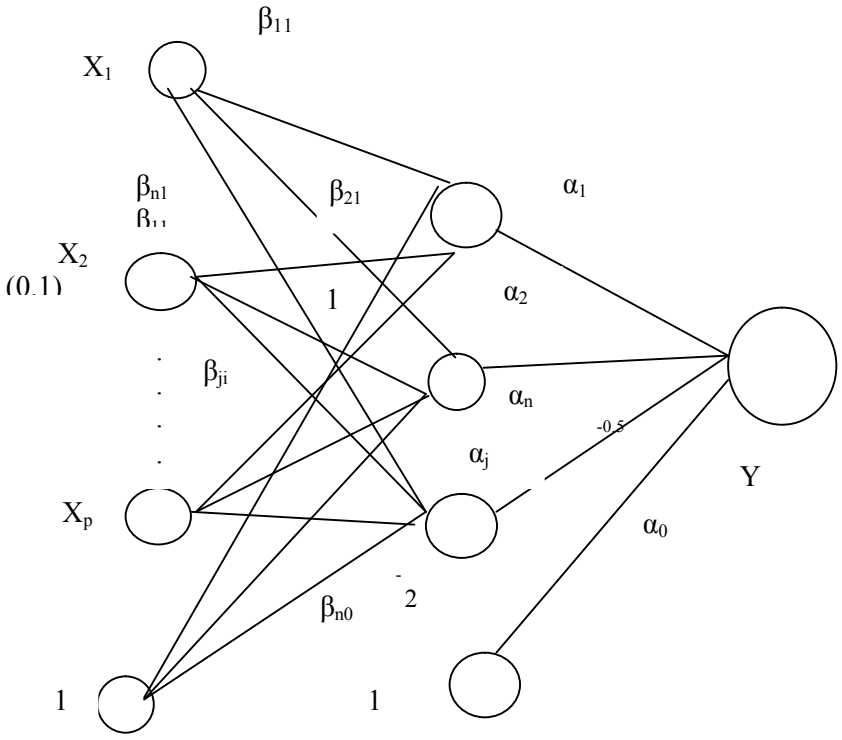


Fig 2.14 Exemple de réseau de neurones du type perceptron multicouches (mêmes notations que 2.8)

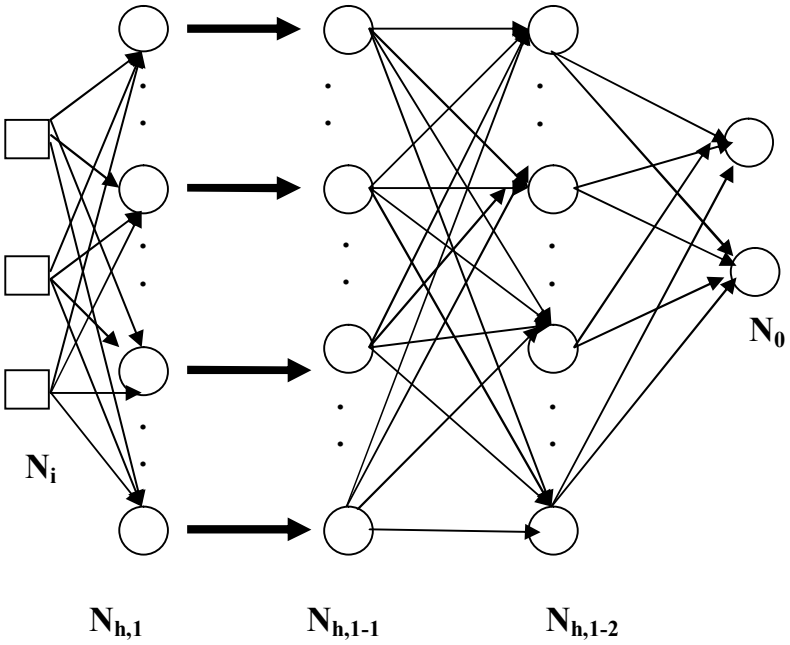


Fig 2.15 un réseau multicouches avec l couches

2.2.3.1. Propriétés des perceptrons multicouches Les perceptrons multicouches ont rapidement suscité l'intérêt des mathématiciens car ils concernent des thèmes généraux classiques (approximation fonctionnelle, processus de Markov, algorithmes adaptatifs,…), mais ils présentent des difficultés du fait de leur caractère non linéaire et si les simulations sont très avancées, le coté mathématique reste à développer. Une importance particulière a été donnée à l'approximation d'une fonction par un perceptron multicouches. Il a ainsi été démontré qu'un perceptron multicouches avec une seule couche cachée pourvue d'un nombre suffisant de neurones, peur approcher n'importe quelle fonction continue sur un compact de IR^p avec la précision voulue. Malheureusement cette propriété ne nous donne pas pour une fonction donnée le nombre maximal de neurones sur la couche cachée, donc nous ne pouvons pas dire que ce résultat mène vers une technique de construction d'architecture.

a) Le perceptron multicouches : un approximateur universel Bien que récent, le problème de l'approximation d'une fonction par des perceptrons multicouches a été un sujet d'intérêt de beaucoup de chercheurs, Cybenko (1989) ; Funahashi (1989) ; Barron (1993) ; Hornik et al. (1989) ont étudié la propriété d'approximation d'une fonction continue sur un compact par des perceptrons multicouches à une seule couche cachée, munis de fonction d'activation sigmoïde. Les travaux de Hornik et al. (1989) étendus ensuite par Leshno et al

(1993) aux perceptrons multicouches munis de fonctions d'activation quelconques non linéaires, non polynomiales ont conduit au résultat suivant :

Théorème 2.4. [Hornik et al. (1989)] En utilisant les notations de l'équation 2.8 avec ψ (.) une fonction d'activation strictement croissante et bornée et K un compact de IR^p alors, pour n'importe quelle fonction f$\in$ C(K) où C(K) est l'ensemble des fonctions continues sur K, et pour tout $\varepsilon > 0$, il existe un entier n et un vecteur de paramètres :

$\theta = \{(\alpha_i)_{0\le i\le n} (\beta_{ji})_{0\le i\le p, 1\le j\le n}\} \in IR^{n\times(p+2)+1}$ tels que $\forall (X_1, X_2, \ldots..X_p) \in K$:

$$\left| f(X_1, X_2 ....., X_p) - \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ji} X_i + \beta_{0j}) + \alpha_0 \right| \prec \varepsilon$$

La démonstration qui peut être retrouvée dans Hornik et al. (1989) pages 359-366 repose sur le théorème de Stone Weierstrass. Attali et Pagès (1995a) aussi donnèrent une preuve simple de cette propriété basée sur les développements de Taylor-Young. Elle fonctionne pour une classe de perceptrons multicouches ; ils ont ainsi obtenu une borne supérieure du nombre de neurones sur la couche cachée mais assez grande.

b) Vitesse de convergence de l'approximation Il n'y a pas suffisamment de résultats sur la vitesse d'approximation d'une fonction de régularité donnée, en fonction du nombre de paramètres (connexions) ou du nombre de neurones sur la couche cachée. Il y a néanmoins les résultats de Attali et Pagès (1995a), de Barron (1993) et ceux de Roynette (1993), dans le cas d'un perceptron multicouches comportant une seule entrée scalaire (p=1) et pour des fonctions d'activation sigmoïdes associées aux neurones de la couche cachée.

Théorème 2.5. [Attali et Pagès (1995a)] Soit K un sous-ensemble compact de IR^p .On pose $M_K = \sup_{x\in K} \|x\|$ et $\delta_K = \sup_{(x, y) \in K^2} \|x - y\|$ .Soit $\psi \in C^\infty$ (IR, IR) une fonction non polynomiale telle que $\forall$ k$\in$ IN $\psi^{(k)} \neq 0$.Soit f une fonction quelconque dont toutes les dérivées jusqu'à l'ordre p appartiennent à C (K, IR) et telle que

$\forall$i, $1\le i \le k$, $\dfrac{\partial f^{(k)}}{\partial x_i}$ soit p-lipschitzienne. Soit $(\varepsilon_{n)\ n>0}$ une suite de valeurs strictement positives, avec $\lim_{\lim_{n\to\infty}} \varepsilon_n = 0$, alors il existe une suite $(g_n)_{n\ge 0}$ de PM (p, n ,1) munis de fonctions d'activation ψ associées aux n neurones de la couche cachée telle que :

$$\|f - g_n\| \le \rho A_p {M_K}^{p+1} \frac{(1+\varepsilon_n)}{n^{p+1}} \qquad (2.9)$$

où A_p est une constante qui dépend uniquement de p.
Nous remarquons que les bornes trouvées sont de l'ordre de $O(1/n^{p+1})$ pour une fonction continue sur un compact, ce qui implique un nombre de paramètres important dés que la dimension p des entrées est grande. Barron (1993) a obtenu un résultat plus intéressant pour une classe de fonctions particulières en supposant que l'on veuille approximer la fonction f par une base finie de réalisations comprenant T individus $(x_i, f(x_i))_{i=1....T}$.

Théorème 2.6. [Barron (1993)] Soit K un sous-ensemble compact de IR^p et f : K → IR une fonction continue sur K. Soit $\tilde{f}_n$ son estimateur de la classe des perceptrons multicouches (Équation 2.8) comportant n neurones sur la couche cachée .Si est T la taille de l'échantillon sur lequel on effectue l'estimation de f .Alors

$$E\left(\left\|f-\tilde{f}_{n,T}\right\|^2\right)\leq O\left(\frac{C^2{}_f}{n}\right)+O\left(\frac{np}{T}\log T\right) \qquad (2.10)$$

où C_f est défini par $C_f=\int_{IR^p}|w|\bar{f}(w)dw$ où $\int_{IR^p}e^{iw^tx}\bar{f}(w)dw=f(x)$ est la représentation de Fourier de f et $|w|_1=\sum_{j=1}^{p}|w_j|$ la norme l_1 de w sur IR^p .
La preuve est basée sur des techniques de Fourier mais le critère de complexité C_f est délicat à manipuler de telle sorte que l'on ne peut pas exprimer les propriétés classiques telles que continuité et dérivabilité avec ce critère.
Nous remarquons que la borne 2.10 nous rappelle un critère du type Akaike et qui nous rend compte des deux aspects contradictoires d'une modélisation non linéaire :

- Minimiser l'erreur d'approximation, qui demande un grand nombre de neurones sur la couche cachée.
- Minimiser l'erreur en généralisation, sur des données nouvelles qui devient grande si le ratio n/T est grand.

c) Autres propriétés du perceptron multicouches Les perceptrons multicouches présentent des caractéristiques ou propriétés particulières qui avec la propriété d'approximation universelle, ont contribué à leur succès :

- **Robustesse à la détérioration** Si un perceptron multicouches à une couche cachée munie d'un nombre suffisant de neurones, supprimer une connexion n'affecte pas le calcul de la sortie puisque les calculs concernent plusieurs neurones et différents chemins de l'entrée vers la sortie existent : on parle de robustesse à la détérioration du modèle.

- **Résistance aux variables d'entrées aberrantes** Si nous sommes dans le cas linéaire, avoir une entrée aberrante implique une prévision aberrante puisque la sortie est linéairement proportionnelle à l'entrée. La propriété particulière des fonctions d'activation sigmoïdes évite cet inconvénient puisque la sortie Y est bornée par $\sum_{j=1}^{n}\left|\alpha_j\right|$.
- **Non unicité du modèle par rapport aux paramètres** Si on permute l'ordre des neurones de la couche cachée on peut avoir les mêmes orties pour deux ensembles de paramètres différents.
- **Echelle des données en entrée et sortie** L'échelle des données n'influe pas sur la Modélisation. En effet, si on a le même modèle qu'en 2.8,et si on fait subir une homothétie et une translation sur les entrées $(X_i)_{i=1,2.....p}$ et la sortie Y :
    - $Y' = Y/a + b \quad a \in IR^*, b \in IR$
    - $X_i' = X_i/c_i + d_i \quad c_i \in IR^*, d_i \in IR \;\; pour \;\; i = 1,2....p$

Alors si on prend le vecteur des paramètres $\theta' = (a\times\alpha_j, \alpha_0 - b, c_j \times \beta_{ij}, \beta_{0j} - d_j)_{1\le i\le p, 1\le j\le n}$, le modèle obtenu est identique à celui de l'équation (2.8) ; c'est une propriété qui nous servira lors de la recherche du meilleur estimateur.

Les perceptrons multicouches munis de fonction d'activation sigmoïde jouissent d'une autre propriété importante concernant la dérivabilité et la contraction :

Proposition 2.1. Soit f une fonction f_θ définie de IR^p dans IR, f_θ de la famille des perceptrons multicouches définies en 2.8 alors f_θ est lipchitzienne et la dérivée de f_θ d'ordre m (m≥1 quelconque) existe et est lipchitzienne.

Démonstration. D'après la définition de f_θ [équation 2.8], f_θ est une combinaison linéaire de fonctions sigmoïdes. Il suffit alors d'appliquer le lemme 2.1 suivant :

Lemme 2.1. Soit σ une fonction définie de IR dans IR, de la famille des fonctions sigmoïdes définies en (2.4), alors σ est lipchitzienne et la dérivée de σ d'ordre m, m ≥ 1 quelconque, est lipchitzienne.

Démonstration. Par définition d'une fonction sigmoïde (2.4) on a :

$$\sigma'_{c,k,r}(x) = \frac{\partial \sigma_{c,k,r}(x)}{\partial x} = 2ck\frac{e^{kx}}{(e^{kx}+1)^2} \tag{2.11}$$

$$= \frac{k}{2c}\left(c^2 - (\sigma_{c,k,r}(x) - r)^2\right) \tag{2.12}$$

L'équation 2.11 donne aussi $\sigma''_{c,k,r}(x) = 2ck^2 e^{kx} \frac{1-e^{kx}}{(e^{kx}+1)^3}$ avec σ''(x)>0 si x<0 et

σ ''(x)< 0 si x>0 .On en déduit que $\forall x, \sigma'(x) \leq \sigma'(0) = (ck)/2$, ce qui implique que $\sigma_{c,k,r}$ est (ck)/2-lipchitzienne. Comme $\sigma'_{c,k,r}(x)$ s'exprime sous forme d'un polynôme en $\sigma_{c,k,r}(x)$ (d'après 2.12) et que $\sigma_{c,k,r}(x)$ est bornée pour tout $x \in IR$, la dérivée de $\sigma_{c,k,r}$ (x) d'ordre m, pour m quelconque existe et est bornée. Puisque pour tout m >1, la dérivée de $\sigma_{c,k,r}$ (x) d'ordre m où $m \geq 1$ est bornée pour tout $x \in IR$, on a que la dérivée de $\sigma_{c,k,r}$ (x) d'ordre m est lipchitzienne.

Dans le paragraphe suivant, on définit le modèle NAR_n (p) basé sur le perceptron multicouches, en commençant par l'introduction des modèles autorégressifs fonctionnels, qui vont nous permettre d'étudier les propriétés statistiques du modèle $NAR_n(p)$.

2.3. Modèles Autorégressifs Fonctionnels (ARF)

Un modèle autorégressif linéaire correspond à l'idée de régression linéaire à chaque instant sur l'espace des observations passées. Ce type de modèle est insuffisant pour la description de certains processus où la relation entre la variable à modéliser et les variables passées n'est pas linéaire. Nous sommes alors tentés d'utiliser une auto régression fonctionnelle (généralement non linéaire), adaptée au phénomène étudié. Ceci aboutit alors à une généralisation du modèle autorégressif classique et qui correspond au modèle associé à l'équation (1.5) du chapitre 1. Ce modèle a été étudié entre autres par Tong (1990), Jones (1994) et Guégan (1994).

Dans ce paragraphe, nous donnons les propriétés générales de ce modèle et les résultats concernant des processus réels de dimension quelconque.

Définition 2.1. Soient deux entiers p, $d \geq 1$. Un processus autorégressif fonctionnel sur IR^d, noté ARF_d (p) est une suite $(X_t)_{t>-p}$ de vecteurs aléatoires vérifiant :

$$X_t = f_\theta(X_{t-1}, \ldots . X_{t-p}) + \varepsilon_t \quad t \succ 0 \qquad (2\text{-}13)$$

où (ε_t) est une suite de bruits blancs indépendants et identiquement distribués et la fonction f est connue. Ici θ est un paramètre appartenant à θ, sous ensemble de IR^s (s entier > 0).

Nous noterons par $X_t^{(p)} = (X_t^{(p)})_{t>0}$ le processus vectorisé associé, défini par :

$$X_t^{(p)} := (X_t, ...X_{t-p+1}) \qquad \text{Pour tout } t > 0.$$

Lorsque p > 1, (X_t) n'est pas une chaîne de Markov; par contre le processus vectorisé $(X_t^{(p)})_{t>0}$ est une chaîne de Markov à valeurs dans $IR^{d\times p}$.

Le processus de contraste des moindres carrés est défini par :

$$U_n(\theta) = \frac{1}{n}\sum_{t=1}^{n}\left\|X_t - f_\theta(X_{t-1}, X_{t-2}, ...X_{t-p}; \theta)\right\|^2 \qquad (2\text{-}14)$$

L'estimateur des moindres carrés est défini pour tout n≥1, par :

$$\hat{\theta}_n := Arg\min_{\theta\in\theta} U_n(\theta) \qquad (2\text{-}15)$$

Nous notons aussi la somme des carrés $S_n(\theta) = nU_n(\theta)$, son gradient DS_n et sa matrice hessienne D^2S_n. (c'est-à-dire la matrice des dérivées secondes de D^2S_n).

Dans ce paragraphe, nous allons étudier les propriétés asymptotiques de $(\hat{\theta}_n)$.

Lorsque la fonction de régression f est linéaire, nous retrouvons le modèle classique $AR_d(p)$ pour lequel on connaît les propriétés de l'estimateur des moindres carrés, voir par exemple Lai et Wei (1983), Hannan et Kavalieris (1986). Si f n'est pas linéaire et dans le cas scalaire (d=1), Klimbo et Nelson (1978) considèrent des processus plus généraux que les $ARF_d(p)$ dans le sens où les bruits ne sont pas indépendants et identiquement distribués et le cas où θ est un ouvert (non nécessairement borné).Ils montrent que sous les conditions :

$$\begin{cases} \bullet \limsup_{n\to\infty}\sup_{\delta\to 0}\frac{1}{n\delta}\sup_{\|\theta-\theta_0\|\le\delta}\left\|D^2S_n(\theta) - D^2S_n(\theta_0)\right\| \prec \infty \quad p.s \\ \bullet \frac{1}{2n}D^2S_n(\theta_0) \xrightarrow{p.s} V \quad avec V \text{ une matrice } s\times s \text{ définie positive} \\ \bullet \frac{1}{n}DS_n(\theta_0) \xrightarrow{p.s} 0. \end{cases} \qquad (2.16)$$

Il existe une suite $(\hat{\theta}_n)$ solution de $DU_n(\theta)=0$, qui converge presque sûrement (p.s) vers θ_0.

La preuve de la consistance quand θ n'est pas nécessairement borné utilise un développement de Taylor d'ordre 2 faisant intervenir le gradient et la Hessienne de $S_n(\theta)$. Lorsque θ est compact, et ceci est le cas qui nous intéresse, la consistance de $(\hat{\theta}_n)$ ne nécessite qu'un bon contrôle du module de continuité de S_n. Si θ est compact, et dans le cas scalaire, Lai (1994)

considère le modèle général de régression stochastique $X_t = f_t(\theta) + \varepsilon_t$. Il a montré que $\hat{\theta}_n$ est fortement consistant sous certaines conditions assez complexes, et la preuve de ce résultat est loin d'être aisée. Mais pour les $ARF_d(p)$ on préfère avoir des conditions suffisantes plus simples. Nous donnons dans ce qui suit pour les modèles $ARF_d(p)$ un ensemble simple de conditions assurant successivement la consistance forte et la normalité asymptotique.
Pour la fonction de régression f, nous exigeons au maximum une régularité C^2 en s'appuyant sur les résultats de la théorie de la stabilité, voir Duflo (1990) ; Meyn et Tweedie ; Duflo (1996). Ces conditions assurent la stabilité de la chaîne vectorisée $(X_t^{(p)})$, et une loi forte des grands nombres pour les fonctionnelles de la chaîne majorée à l'infini par une fonction moment. Dans la suite, nous précisons le type de loi forte des grands nombres pour le modèle $ARF_d(p)$, et les conditions d'application. Nous établissons les premières propriétés du processus $(U_n)_n$ ainsi que la consistance forte de $\hat{\theta}_n$ et enfin sa normalité asymptotique.

2.3.1. Loi forte des grands nombres pour les fonctions non bornées d'un processus AFR_d (p)

La chaîne vectorisée $(X_t^{(p)})$ t > 0 vérifie l'équation itérative suivante :

$$X_t^{(p)} = \begin{pmatrix} X_t \\ X_{t-1} \\ . \\ . \\ .X_{t-p+1} \end{pmatrix} = \begin{pmatrix} f(X_{t-1},...X_{t-p};\theta) \\ X_{t-1} \\ . \\ . \\ .X_{t-p+1} \end{pmatrix} + \begin{pmatrix} \varepsilon_t \\ 0 \\ . \\ . \\ .0 \end{pmatrix} =: F(X_{t-1}^{(p)};\theta) + \eta_t \qquad (2\text{-}17)$$

Avec les définitions implicites correspondantes pour F et η, $(X_t^{(p)})$ t>0 est alors un modèle itératif Markovien. Nous notons par $P_{\theta 0}$ la loi sous le vrai modèle, et toute convergence $\xrightarrow{p.s}$ (respectivement $\xrightarrow{CL}$) signifie la convergence p.s (resp. « en loi ») sous $P_{\theta 0}$, et ceci quelle que soit la loi initiale de la chaîne $(X^{(p)})$. La loi forte des grands nombres (LFGN) pour les fonctions moments d'un ordre suffisant de cette chaîne est l'un des facteurs clés de notre étude. Plus précisément, nous nous plaçons dans le cadre suivant :

2.3.2. Hypothèse [S] de stabilité Nous supposons que la chaîne $(X^{(p)})$ possède sous θ_0 une loi unique invariante $\mu_{\theta 0}$ satisfaisant, pour un $a \geq 1$:

a. Pour tout t et toute loi initiale, $E_{\theta 0}\left|X_t^{(p)}\right|^a \prec \infty$

b. $\mu_{\theta 0}(|.|^a) := \int_{(R^d)^p} |x|^a \mu_{\theta 0}(dx) \prec \infty$

c. Pour toute fonction ϕ de $(IR^d)^p$ dans IR, $\mu_{\theta 0}$ p.s continue, satisfaisant :

$$|\phi(.)| \leq cte(1+|.|^a)$$

Nous avons, pour toute loi initiale, une loi forte des grands nombres, i.e

$$\frac{1}{n}\sum_{t=1}^{n}\phi(X_t^{(p)}) \xrightarrow{p.s} \int_{(IR^d)^p}\phi(x)\mu_{\theta 0}(dx)$$

Les développements sur la stabilité des chaînes de Markov ont motivé cette formulation, Meyn et Tweedie (1993) et Duflo (1996) . Notons que la condition (c) implique la stabilité de la chaîne $X^{(p)}$. Pour un $ARF_d(p)$, nous donnons ci-aprés des conditions suffisantes assurant ce type de loi forte des grands nombres.

Théorème 2.7. Si nous supposons que le modèle $ARF_d(p)$ (2.13) vérifie l'ensemble de conditions (2. 18) ou (2.19) suivantes :

$$\begin{cases} \bullet\ le\ bruit\ (\varepsilon_t)\ a\ un\ moment\ d'ordre\ a \geq 1 \\ \bullet\ il\ existe\ p\ nombres\ positifs\ \lambda_1,....\lambda_p\ tels\ que \lambda_1+....\lambda_p \prec 1 \\ \quad satisfaisant\ pour\ tout\ x, y \in (IR^d)^p \\ \quad \|f(x;\theta_0)-f(y;\theta_0)\| \leq \lambda_1\|x_1-y_1\|+....\lambda_p\|x_p-y_p\| \end{cases} \qquad (2.\ 18)$$

$$\begin{cases} \bullet\ le\ bruit\ (\varepsilon_t)\ a\ une\ densité\ strictement\ positive\ par\ rapport \\ \quad à\ la\ mesure\ de\ Lebesgue, et\ possède\ un\ moment\ d'ordre\ a>1 \\ \bullet\ il\ existe\ p\ nombres\ positifs\ \lambda_1,....\lambda_p tels\ que \lambda_1+....\lambda_p \prec 1 \\ \quad et\ une\ cons\tan te\ k \geq 0\ satisfaisant\ pour\ tout\ x \in (IR^d)^p \\ \qquad \|f(x;\theta_0)\| \leq \lambda_1\|x_1\|+.....\lambda_p\|x_p\|+k \end{cases} \qquad (2.19)$$

Alors, le modèle $AFR_d(p)$ tel que défini par (2.1) sous θ_0 satisfait l'hypothèse de stabilité (S).

Remarques : Ce résultat fournit des critères simples pour une loi forte des grands nombres du type [S] (c).Dans les deux cas (2.18)et (2.19), c'est une extension directe du modèle linéaire $AR_d(p)$.Le système de conditions (2.18) provient de Duflo (1996). Le principal intérêt de ces conditions est qu'elles n'imposent pas au bruit une densité.

2.3.3. Ergodicité

L'ergodicité d'un processus implique que sa moyenne temporelle converge vers sa moyenne statistique, d'où l'existence d'une loi invariante pour le processus considéré, notion utile pour l'utilisation des théorèmes concernant la loi des grands nombres. Si (X t) t≥0 est une suite de variables aléatoires indépendantes identiquement distribuées on sait que $\frac{1}{n}\sum_{t=1}^{n} X_t \xrightarrow{p.s} \mu$ si et seulement si $E(X_1)$ existe et est égale à μ. S'il n' y a pas d'indépendance, ce n'est plus vrai mais dans le cas ergodique la partie si est varie, c'est l'objet du théorème ergodique.

Définition 2.2. Soit (Ω, C, p) l'espace de probabilité d'une chaîne de Markov $(X_t)_{t\geq 0}$. Cette chaîne est dite géométriquement ergodique s'il existe une mesure de probabilité π sur C et un réel $\rho < 1$ telle que

$$\forall x \in \Omega : \left\| P(X_t \in . / X_0 = x) - \pi(.) \right\| = O(\rho^t) \quad \text{pour presque tout x.} \qquad (2\text{-}20)$$

Remarque Soit $(X_{t)\ t\in IN}$ un processus ergodique à valeurs dans IR^d , alors, $\forall g$ fonction intégrable, $g : IR^d \rightarrow IR^d$, le processus $(g(X_t))_{t\in IN}$ est aussi un processus ergodique et pour p>0, le processus $(X_t^{(p)})_{t\in IN}$ est aussi un processus ergodique.

2.3.3.1. Cas du processus ARFd (p) Le théorème suivant donne des hypothèses suffisantes pour que le modèle $ARF_d(p)$ soit géométriquement ergodique.

Théorème 2.8. (Doukhan, 1994 b) Soit le modèle $ARF_d(p)$ défini en (2.13). Sous le système de conditions (2.19), la chaîne vectorielle $(X^{(p)}_t)_{t\geq 1}$ associée au modèle $ARF_d(p)$ est géométriquement ergodique.
L'ergodicité géométrique est une notion propre aux processus Markoviens.

2.3.4. Modèles, estimateur des moindres carrés et fonction de contraste associée

Nous formulons ci-dessous le cadre exact dans lequel nous nous plaçons. Une fonction $g : [0\ \infty[\rightarrow : [0\ \infty[$ est un module de continuité si

- g est croissante
- $\lim_{x\rightarrow 0} g(x) = g(0) = 0$

2.3.4.1. Cadre [M] des modèles étudiés

- **Bruit et paramétrage** Nous considérons une famille de modèles ARF $_d$(p) définis par (2.13) et telle que :

  (a) $(\varepsilon_t)_{t>0}$ est une suite de bruits constituée de variables aléatoires indépendantes et identiquement distribuées à valeurs dans IR ^{d}, centrées indépendante de l'état initial $X_0^{(p)}$ de la chaîne $X^{(p)}$.

  (b) La famille de modèles est identifiée par la famille de fonctions de régression $\{f(.;\theta)\}$ toutes de (IR ^{d}) ^{p} dans IR ^{d}, où le paramètre θ appartient à un compact θ de IR ^{s}, s entier > 0, tel que $\theta \in \overset{0}{\theta}$.

- **Stabilité** Pour le vrai modèle, la fonction de régression f (θ) ainsi que le bruit $(\varepsilon_t)_{t>0}$ satisfont l'hypothèse S de stabilité avec a ≥ 2.

- **Continuité et croissance à l'infini**

  (c) Pour tout θ $x \mapsto f(x;\theta)$ est p.s continue.

  (d) Il existe un module de continuité G tel que $\forall x \in (IR^d)^p, \forall(\alpha,\beta) \in \theta^2$

$$\|f(x;\alpha) - f(x;\beta)\| \le G(\|\alpha - \beta\|)\,(1 + |x|^{\alpha/2})$$

La condition [M]-(d) impose une continuité en θ à x fixe, et une croissance en x majorée par $|x|^{\alpha/2}$.Remarquer que puisque θ est borné, Pour tout α, β :

$G(\|\alpha - \beta\|) \le G(2diam(\theta)) < \infty$

La méthode d'estimation consiste à estimer θ_0 en minimisant une fonctionnelle [U_n (θ)] convenable. Pour des détails, voir les travaux (cas ergodique) Dacunha-Castelle et Duflo (1993) et pour le cas non ergodique, Guyon, (1995) ; Baymog et al (1996).Le résultat qui va suivre concerne l'identification de la fonction de contraste pour le paramètre θ_0 associé au contraste des moindres carrés U_n (2.14).

Proposition 2.2. Dans le cadre [M], nous avons p.s et pour toute loi initiale de $X_0^{(p)}$,

$$\lim_{n\to\infty}[U_n(\theta) - U_n(\theta_0)] = \int_{(IR^d)^p} \|f(x;\theta) - f(x;\theta_0)\|^2 \mu\theta_0(dx) =: K(\theta,\theta_0) \qquad (2.21)$$

De plus, K (θ, θ_0) est une fonction continue en θ.

Démonstration. Notons par $\Delta f_t = f(X_t^{(p)};\theta_0) - f(X^{(p)}{}_t;\theta)$. Nous avons :

$$[U_n(\theta) - U_n(\theta_0)] = \int_{(IR^d)^p} \|f(x;\theta) - f(x;\theta_0)\|^2 \mu\theta_0(dx) =: K(\theta,\theta_0)$$

$$U_n(\theta) - U_n(\theta_0) = \frac{B_n}{n} + \frac{C_n}{n} \quad \text{Avec } B_n = \sum_{0\le t\le n} \|\Delta f_t\|^2, \quad C_n = 2\sum_{0\le t\le n} \langle \varepsilon_{t+1}, \Delta f_t \rangle$$

D'après le [M]-(d), $\|f(x;\theta) - f(x;\theta_0)\|^2 \le \text{cte}\,(1+|x|^a), \quad x \in (IR^d)^p.$ (2-22)

Puisque le modèle sous θ_0 vérifie l'hypothèse [S], la loi forte des grands nombres [S] (d) assure que : $\frac{B_n}{n} \to \int_{(IR^d)^p} \|f(x;\theta) - f(x;\theta_0)\| \mu\theta_0(dx)$.$M_n := C_n/2$ est une martingale de carré intégrable ([S]-(a)).Son crochet $\langle M \rangle_n$ qui vaut :$\langle M \rangle_n = \sum_{0\le t\le n} {}^t\Delta f_t \Gamma \Delta f_t$ tend vers $M_\infty \le \infty$

D'après la loi des grands nombres pour les martingales de carrés intégrables [voir Duflo, (1990)], sur $\{M_\infty = \infty\}$, M_n converge vers une variable finie, et donc M_n/n tend vers 0. Sur $\{M_\infty < \infty\}$, comme $M_n / \langle M \rangle_n$ converge vers 0, il en est de même pour M_n / n. Ainsi, C_n / n tend vers 0 dans tous les cas. D'autre part, l'hypothèse [M]-(d), et l'inégalité (2.22) assurent que $\theta \to K(\theta, \theta_0)$ est continue. θ_0 est clairement un minimum absolu de la fonction K. C'est le seul si le modèle vérifie la condition suivante:

2.3.4.2. Condition d'identifiabilité [D] Le modèle [M] est dit identifiable si :
Pour tout $\theta \in \Theta$, $f(.;\theta) = f(.;\theta_0)$ $\mu_{\theta 0}$–p.s implique que $\theta = \theta_0$.

2.3.5. Consistance forte

Nous allons établir la consistance de l'estimateur des moindres carrés défini par (2.14) et (2.15).

Théorème 2.9. Si les conditions [M] de la section 2.3.4.1 et [D]de la section 2.3.4.2 sont satisfaites, alors l'estimateur des moindres carrés $(\hat{\theta}_n)$ est fortement consistant.

Démonstration Notons par W_n le module de continuité uniforme de U_n, i.e.

$$W_n(\eta) = \sup_{\alpha\beta\in\Theta} [\, U_n(\alpha) - U_n(\beta) \,] \qquad \eta \succ 0$$

Une condition suffisante assurant la consistance forte de $(\hat{\theta}_n)$ est d'après Guyon (1995)

$$P_{\theta 0}\left[\limsup_{n\to\infty} \left\{ W_n\left(\frac{1}{k}\right) \ge \varepsilon_k \right\}\right] = 0 \qquad (2\text{-}23)$$

Notons par $\alpha, \beta \in \Theta$ $\delta(x;\alpha,\beta) := f(x;\alpha) - f(x;\beta)$.D'après [M]-(iii), on a :

$$n|U_n(\alpha) - U_n(\beta)| = \Big| \sum_{0\le t\prec n} \left\langle \delta(X_t^{(p)};\theta_0,\alpha) + \delta(X_t^{(p)};\theta_0,\beta) + \varepsilon_{t+1}, \delta(X_t^{(p)};\alpha,\beta) \right\rangle \Big|$$

$$\leq G(\|\alpha-\beta\|)(1+\left|X_t^{(p)}\right|^{a/2})\sum_{0\leq t\prec n}\left[cte(1+\left|X_t^{(p)}\right|^{a/2})+\|\varepsilon_{t+1}\|\right]$$

$$\leq G(\|\alpha-\beta\|)\sum_{0\leq t\prec n}\left[\frac{1}{2}\|\varepsilon_{t+1}\|^2+cte(1+\left|X_t^{(p)}\right|^{a})\right] \qquad (2.24)$$

Soit S_n la somme dans la dernière inégalité. Par la loi forte des grands nombres appliquée à la suite des variables indépendantes, identiquement distribuées et intégrables ($\|\varepsilon_{t+1}\|^2$) d'une part, à la fonction ($1+|x|^a$) d'autre part, S_n / n tend p.s vers une limite constante $l>0$. D'après (2.24), $W_n(\eta) \leq G(\eta) S_n/n$ .Pour k entier positif, définissons $\varepsilon_k = 2lG(1/k)$. C'est une suite décroissante vers 0. Alors, pour tout k fixe (nous notons i.s pour infiniment souvent),

$$\limsup_{n\to\infty}\left\{W_n(\frac{1}{k})\geq\varepsilon_k\right\}=\left\{W_n(\frac{1}{k})\geq\varepsilon_k\ i.s\right\}\subset\left\{G(\frac{1}{k})(\frac{S_n}{n})\geq\varepsilon_k\ i.s\right\}=\left\{\frac{S_n}{n}\geq 2l\ \ i.s\right\}$$

Sur A : $=\left\{\frac{S_n}{n}\geq 2l\ \ i.s\right\}$, S_n / n ne peut converger vers l; A est donc un événement négligeable. La condition (2-23) est satisfaite, et la consistance forte est alors établie.

2.3.5.1. Normalité asymptotique Le théorème limite central pour $(\hat{\theta}_n)$ nécessite des conditions supplémentaires et usuelles sur la dérivabilité d'ordre 2 du processus de contraste (Un). Si $\varphi(\theta)$ est une fonction scalaire, ses dérivées partielles premières et secondes sont respectivement notées par $D_i\phi=\frac{D\phi}{\partial\theta_i}\ \ et\ \ D^2{}_{ij}=\frac{\partial^2\phi}{\partial\theta_i\partial\theta_j}$, son gradient par $D\phi$ et sa matrice hessienne par $D^2\varphi$. Nous supposons les hypothèses suivantes.

Hypothèse [N] Nous supposons que les conditions du cadre [M] et d'identifiabilité [D] (sections 2.3.4.1 et 2.3.4.2) sont satisfaites, et de plus qu'il existe un voisinage V de θ_0 sur lequel pour tout $x\in(IR^d)^p$, les d fonctions coordonnées $f_1,\ldots..f_d$ de $\theta\rightarrow f(x;\theta)$ sont deux fois continûment dérivables et telles que pour tout k=1,….d et i, j=1,….s, on ait :

(a)Pour tout $\theta\in V$ $x\rightarrow D_i f_k(x;\theta)$ et $x\rightarrow D^2{}_{ij} f_k(x;\theta)$ sont $\mu_{\theta 0}$ p.s continues.

(b)Pour tout $x\in(IR^d)^p$, $\left|D_i f_k(x;\theta_0)\right|\leq cte(1+\left|x\right|^{a/2})$, $\left|D^2{}_{ij} f_k(x;\theta_0)\right|\leq cte(1+\left|x\right|^{a/2})$.

(c) Il existe un module de continuité σ i j k tel que

$$\left|D^2{}_{ij} f_k(x;\theta)-D^2{}_{ij}(x;\theta_0)\right|\leq\sigma_{ijk}(\|\theta-\theta_0\|)(1+\left|x\right|^{a/2}) \qquad \theta\in V \qquad (2.25)$$

Remarquons que la condition [N]-(c) est analogue à [M]- (d) fournissant un contrôle (en x) de la croissance de ces fonctions à l'infini. De même, la compacité de Θ et [N]-(a)-(b) impliquent qu'il existe une constante γ telle que : $\forall\ x\in(IR^d)^p$

$$\forall i,j,k,\quad \forall \theta \in V,\quad \forall x \in (IR^d)^p \qquad \left|D^2{}_{ij} f_k(x;\theta)\right| \le \gamma(1+|x|^{a/2}) \tag{2.26}$$

Nous en déduisons un contraste d'accroissement des dérivées premières :

$$\forall i,j,k,\quad \forall \theta \in V,\quad \forall x \in (IR^d)^p \quad \left|D_i f_k(x;\theta) - D_i f_k(x;\theta_0)\right| \le \gamma \left\|\theta-\theta_0\right\|(1+|x|^{a/2}) \tag{2.27}$$

Et enfin il existe une autre constante γ' telle que : $\forall i,k \quad \forall \theta \in V,\quad \forall x \in (IR^d)^p$

$$\left|D_i f_k(x;\theta)\right| \le \gamma'(1+|x|^{a/2}) \tag{2.28}$$

Posons aussi les matrices $Df(x;\theta) := \left[D_j f_k(x;\theta)\right]_{1\le k\le d,1\le j\le s}$ *matrice* $d\times s$

$$M(x;\theta) := {}^t Df(x;\theta) Df(x;\theta) \qquad \textit{matrice } s\times s$$

(2.29)

$$D^2 f_{ij}(x;\theta) := \left[D^2{}_{ij} f_k(x;\theta)\right]_{1\le k\le d} \quad \textit{vecteur } d\times 1 \;\; 1\le i,\, j\le s \tag{2.30}$$

Les contrôles (2.27) et (2.28) entraînent :

$$\left\|M(x;\theta) - M(x;\theta_0)\right\| \le cte\left\|\theta-\theta_0\right\|(1+|x|^a)\;\; \theta \in V,\quad x \in (IR^d)^p \tag{2.31}$$

$$\left\|M(x;\theta)\right\| \le cte(1+|x|^a)\;\; \theta \in V,\quad x \in (IR^d)^p \tag{2.32}$$

Le vecteur gradient et la matrice Hessienne du contrôle du contraste U_n s'écrivent respectivement :

$$DU_n(\theta) = -\frac{2}{n}\sum_{0\le t\prec n} {}^t\varepsilon_{t+1} Df(X_t{}^{(p)};\theta) \tag{2.33}$$

$$\frac{1}{2}D^2U_n(\theta) = \frac{1}{n}\sum_{0\le t\prec n} M(X_t{}^p;\theta) - \frac{1}{n}\left[\sum_{0\le t\le n} {}^t\varepsilon_{t+1} D_{ij}{}^2 f(X_t{}^{(p)};\theta)\right]_{1\le i,j\le s} \tag{2.34}$$

Nous montrons tout d'abord deux résultats sur $[D\,U_n(\theta_0)]$ et $[D^2U_n(\theta_0)]$.

Proposition 2.3. Nous nous plaçons dans le cadre [N] de la section 2.3.5.1. Nous avons pour toute loi initiale de la chaîne $X^{(p)}$:

$$\text{(a)}\; D^2U_n(\theta_0) \xrightarrow{p.s} I_0 \qquad avec \quad I_0 := 2\int_{(IR^d)^p} M(x;\theta_0)\mu_{\theta 0}(dx) \tag{2.35}$$

$$\text{(b)}\; \sqrt{n}DU_n(\theta_0) \xrightarrow{CL} N(0,J_0) \qquad avec \quad J_0 := 4\int_{(IR^d)^p} {}^t Df(x;\theta_0)\Gamma Df(x;\theta_0)\mu_{\theta 0}(dx) \tag{2.36}$$

Remarque Dans le cas scalaire (d=1), la variance $\Gamma := \sigma^2$ du bruit est scalaire, alors on a $J_0 = 2\,\sigma^2\, I_0$.

Démonstration Partie (a): Pour le premier terme, dans l'expression (2.34) de D^2U_n écrite pour θ_0, le premier terme converge p.s vers la matrice I_0. En effet, la loi forte des grands nombres [S] (c) s'applique d'après le contrôle (2.32) de la fonction matricielle $M(x\,;\theta_0)$.

Pour le deuxième terme, son élément (i,j), $M_n = \sum_{0\le t\prec n} {}^t\varepsilon_{t+1} D^2{}_{ij} f(X_t^{(p)};\theta_0)$ est une martingale de carré intégrable .Son crochet vaut : $\langle M \rangle_n = \sum_{0\le t\prec n} tr\left[\Gamma.(D^2{}_{ij} f\, {}^t D^2{}_{ij} f)(X_t^{(p)};\theta_0)\right.$

Comme nous avons (2.26), en raisonnant de la même façon que la proposition 2.1, nous concluons que M_n/n tend p.s vers 0, d'où la conclusion (i).

Partie (b): Notons : $M_n = -\frac{n}{2} DU_n(\theta_0) = \sum_{0\le t\prec n} {}^t\varepsilon_{t+1} Df(X_t^{(p)};\theta_0)$ (2.37)

C'est une martingale vectorielle, de carré intégrable d'après (2.28) son crochet vaut :

$$\langle M \rangle_n = \sum_{0\le t\prec n} {}^t Df(X_t^{(p)};\theta_0)\,\Gamma\, Df(X_t^{(p)};\theta_0) \tag{2.38}$$

Toujours d'après (2.28) chaque terme de la fonction matricielle

$$x \to J(x;\theta_0) := {}^t Df(X_t^{(p)};\theta_0)\Gamma Df(X_t^{(p)};\theta_0)$$

est majoré (en module) par cte $(1+|x|^a)$.Donc d'après la loi forte des grands nombres [S]-(c),

$$\frac{1}{n}\langle M\rangle_n \xrightarrow{p.s} \int_{(IR^d)^p} J(x;\theta_0)\mu_{\theta 0}(dx) = \frac{J_0}{4} \tag{2.39}$$

Le théorème limite central (ii) sera prouvé si (M_n) satisfait la condition de Lindeberg suivante [cf. Duflo (1990)], corollaire 3.II.11 ou (Hall et Heyde, (1980)) : pour tout ε>0, en notant

$$\Delta_t := M_t - M_{t-1} = \varepsilon_{t+1} Df(X_t^{(p)};\theta_0) \qquad L_n := \frac{1}{n}\sum_{0\le t\prec n} E\left[\|\Delta_t\|^2 \mathrm{II}_{\|\Delta_t\|\ge\varepsilon\sqrt{n}} |F_{t-1}\right] \xrightarrow{P\theta 0} 0 \tag{2.40}$$

Soit A>0 et $F_n(A) := \frac{1}{n}\sum_{0\le t\prec n} E\left[\|\Delta_t\|^2 \mathrm{II}_{\|\Delta_t\|\ge\varepsilon A} |F_{t-1}\right] = \frac{1}{n}\sum_{0\le t\le n} h(X_t^{(p)};A)$ avec

$h(x;A) = E\left[{}^t Df(x;\theta_0)\varepsilon_t\, {}^t\varepsilon_t Df(x;\theta_0)\mathrm{II}_{\{\|Dff(x;;\theta 0)\varepsilon t\|\succ A\}}\right]$

D'après (2.28) nous avons $h(x;A) \le cte(1+|x|^a)$ (2.41)

D'où toujours en vertu de [S]-(c), $F_n(A) \xrightarrow{p.s} \phi(A) := \int_{(IR^d)^p} h(x;A)\mu_{\theta 0}(dx)$

Φ est positive et décroissante. Le théorème de la convergence dominée montre que quand A tend vers ∞, φ (A) tend vers 0. En effet :

- L'équation (2.41) donne la domination puisque $|.|^a$ est $\mu_{\theta 0}$–intégrable ;
- Pour x fixé, posons $\zeta := {}^t Df(X_t^{(p)};\theta_0)\varepsilon_1$. $\|\varepsilon\|^2$ est une variable intégrable, puisque $E\|\varepsilon_1\|^2$ est finie ; alors

$$0 \le h(x;A) = E\left[\|\varepsilon\|^2 \mathrm{II}_{\{\|\varepsilon\|^2\succ A^2\}}\right] \text{ tend vers 0 quand A}\to\infty.$$

à A fixé, nous avons $\varepsilon\sqrt{n} \succ A$ pour n assez grand et $L_n = F_n(\varepsilon\sqrt{n}) \le F_n(A)$, donc presque sûrement $\limsup_n L_n \le \phi(A)$. En faisant tendre A vers ∞, on obtient p.s $\lim_{n\to\infty} L_n = 0$.

La condition de Lindeberg est donc satisfaite et nous avons $M_n / \sqrt{n} \xrightarrow{CL} N(0, J_0 / 4)$

Théorème 2.10. Nous supposons les hypothèses [H] satisfaites, alors pour toute loi initiale de la chaîne vectorisée $X^{(p)}$,

$$\sqrt{n} I_0 \left[\hat{\theta}_n - \theta_0\right] \xrightarrow{CL} N(0, J_0)$$

Démonstration Puisque $\hat{\theta}_n \xrightarrow{p.s} \theta_0$ pour presque tout ω, il existe $n_0(\omega)$ tel que pour $n \ge n_0(\omega)$, $\hat{\theta}_n \in V$ et nous avons d'après la formule de Taylor avec reste intégrale :

$$0 = DU_n(\hat{\theta}_n) = DU_n(\theta_0) + \Delta_n(\hat{\theta}_n)(\hat{\theta}_n - \theta_0) \qquad (2.42)$$

(avec $\Delta_n(\hat{\theta}_n) = \int_0^1 D_\theta{}^2 U_n\left[\hat{\theta}_n + u(\hat{\theta}_n - \theta_0)\right] du$

Supposons vérifiée la condition suivante (cf . lemme 2.2) :

$$\Delta_n(\hat{\theta}_n) - D_\theta{}^2 U_n(\theta_0) \xrightarrow{P\theta 0} 0 \qquad (2.43)$$

et compte tenu de la proposition 2.3, nous avons le résultat énoncé.

Lemme 2.2. Si les conditions du théorème 2.8 sont satisfaites, nous avons :

$$\Delta_n(\hat{\theta}_n) - D_\theta{}^2 U_n(\theta_0) \xrightarrow{p.s} 0 \qquad \Delta_n(\hat{\theta}_n) \xrightarrow{p.s} I_0 \qquad (2.44)$$

Démonstration Pour $\theta \in V$, notons (k_j) une suite de constantes positives, nous avons d'après (2.26), (2.31) et (2.34) : $\frac{n}{2}\left\| D_\theta{}^2 U_n(\theta) - D_\theta{}^2 U_n(\theta_0)\right\|$

$$= \left\| \sum_{0\le t\prec n} \left[M(X_t^{(p)};\theta) - M(X_t^{(p)};\theta_0)\right] - \sum_{0\le t<n} {}^t\varepsilon_{t+1}\left[D^2{}_{ij} f(X_t^{(p)};\theta) - D^2{}_{ij} f(X_t^{(p)};\theta_0)\right]_{1\le i,j\le s}\right\|$$

$$\le k_1\|\theta - \theta_{01}\| \sum_{0\le t\prec n}(1 + \left|X_t^{(p)}\right|^a) + k_2 \sum_{1\le i,j\le s, l\le k\le d} \sigma_{ijk}(\|\theta - \theta_0\|) \sum_{0\le t<n} \|\varepsilon_{t+1}\|(1 + \left|X_t^{(p)}\right|^a)$$

$$\le k_3\sigma(\|\theta - \theta_{01}\|) \sum_{0\le t<n} \|\varepsilon_{t+1}\|^2 + k_4\left[\|\theta - \theta_0\| + \sigma(\|\theta - \theta_0\|)\right] \sum_{0\le t\prec n}(1 + \left|X_t^{(p)}\right|^a),$$

Où nous avons noté $\sigma(z) := \sum_{i,j,k} \sigma_{ijk}(z)$ .D'autre part, d'après (2.42) :

$$\left\|\Delta_n(\hat{\theta}_n) - D_\theta{}^2 U_n(\theta_0)\right\| = \left\| \int_0^1 \left\{D_\theta{}^2 U_n\left[\hat{\theta}_n + u(\hat{\theta}_n - \theta_0)\right] - D_\theta{}^2 U_n(\theta_0)\right\} du \right\|$$

$$\le 2k_3\sigma(\left\|\hat{\theta}_n - \theta_0\right\|)\frac{1}{n}\sum_{0\le t<n}\|\varepsilon_{t+1}\|^2 + 2k_4\left[\|\theta - \theta_0\| + \sigma(\|\theta - \theta_0\|)\right]\frac{1}{n}\sum_{0\le t\prec n}(1 + \left|X_t^{(p)}\right|^a),$$

et comme les deux séries convergent p.s et $\hat{\theta}_n \xrightarrow{p.s} \theta_0$, alors $\Delta_n(\hat{\theta}_n) - D_\theta{}^2 U_n(\theta_0) \xrightarrow{p.s} 0$

Le résultat est alors une conséquence de la proposition 2.3.

Dans le paragraphe suivant, nous aborderons le modèle neuronal ; nous essayons de dégager ses traits particuliers, nous reprenons les conditions associées aux propriétés statistiques décrites ci dessus pour les adapter au modèle de neurones.

2.3.6. Le modèle paramétrique NAR_n (p) basé sur le perceptron multicouches

Un modèle autorégressif linéaire s'avère inefficace dés qu'on veut décrire certains processus pour lesquels la relation entre la variable à modéliser à l'instant t et les variables passées n'est pas linéaire. On peut alors s'intéresser à l'utilisation d'une auto régression basée sur le perceptron multicouches et adaptée au phénomène étudié. La propriété d'approximation universelle de ce dernier quand il est muni d'une seule couche cachée a favorisé l'étude d'un modèle basé sur ce type de perceptron multicouches, on est ainsi conduit à une extension non linéaire du modèle autorégressif classique.

Définition 2.3. Dans le cadre du modèle de perceptron multicouches défini équation 2.8 et si n et p sont deux entiers p, n≥1, un processus autorégressif fonctionnel sur IR^p, dont la fonction associée est de la classe des perceptrons multicouches à une seule couche cachée munis de fonctions de transfert σ sigmoïdes est une suite $(X_t)_{t\in IZ}$ de variables aléatoires à valeurs dans IR vérifiant

$$X_t = f_\theta(X_{t-1}^{(p)}) + \varepsilon_t = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ji} X_{t-i} + \beta_{j0}) + \alpha_0 + \varepsilon_t \qquad (2.45)$$ où

- $\theta = \{(\alpha_j)_{0\le j\le n} (\beta_{ji})_{0\le i\le p, 1\le j\le n}\} \in IR^{n\times(p+2)+1}$ est le vecteur des paramètres.
- (ε_t) est un bruit (indépendant identiquement distribué).

Par la suite, ce modèle est appelé NAR_n (p) (neural autoregression); le processus vectorisé qui lui est associé est noté $X_t^{(p)} = (X_t^{(p)})_{t\in IZ}$ où $X_t^{(p)} := (X_t, ...X_{t-p+1})$, (X_t) n'est pas une chaîne de Markov (p>1) mais $(X_t^{(p)})$ l'est (à valeurs dans IR^{dp}).Ainsi, nous sommes en présence d'un modèle ARF_1 (p) particulier décrit au paragraphe 1.

Remarques

- Au modèle $NAR_n(p),n)$, est associé un réseau multicouches à une seule couche cachée ayant n neurones et une couche constituée de $(X_{t-1}, X_{t-2}, ...X_{t-p})$ et une seule sortie.
- Nous pouvons définir des modèles incluant d'autres variables aléatoires dites variables explicatives ou exogènes, et /ou certains retards du bruit (ε_t), ainsi nous pouvons avoir :
  - Le modèle autorégressif neuronal avec variables exogènes ($NARX_n(p, m)$)

$$X_t = = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ji} X_{t-i} + \sum_{l=1}^{m} \beta'_{ji} Y^l{}_t + \beta_{j0}) + \alpha_0 + \varepsilon_t \qquad (2.46)$$

  - Le modèle autorégressif neuronal avec moyenne mobiles ($NARMA_n(p,q)$

$X_t = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ji} X_{t-i} + \sum_{k=1}^{q} \beta''_{jk} \varepsilon_{t-k} + \beta_{j0}) + \alpha_0 + \varepsilon_t$ où les $(\varepsilon_{t-k})_{k=1\ldots q}$ sont les q résidus passés.

  - Le modèle autorégressif neuronal avec moyenne mobiles et variables exogènes

NARMAX (p,q,m) $X_t = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ji} X_{t-i} + \sum_{l=1}^{m} \beta'_{jl} Y^l{}_t + \sum_{k=1}^{q} \beta''_{jk} \varepsilon_{t-k} + \beta_{j0}) + \alpha_0 + \varepsilon_t$

- Le modèle NAR_n et ses variantes ne peuvent pas concerner de processus incluant des tendances puisque les fonctions d'activation sont des fonctions sigmoïdes bornées.

2.3.6.1. Correspondance avec le modèle ARMA et dérivés Si dans (2.45) $\alpha_i = 0$ i=0…l et $\Psi(x) = x$ (l'identité), le modèles $NAR_0(p)$ est un modèle autorégressif AR (p), à l'instant t, X_t est fonction linéaire de ses valeurs passées et d'une perturbation aléatoire ε_t. Si (β_0, β_1, .. β_p) sont les p+1 réels associés au modèle X_t est de la forme :

$$X_t = \sum_{i=1}^{p} \beta_i X_{t-i} + \beta_0 + \varepsilon_t \qquad t \in Z \qquad (2.47)$$

où Z est l'ensemble des entiers relatifs.

Correspondances

Si on remplace les fonctions d'activation sigmoïdes des neurones par des fonctions linéaires, le modèle neuronal devient un modèle linéaire classique mais s'il y a des couches cachées, celui-ci est surparamètré. Les fonctions d'activation sigmoïdes utilisées telles que la fonction logistique ou la fonction tangente hyperbolique peuvent être approchées au voisinage de zéro par une fonction linéaire. Ceci peut être montré par un petit développement limité au voisinage de zéro tel que ci après :

Pour la fonction logistique, $\sigma(x)= 1 / (1+e^{-x})$ $\lim_{x\to\infty} \sigma(x) = 1$ $\lim_{x\to-\infty} \sigma(x) = 0$ .Pour x très petit, $\sigma(x) \approx \sigma(0) + \sigma'(0).x \approx \frac{1}{2} + 1/4\ x$. Donc $\sigma(x)$ est presque linéaire. Ceci implique que les réseaux à fonction d'activation la fonction logistique ou tangente hyperbolique et utilisant des poids assez petits, de sorte que l'excitation totale arrivant aux nœuds est petite, agissent presque comme des systèmes linéaires. On a le lemme :

Lemme 3.2. Soient T et n deux entiers non nuls et soit $\phi = (\phi_0, \phi_1, ...\phi_p)$ un vecteur de p réels. Alors pour toute suite $(X_t)_{t=1,2,...T}$ extraite d'un processus autorégressif linéaire AR (p) de vecteur de paramètres φ et de vecteur initial $X_0^{(p)}$, et pour tout η >0, il existe un processus NAR_n (p) noté $(Y_t)_{t\geq 1\text{-}p}$, de vecteur de paramètres $\theta=\{(\alpha_j)_{0\leq j\leq n}(\beta_j i)_{0\leq i\leq p, 1\leq j\leq n}\} \in IR_{n\times(p+2)+1}$ et de même vecteur initial $X_0^{(p)}$, tel que, pour tout $t \in \{1,2,...T\}$, $|X_t - Y_t| \prec \eta$

Démonstration. Nous commençons par le cas simple où n=p=1 (un processus AR (1) est un modèle NAR_1 (1)), donc une seule entrée et une seule unité cachée ; la démonstration peut être alors étendue au cas où n et p sont des entiers n, p >1.Donc pour ε>0 on doit montrer que pour tout compact K de IR et pour tout $\gamma \in IR$ *il existe* $\alpha, \beta \in IR$ *tels que*

$$|\alpha \tanh(\beta x) - \gamma x| \prec \varepsilon \qquad (2.48)$$

Le développement limité de x → tanh(x) au voisinage de 0 est tanh (x)=x+O(x^3).Il existe donc une constante C >0 et un voisinage S de 0 tels que pour tout $x \in S$ | tanh (x)-x | ≤ C x^3.D'autre part, quelque soit le compact K choisi, il existe β non nul tel que $\beta K = \{\beta x,\ x \in K\} \subset S$ on peut alors écrire pour tout $x \in K$, et pour tous α et γ :

$$|\alpha \tanh(\beta x) - \gamma x| = |\alpha \tanh(\beta x) - \alpha\beta x + \alpha\beta x - \gamma x| \leq |\alpha\beta x - \gamma x| + C|\alpha| \,.\, |\beta x|^3 \qquad (2.49)$$

Posons $A = |\alpha\beta x - \gamma x|$ .on peut prendre α tel que αβ= γ (α= γ/β), pour tout $x \in K$ on a alors A=0, et l'inégalité 2.49 peut s'écrire : $|\alpha \tanh(\beta x) - \gamma x| \leq C|\gamma|\beta^2|x|^3$ (2.50)

En posant M= $\max_{x\in K}|x|$, on a alors $|\alpha \tanh(\beta x) - \gamma x| \leq C|\gamma|\beta^2 M^3$

(2.51)

Alors si on choisit β suffisamment petit pour que $\leq C|\gamma|\beta^2 M^3 < \varepsilon$ on obtient l'inégalité (2.48).

2.3.6.2. Propriétés probabilistes du processus NAR_n (p) Cette partie reprend les conditions pour que le modèle NAR_n (p) possède des propriétés de stabilité qui est utile pour

obtenir la consistance et la normalité asymptotique de l'estimateur des moindres carrés des paramètres de ce modèle.

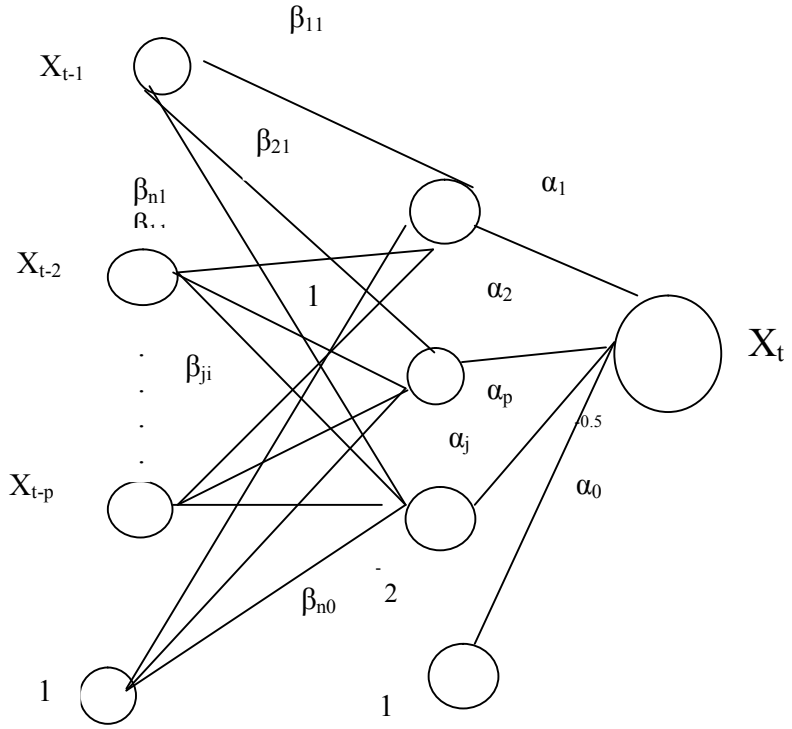


Fig 2.16 Le modèle NAR_n (p) avec les mêmes notations que l'équation 2.45

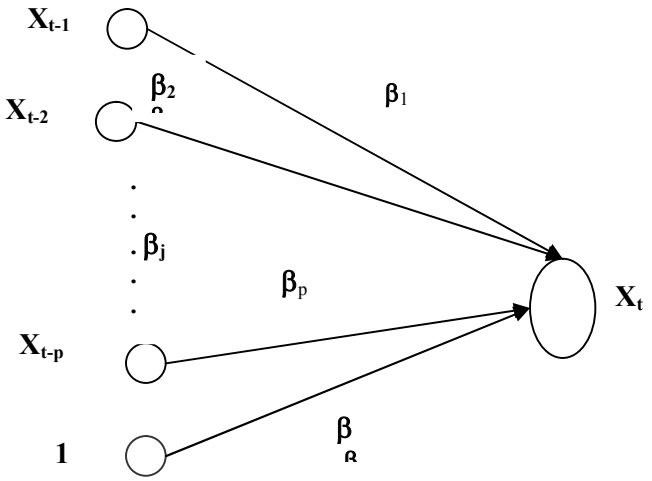


Fig 2.17 Le modèle AR (p) avec les notations de 2.47

a) Stabilité du processus NAR_n (p) Nous considérons dans cette partie le modèle du type perceptrons multicouches défini en 2.8 tel que les fonctions d'activation associée aux neurones de la couche cachée sont des sigmoïdes. Le but est de trouver les conditions dans lesquelles le modèle est stable. Pour cela, nous reprenons les théorèmes concernant le cas général des modèles autorégressifs fonctionnels quelconques ARF_d (p), nous les appliquerons au modèle NAR_n (p) qui est un modèle ARF_1 (p).

Théorème 2.11. Soit le modèle NAR_n (p) défini en (2.45), avec

$$f_\theta(X_{t-1}^{(p)})=\sum_{j=1}^{n}\alpha_j\sigma_{c,k,r}(\sum_{i=1}^{p}\beta_j iX_{t-i}+\beta_{j0})+\alpha_0$$

où $\sigma_{c,k,r}$ est une fonction sigmoïde et θ_0 est le vrai paramètre $\theta_0 = \left\{(\alpha^*{}_i)_{0\le i\le n}(\beta^*{}_{ij})_{0\le i\le p,1\le j\le n}\right\}$

Supposons que pour a≥1, l'un ou l'autre ensemble de conditions suivantes soit satisfait :

$$\begin{cases}(a)\ le\ bruit(\varepsilon_t)\ a\ un\ moment\ d'ordre\ a;\\ \\ (b)\ \sum_{j=1}^{n}\sum_{i=1}^{p}\left|\alpha_j{}^*\beta_{ji}{}^*\right| \prec \frac{2}{ck};\end{cases} \qquad (C.1)$$

$$\begin{cases}(a)\quad le\ bruit\ (\varepsilon_t)\ a\ une\ densité\ strictement\ positive\ par\ rapport\\ \qquad à\ la\ mesure\ de\ Lebesgue;\\ \\ b)\quad le\ bruit\ (\varepsilon_t)\ possède\ un\ moment\ d'ordre\ a+\gamma\ \ pour\ \ un\ \gamma>0\end{cases} \qquad (C.2)$$

Alors, le modèle $NAFR_n$ (p) est stable.

Démonstration. Le modèle NAR_n(p) est en fait un modèle ARF_1 (p) particulier .Si nous revenons alors au paragraphe 2 nous voyons que les conditions (C.1)et (C.2) impliquent respectivement (2.18) et (2.19) avec d=1 du théorème 2.7 pour le vrai modèle en θ_0
Par définition $\sigma_{c,k,r}$ est bornée par |c| + |r|, ce qui implique que pour un vecteur de paramètres θ fixé, f_θ (.) est bornée d'où les conditions (2.19) sont satisfaites. D'autre part, le lemme 2.1 dit que $\sigma_{c,k,r}$ est (ck)/2 lipchitzienne. Il suffit donc d'avoir (b), pour que la condition (2.18) exigeant que f soit contractante, soit remplie.

b) Ergodicité du processus NAR_n(p) Nous reprenons le cadre et les définitions sur les propriétés d'ergodicité du paragraphe 1 pour les processus ARF_d (p), avec ici d=1.

Théorème 2.12. Soit le modèle NAR_n (p) défini en (2.45) pour t ≥ 1-p, de vecteur de paramètres initial $X_0^{(p)}$ et tel que $f(X_{t-1}^{(p)}) = \sum_{j=1}^{n}\alpha_j\sigma_{c,k,r}(\sum_{i=1}^{p}\beta_{ji}X_{t-i}+\beta_{j0})+\alpha_0$ et $\sigma_{c,k,r}$ une fonction du type sigmoïde .Notons θ_0 le vrai paramètre $\theta_0 = \left\{(\alpha^*{}_i)_{0\le i\le n}(\beta^*{}_{ij})_{0\le i\le p,1\le j\le n}\right\}$. Si pour a≥1, l'ensemble de conditions [C.2] du théorème 2.11 est satisfait, alors le processus $(X_t)_{t\ge 1-p}$, est géométriquement ergodique quelque soit la loi initiale de $X_0^{(p)}$.

Démonstration. D'après le théorème 2.8, il suffit de remplir les conditions du système (2.19) avec d=1 pour que le modèle soit géométriquement ergodique. Or nous venons de voir au théorème 2.11 que le système de conditions [C.2] implique le système de conditions (2.19).

2.3.6.3. Problème de la prévision d'ordre supérieur à 1 Dans le cas linéaire, il a été montré que la meilleure prévision à k pas, k≥1 d'une série linéaire quelconque $(X_t)_{t>-p}$ par un modèle AR est calculée par récurrence comme une prévision à un pas, où on remplace les réalisations précédentes par leurs prévisions.

Soit un processus autorégressif telle que : $X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \varphi_0 + \varepsilon_t$, on a :

$\hat{X}_t = E\left(X_t \middle| X_{t-1}, X_{t-2}, \ldots X_{t-p}\right)$ $\hat{X}_{t+1} = E\left(X_{t+1} \middle| X_{t-1}, X_{t-2}, \ldots X_{t-p}\right) =$

$E\left(\sum_{i=1}^{p} \phi_i X_{t-i+1} + \varphi_0 + \varepsilon_t \middle| X_{t-1}, X_{t-2}, \ldots X_{t-p}\right)$

$= \varphi_1 E\left(X_t \middle| X_{t-1}, X_{t-2}, \ldots X_{t-p}\right) + \sum_{i=2}^{p} \phi_i X_{t-i+1} + \varphi_0 + E(\varepsilon_t) = \varphi_1 \hat{X}_t + \sum_{i=2}^{p} \varphi_i X_{t-i+1} + \varphi_0$ (ε_t centré).

Si f_{AR} désigne la fonction relative à un modèle autorégressif : $f_{AR}(x_0, x_1, \ldots x_p) = \sum_{i=0}^{p} \varphi_i x_i$, on calcule alors de façon optimale la prévision de X_{t+k}, k≥0 de la manière suivante :

$$\hat{X}_t = f_{AR}(X_{t-1}, X_{t-2}, , \ldots X_{t-p})$$
$$\hat{X}_{t+1} = f_{AR}(\hat{X}_t, X_{t-1} \ldots X_{t-p+1})$$
(2.52)
$$\hat{X}_{t+k} = f_{AR}(\hat{X}_{t+k-1}, \hat{X}_{t+k-2}, \ldots \hat{X}_{t+k-p}) \qquad \text{Si } k>p$$

Ce processus itéré un nombre de fois infini entraîne une convergence de la série $(\hat{X}_t)_{t\geq 0}$ ainsi générée vers sa moyenne. Mais dans le cadre non linéaire associé aux perceptrons multicouches, si nous voulons procéder de la même manière en remplaçant dans le système d'équations (2.52) la fonction f_{AR} par une fonction de la classe des perceptrons multicouches, cette propriété n'est plus vérifiée (fig. 2.18) . Ainsi, dés que l'on injecte en entrée la sortie d'un perceptron, la fonction récurrente non linéaire peut posséder un ou plusieurs points d'attraction et un /ou plusieurs points de répulsion. La valeur limite de la série ainsi constituée dépend donc du point de départ de la prévision à k pas qui est le départ de la récurrence.

Par exemple, pour un perceptron multicouches muni de deux neurones sur la couche cachée (voir Fig.2.19). F : IR $\longrightarrow$ IR tel que : x $\longrightarrow 2\tanh(-6x)+3\tanh(3x)$, on peut calculer les points selles (tels que f(x)=x). F possède deux points d'attraction (a_1 = -0.9837 et puisque F est impaire, a_2 = 0.9837) et trois points de répulsion (r_1 = - 0.3215, r_2=0 et r_3=0.3215). Au vu des figures 2.21 et 2.22, la fonction de récurrence associée à ce modèle peut converger vers l'un des points d'attraction a_1 ou a_2 ou avoir un comportement chaotique et osciller entre les points de répulsion r_1 ou r_2 suivant la valeur du point de départ de la récurrence :

$$\lim_{n\to\infty} \hat{X}_{t-n} = \begin{cases} a_1 \ si \ X_t < r_1 \\ a_{12} \ si \ X_t > r_2 \end{cases} \qquad \forall\, n \geq 1,\ \hat{X}_{t+n} \in]r_1\ , r_2[\ si \ X_t \in]r_1\ , r_2[$$

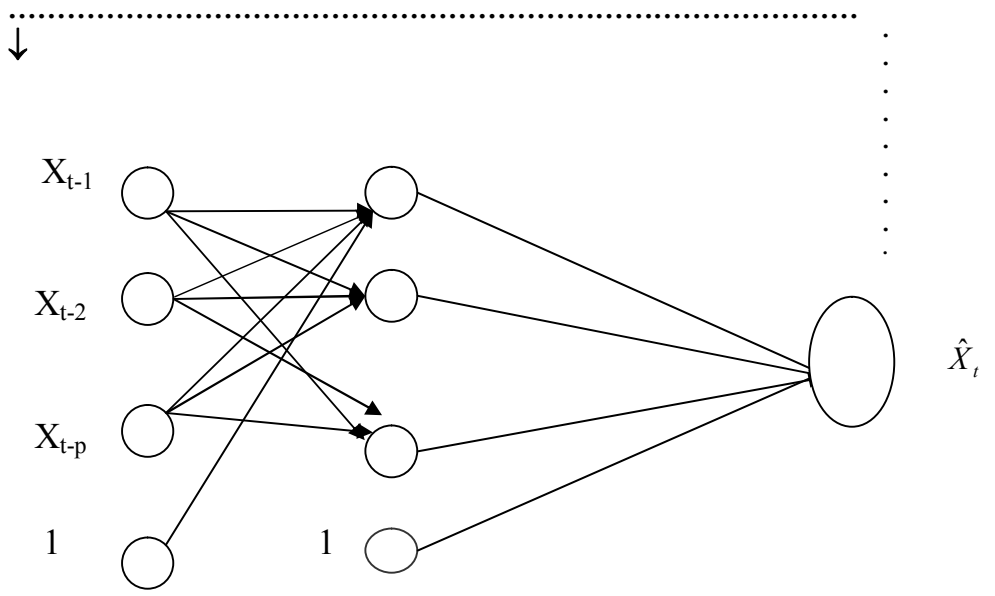


Fig.2.18 Réinjection de la sortie vers l'entrée

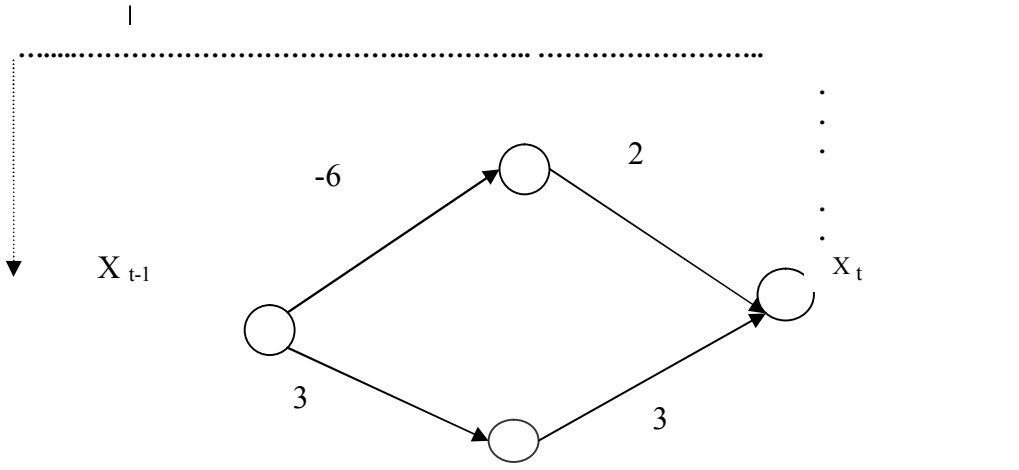


Fig.2.19 Perceptron multicouches de l'exemple

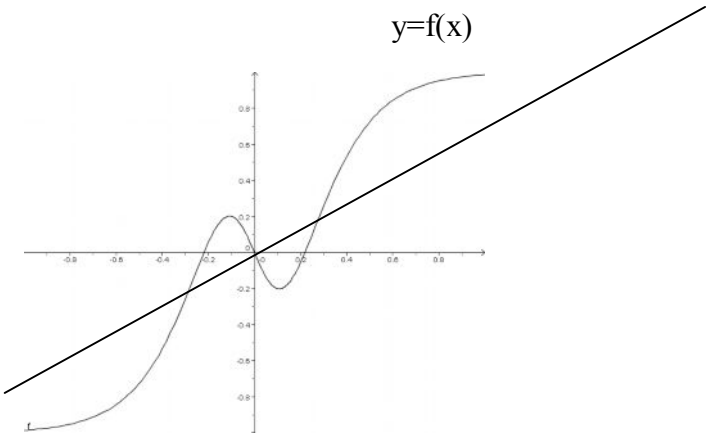


Fig 2.20 graphe de la fonction x→ 2 tanh (-6x)+3tanh(3x) Les valeurs de x vérifiant x=f(x) sont {-0.9837, 0.32150.98370.32150}

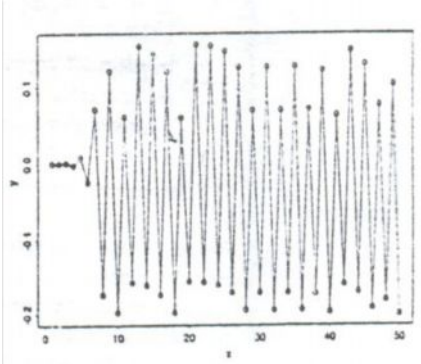


Fig 2.21 point de départ : 0.33

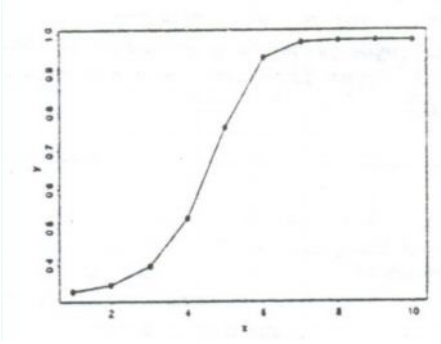


Fig 2.22 point de départ : 0.001

Dans le paragraphe suivant, nous allons justement aborder le problème de la prévision dans le cadre neuronal en relation étroite avec le problème d'estimation appelée apprentissage dans ce cas, et qui va en constituer la partie la plus importante, avec l'introduction de plusieurs méthodes et algorithmes d'optimisation.

2.4. Estimation des paramètres d'un modèle neuronal

2.4.1. Introduction

Dans plusieurs problèmes de prévision, nous avons à étudier des séries chronologiques, qui dépendent non seulement des valeurs passées mais aussi de certaines variables exogènes explicatives. Par exemple, si on considère la consommation d'électricité dans une ville durant une journée ou une autre période, celle ci dépend non seulement de la consommation des journées précédentes mais aussi d'autres variables météorologiques comme la température et de la nature du jour (jour de semaine, week-end, vacances…etc.).

Nous nous intéressons aux séries chronologiques qui peuvent être vues comme des modèles autorégressifs non linéaire généralisés avec variables exogènes (ARX).Nous cherchons une représentation de ces processus, définis par l'équation de récurrence :

$$X_t = f_\theta \left(X_{t-1}, X_{t-2}, ...X_{t-p},\ Y_t\right) + \varepsilon_t \qquad (2.53)$$

où

- (Y_t) est une suite à valeurs dans IR^q appelée variable exogène.
- f_θ une fonction généralement non linéaire :$IR^{p+q} \to IR$
- θ le vecteur des paramètres ($\in IR^m$) du modèle.
- (ε_t) est une suite de variables aléatoires indépendantes de même distribution, de moyenne nulle et de variance σ^2 telle que ε_t est indépendant du passé X_s, $s<t$

L'équation (2.53) définit une classe large et générale. Le choix de la fonction f_θ est justifié par les résultats classiques concernant les capacités d'approximation des perceptrons multicouches, voir [G.Cybenko, K.Funahashi, K.Hornik, M.Stinchcombe, H.White et M.Leshno].Soit alors R (d,k,ψ) l'ensemble de toutes les fonctions de IR^d dans R qui peuvent être associées à un perceptron multicouches, avec d unités d'entrées linéaires, k unités cachées de fonction d'activation ψ continue et localement bornée et une unité de sortie linéaire. Alors pour chaque ensemble compact B de IR^d ,chaque fonction continue définie sur B peut être uniformément approchée par des éléments de R (d, k,ψ), si nous augmentons le nombre k des unités cachées. De ces résultats, l'idée est de considérer la classe des fonctions f_θ associées à un perceptron multicouches avec une couche cachée non linéaire de fonction d'activation ψ, une couche d'unités d'entrées avec $\left(X_{t-1}, X_{t-2}, ...X_{t-p}, Y_t\right)$ comme entrées et une unité de sortie linéaire avec X_t comme valeur désirée. Dans cette partie, nous présentons des résultats où la variable exogène est omise, cependant, ces derniers peuvent être généralisés aux modèles ARX.(voir la partie pratique).Nous voulons donc modéliser à l'aide d'un modèle neuronal un phénomène chronologique quelconque. Ce phénomène peut s'écrire sous une forme auto régressive fonctionnelle :

$$\forall t \in IZ, \quad X_t = f\left(X_{t-1}^{(p)}\right) + \varepsilon_t \qquad (2.54)$$

avec pour $p \geq 1$ $X_{t-1}^{(p)} = (X_{t-1}, X_{t-2}, ...X_{t-p})$, où la fonction f est approchée par un modèle neuronal de structure définie donnée par la fonction f_θ qui est de la classe des perceptrons multicouches où $\theta \in IR^l$ est le vecteur des paramètres. Pour estimer les paramètres $(\theta_i)_i$ qui

rendent compte du phénomène, on utilise la méthode des moindres carrés, connue dans le cas de la régression et l'autorègression. C'est donc bien un problème d'optimisation.

Avant d'avancer, commençons par donner un bref rappel des résultats sur l'estimation dans un modèle auto régressif (AR) linéaire classique.

2.4.2. Estimation dans le cas linéaire

Soit le processus auto régressif linéaire (X_t) représenté par l'équation (voir Fig 2.23) :

$$X_t = \sum_{i=1}^{p} \beta_i X_{t-i} + \beta_0 + \varepsilon_t \qquad t \in Z \tag{2.55}$$

Supposons les hypothèses (S) suivantes vérifiées :

- Le processus (X_t) est un processus stationnaire de second ordre.
- Les racines de l'équation polynomiale $\sum_{i=0}^{p} \beta_i z^{p-i} = 0$ avec $\beta_0 = -1$ sont à l'intérieur du cercle unité (ceci implique que ε_t est indépendant du passé du processus).

Il est clair que (2.55) peut être associé à un réseau de neurones linéaire (ADALINE) comme à La fig 2.23 avec p entrées, une entrée supplémentaire fixée à 1 pour le biais β_0, pas de couche cachée et une seule unité de sortie. Les coefficients (β_0, β_1, .. β_p,) sont les poids de connexion. Etant donnés T+p observations (T+p valeurs successives de la série $(X_t)_{1 \le t \le T+p}$), l'estimation consiste à minimiser la somme des résidus carrés (appelée fonction de coût ou fonction erreur) $S_T(\beta) = \sum_{t=p+1}^{T+p} (X_t - \hat{X}_t)^2$ où $\beta = (\beta_0, \beta_1, .. \beta_p)$ et $\hat{X}_t = \sum_{i=1}^{p} \beta_i X_{t-i} + \beta_0$.

C'est la méthode de Box-Jenkins des moindres carrés conditionnels. Elle est appelée conditionnelle, parce qu'elle est conditionnée par les p premières valeurs du processus (X_t). Si Σ désigne la matrice (p x p) d'auto covariance, avec $(\Sigma)_{ij} = \mathrm{Cov}(X_{t+i}, X_{t+j}) = \gamma_{|i-j|}$, i.e

$$\Sigma = \begin{bmatrix} \gamma_0 & \gamma_1 & & \gamma_{p-1} \\ \gamma_1 & \gamma_0 & & \gamma_{p-2} \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_{p-1} & \gamma_{p-2} & \cdots & \gamma_0 \end{bmatrix}$$

Rappelons les formules pour estimer β, σ^2 et Σ :

Sans perte de généralité, nous pouvons soustraire $E(X_t)$ de X_t. Donc $E(X_t) = 0$, $\beta_0 = 0$ et $\gamma_h = \mathrm{Cov}(X_t, X_{t+h}) = E(X_t X_{t+h})$.Utilisant une écriture vectorielle où A^T désigne la

transposée d'une matrice ou d'un vecteur A, définissons $x = (X_{p+1} , X_{p+2} , \ldots X_{p+T})^T$, $\varepsilon = (\varepsilon_{p+1} , \varepsilon_{p+2} , \ldots \varepsilon_{p+T})^T$ et $\beta = (\beta_1 , \beta_2 , \ldots \beta_p)^T$, de sorte que

$$\mathbf{x = X \beta + \varepsilon} \qquad (2.56)$$

Alors l'estimateur des moindres carrés pour β est $\hat{\beta} = (X^T X)^{-1} X^T x$.

où

$$X_{(T\times p)} = \begin{bmatrix} X_p & X_{p-1} & & X_1 \\ X_{p+1} & X_p & & X_2 \\ \ldots & \ldots & \ldots & \ldots \\ X_{p+T-1} & X_{p+T-2} & \ldots & X_T \end{bmatrix}$$

Nous remarquons que cet estimateur est le même que celui de la régression. La variance σ^2 est estimée par la somme moyenne des résidus carrés comme d'habitude. Il s'ensuit que :

$$\hat{\sigma}^2 = \frac{1}{T} S_T(\hat{\beta}) = \frac{1}{T} \sum_t (X_t - \hat{X}_t)^2$$

Comme $E (X^T X) = T \Sigma$, on prend $\hat{\Sigma} = \frac{1}{T} X^T X$ comme un estimateur de Σ. L'estimateur $\hat{\beta}$ des paramètres est celui des moindres carrés dont on connaît les propriétés : étant données les hypothèses S, on peut montrer avec le théorème central limite que $\hat{\beta}$ est asymptotiquement gaussien, plus précisément : $\sqrt{T}(\hat{\beta} - \beta) \xrightarrow{D} N_p(0, \sigma^2 \Sigma^{-1})$ quand T tend vers l'infini.

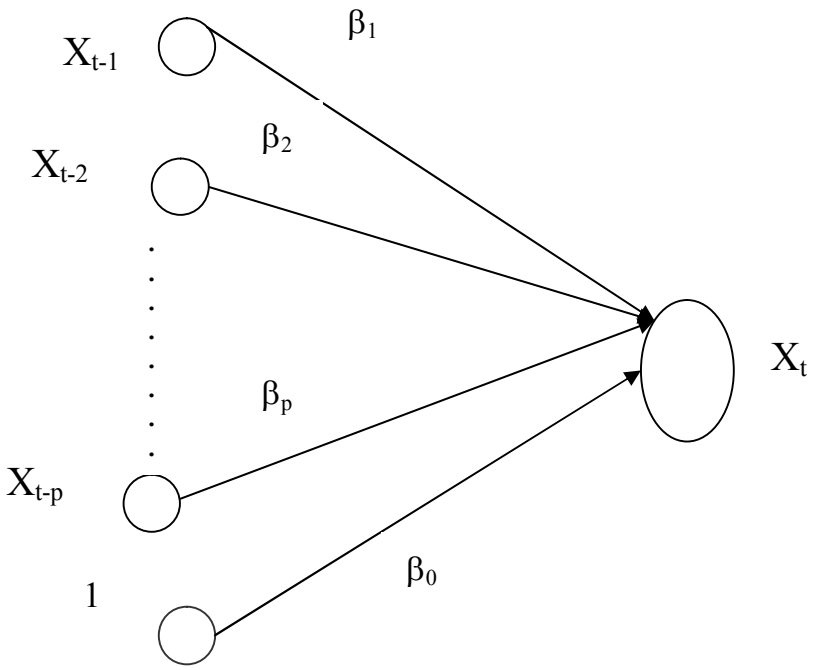


Fig 2.23 Réseau de neurones représentant un processus auto régressif linéaire

Remarques

- $\frac{\partial^2 S}{\partial \beta_i \partial \beta_j} = 2\sum_t X_{t-j} X_{t-i}$ donc la Hessienne de la fonction erreur S est $\nabla^2 S = 2X^T X$ et nous avons $\hat{\Sigma} = \frac{1}{2T}\nabla^2 S$; soit X_t la ligne (t-p) de X et définissons f_t par $f_t(\beta) = \Sigma_i \beta_i X_{t-i} = X_t \beta$; alors $E[(\nabla f_t)(\nabla f_t)^T] = E(X_t^T X_t) = \Sigma$ et $\hat{\Sigma}$ peut être aussi exprimée par $\hat{\Sigma} = \frac{1}{T}\sum_t \left[(\nabla f_t)(\nabla f_t)^T\right] X^T X$.
- La stationnarité de la série est essentielle, sinon il sera nécessaire de pré traiter les données pour écarter les tendances et la périodicité ; par exemple, si B est l'opérateur retard, la transformation (Id-B) (X_t) supprime une tendance linéaire et la transformation (Id-B_7) (X_t) supprime n'importe quelle tendance linéaire et périodicité hebdomadaire. Ces résultats restent valables pour un modèle ARX linéaire.

2.4.3. Le cas général des modèles neuronaux

2.4.3.1. Apprentissage d'un réseau de neurones Dans le jargon connexionniste, l'estimation des paramètres s'appelle l'apprentissage. Soit un perceptron multicouches défini par une architecture à p entrées et à une seule sortie. On présente successivement des éléments d'une base d'exemples de taille T. Cette base est composée d'un ensemble de vecteurs $(X^t)_{t=1,2.....T}$, $X^t=(X^t{}_1, \ldots X^t{}_p)$, et d'un ensemble de valeurs $(Y^t)_{t=1,2\ldots T} \in IR$. On appelle les valeurs $(Y^t)_{t=1,2\ldots T} \in IR$ les valeurs désirées. Lorsqu'on présente X^t, on ajuste alors le vecteur des paramètres à estimer θ^t, à partir du vecteur des paramètres précédent θ^{t-1}, de sorte que la sortie X_t du réseau s'approche de la sortie désirée Y^t. Puisque les sorties désirées sont connues a priori, on parle d'apprentissage supervisé. Nous pouvons en fait rencontrer deux types distincts d'apprentissage:

- l'apprentissage supervisé dans lequel le réseau apprend en lui fournissant des échantillons d'entrées et sorties correspondantes à ces dernières.
- l'apprentissage non supervisé ou Self Organisation dans lequel une unité de sortie apprend pour répondre à des groupes de modèles à l'intérieur de l'entrée.

Pour ces deux types d'apprentissage, il y a également un choix traditionnel entre :

- l'apprentissage << off line >> : toutes les données sont dans une base d'exemples d'apprentissage qui sont traités simultanément;
- l'apprentissage << on-line >> : les exemples sont présentés les uns après les autres au fur et à mesure de leur disponibilité.

Nous nous limitons, dans ce travail, à l'apprentissage supervisé à partir d'une base d'exemples. Rappelons que d'après (2.54), $\forall t \in IZ, \quad X_t = f_\theta(X_{t-1}^{(p)}) + \varepsilon_t$, f_θ étant de la classe des perceptrons multicouches (Fig. 2.24) où $\theta \in IR^l$ est le vecteur des paramètres, c'est-à-dire :

$$f_\theta(X_{t-1}, X_{t-2}, ..., X_{t-p}) = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ij} X_{t-i} + \beta_{0j}) + \alpha_0 \qquad (2.57)$$

D'après les notations de Fig 2.24, il y a (p+1) unités d'entrées $(1, X_{t-1}, X_{t-2}, ...X_{t-p})$, n unités cachées avec une fonction d'activation sigmoïde ψ et une unité de sortie linéaire X_t.
Soit $(X)_{t\ 1-p \le t \le T}$, T+p valeurs successives de la série, cette suite est la base d'apprentissage.
Notons $S_T(\theta)$ la somme des erreurs quadratiques appelée aussi fonction erreur :

$$S_T(\theta) = \sum_{t=1}^{T} (X_t - \hat{X}_t)^2 \qquad (2.58)$$

Où $\hat{X}_t = f_\theta(X_{t-1}, ...X_{t-p})$ .Si nous voulons tester le modèle sur des valeurs de la série qui sont en dehors de la base d'apprentissage, par exemple $(X_{T+j})_{j=1,2...T'}$, cette nouvelle suite de réels est appelée base de test. On reconnaît un bon apprentissage par le fait que la valeur de la fonction de coût obtenue sur la base d'apprentissage et celle obtenue sur la base de test sont significativement égales. Dans le cas contraire, on parle de mauvaise adéquation entre le modèle et le phénomène à modéliser ou encore mieux de mauvaise généralisation. C'est un problème d'optimisation, pour lequel la fonction de coût à minimiser est une fonction quadratique facile à étudier si la fonction f_θ est linéaire.

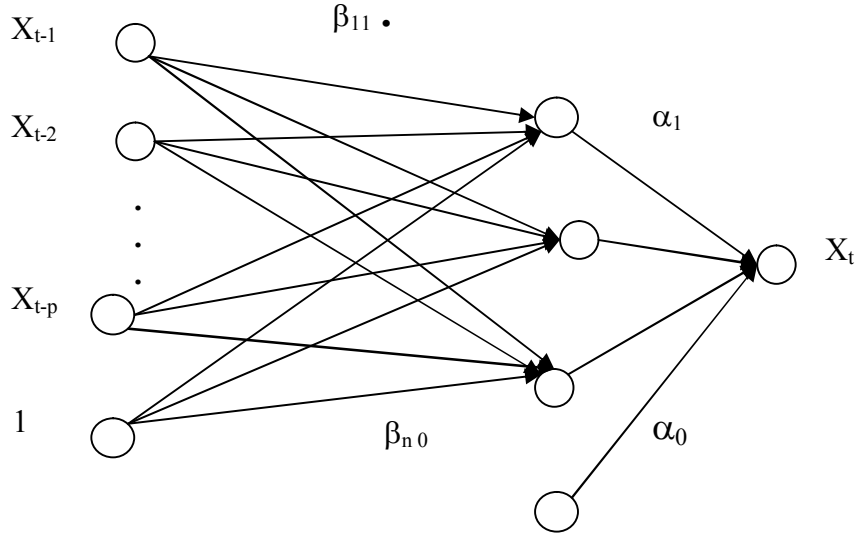


Fig 2.24

Si la fonction de coût est non linéaire, on doit recourir à des algorithmes itératifs basés en général sur le gradient de cette fonction. Or c'est notre cas puisqu'on on est en présence d'un modèle $NAR_n(p)$ (équation 2.45), la fonction de coût à minimiser est non linéaire. Le vecteur θ est alors soumis à des conditions de cohérence à cause de l'architecture particulière des perceptrons multicouches :

Définition 2.4. Soit un modèle de type $NAR_n(p)$ (équation 2.45) à p entrées et n neurones cachés. Soit $\theta = \left\{(\alpha_i)_{0\le i\le n}(\beta_{ij})_{0\le i\le p,1\le j\le n}\right\}$ le vecteur des paramètres de ce modèle. On appelle conditions de cohérence du modèle $NAR_n(p)$ l'ensemble de conditions suivantes :

- Le poids α_0 est différent de 0 ;
- Pour n'importe quel j=1,….n, α_j est nul ssi $\beta_{ij}=0$ pour tout i=0,1,…,p ;
- Pour n'importe quel j=1,….n, si $\beta_{1j}=\ldots=\beta_{pj}=0$ alors $\beta_{0j}=0$;

Ces contraintes sont indispensables pour la cohérence du réseau. En effet, s'il existe i et j tels que $\alpha_j=0$ et $\beta_{ij}\neq 0$ le poids β_{ij} n'est pas nul mais devient inutile dans le calcul de la sortie.

Dans la suite de notre travail, nous utiliserons $\hat{\theta}_T := \underset{\theta\in\Theta}{\text{Arg min}}\, S_T(\theta)$. Dans le cadre de la prévision de séries temporelles, un autre critère de comparaison des performances de l'optimisation est général utilisé ; ce critère provient de l'erreur quadratique brute mais il est normalisé pour être indépendant de l'unité de mesure des valeurs de la série. Ce critère est appelé NMSE (θ) pour normalized mean square error.

Définition 2.5. Dans le cadre défini paragraphe 3, on appelle NMSE (θ) le critère de la forme :

$$\text{NMSE}(\theta) = \frac{S_T(\theta)}{\text{Var}^T(X)} \qquad (2.59)$$

avec

$$Var^T(X) = \frac{1}{T}\sum_{t=1}^{T}(X_t - \overline{X}^T)^2 \quad \text{et} \quad \overline{X}^T = \frac{1}{T}\sum_{t=1}^{T}X_t$$

$\text{Var}^T(X)$ est la variance empirique des valeurs de la série et peut être vue comme la performance d'un modèle trivial M_0 tel que pour chaque temps t une prévision de X_t est égale à la moyenne de la série. Ainsi si le critère NMSE (θ) est égal (respectivement supérieur) à 1, le modèle est aussi performant que le modèle M_0. Enfin, plus ce critère est proche de 0 et plus le modèle est performant ; on remarque aussi que $S_T(\theta)$ et NMSE (θ) sont proportionnelles à une constante positive prés, leurs minimisations sont donc deux problèmes équivalents.

2.4.3.2. Consistance de l'estimateur des moindres carrés Des résultats du théorème 2.11 et avec les mêmes notations, on a la consistance de l'estimateur des moindres carrés du vecteur des paramètres. Nous allons les reprendre et les appliquer à l'estimateur des moindres carrés d'un modèle neuronal basé sur un perceptron multicouches noté $NAR_n(p)$. Rappelons que $\theta = (\theta_1, \theta_2, \ldots \theta_l)$, $l \geq 1$ est le vecteur des paramètres, ou le vecteur des l poids synaptiques. Comme dans ce qui précède, θ_0 désigne le vecteur des vrais paramètres qui correspond au modèle optimal et $\hat{\theta}_T$ est l'estimateur des moindres carrés. Le modèle est supposé identique à celui de 2.45. En particulier en posant $X_t^{(p)} := (X_t, \ldots . X_{t-p+1})$, la série $(X_t^{(p)})$ est une chaîne de Markov homogène dans IR^p.

Théorème 2.13. Soit un modèle $NAR_n(p)$ (équation 2.45) avec les fonctions d'activation sigmoïdes ψ. Supposons que les conditions [R] suivantes soient satisfaites :

a- $(\varepsilon_t)_{t>0}$ est une suite de variables aléatoires centrées, indépendantes, identiquement distribuées et indépendantes de l'état initial de la chaîne $(X_t^{(p)})$.

b- ε_t a une densité positive par rapport à la mesure de Lebesgue avec $E(\varepsilon_1)=0$ et $E(\varepsilon^2_1) < \infty$.

c- θ appartient à un sous ensemble compact Θ de IR^l, tel que $\theta_0 \in \overset{0}{\Theta}$.

d- (Condition d'identifiabilité) pour tout θ, $\theta \neq \theta_0$, $f_\theta \neq f_{\theta 0}$ c'est-à-dire qu'il existe un x tel que $f_\theta(x) \neq f_{\theta 0}(x)$.

Alors l'estimateur des moindres carrés est fortement consistant.

Démonstration Le modèle $NAR_n(p)$ est un modèle $ARF_1(p)$ particulier. On vérifie alors que les conditions [M] et [D] du théorème 2.9 relatif au modèle $ARF_d(p)$ (d=1) sont satisfaites.

Conditions [M] : nous pouvons vérifier que les conditions [R] du théorème 2.13 satisfont les hypothèses [C.2] du théorème 2.11.En particulier sous les conditions [R](a) et [R](b), la chaîne issue du modèle $NAR_n(p)$ est stable et a une unique loi invariante et ceci implique la condition de stabilité de [M]. Les conditions [M]-(a) et [M](b) sont vérifiées par les conditions [R](a) et [R](b).Puisque la fonction $f_\theta : IR^p \rightarrow IR$ est lipschitzienne en x et bornée, comme nous l'avons vu à la proposition 2.1, et que Θ est compact, alors la condition[R](c), $(x,\theta) \rightarrow f_\theta(x)$ est uniformément continue sur $IR^p \times \Theta$, ce qui satisfait [M]-(c) et [M]-(d).

Conditions [D] : pour l'identifiabilité du modèle, puisque nous avons [R](a) et [R](b), le bruit (ε_t) a donc une densité strictement positive par rapport à la mesure de Lebesgue. Nous savons alors que pour le modèle NAR_n (p), la loi stationnaire de la chaîne a aussi une densité strictement positive par rapport à la mesure de Lebesgue. La Condition [R] (d) suffit alors pour satisfaire la condition d'identifiabilité.

Ce résultat est important puisqu'il assure la convergence presque sure de l'estimateur vers le vrai paramètre lors de l'optimisation de la fonction de coût (équation 2.58).

Remarques

1. Pour un échantillon fini, pour la même architecture, la classe des modèles linéaires peut être vue comme un sous modèle approché du modèle non linéaire. Donc il est absurde qu'un perceptron multicouches non linéaire ne puisse pas être plus mauvais que le modèle linéaire. Si le modèle le plus adapté est linéaire, les poids peuvent être ajustés de telle sorte que la fonction ψ agit au voisinage de zéro, où elle est presque linéaire. Comme le modèle est surparamètré, il peut y avoir aussi d'autres solutions.

2. Comme nous pouvons le remarquer en utilisant (2.57) et le fait que $|\psi(t)| \leq 1$, nous avons $\left| \hat{X}_t \right| \leq \sum_{j=1}^{n} |\alpha_j| + |\alpha_0|$. Ainsi, un tel réseau de neurones peut correctement représenter, mais seulement de manière limitée, des variables aléatoires ou plus généralement la variable aléatoire qui peut être tronquée [$\exists$ A$\in$IR tel que P ($|A| > 0$) $\approx$ o]. Donc, un perceptron multicouches ne peut pas modéliser une série chronologique contenant une tendance.

Il a été démontré [voir P.Doukhan et M.Ghindes, A.Tsybakov, A.Mokaddem et H.Tong] que sous certaines conditions faibles (des conditions portant sur la fonction d'activation ψ), les propriétés asymptotiques de l'estimateur des moindres carrés dans le cas linéaire peuvent être généralisées aux processus autorégressifs non linéaires. Plus précisément, il a été établi que l'estimateur des moindres carrés est asymptotiquement gaussien (quand $T \to \infty$) comme dans le cas linéaire. C'est une conséquence des résultats types du théorème central limite, en fait on a : $\sqrt{T}\left(\hat{\theta} - \theta\right) \xrightarrow{D} N_m\left(0, \sigma^2 \Sigma^{-1}\right)$ quand T tend vers l'infini.où ρ^2 est la variance

résiduelle estimée par [S ($\hat{\theta}$) / T], et Σ est la matrice Hessienne de la fonction erreur S (θ) multipliée par (1/2T) et estimée par (1/2T) ∇^2 S ($\hat{\theta}$).
L'aspect statistique le plus intéressant de ce résultat est la possibilité de connaître la distribution asymptotique de l'estimateur des moindres carrés de θ. Ainsi, pour chaque paramètre poids θ_{ij} , il est possible de construire un (1- α) intervalle de confiance approché.

Rappelons que: étant données les observations, nous essayons de trouver un bon modèle aussi parcimonieux que possible, puisque les modèles simples sont robustes, aisés pour l'interprétation des paramètres et enfin, ils facilitent les calculs. Pour ce faire, plusieurs problèmes doivent être résolus dont les plus importants sont :

- Comment initialiser l'architecture et les poids ?
- Quelle est la méthode la plus adéquate pour l'apprentissage, c'est-à-dire pour minimiser la fonction erreur ?

La première question reste un sujet de recherche actif malgré qu'il y a eu des réponses avec des travaux [voir M.Cottrel, B et Y. Girard et M.Mangeas] qui proposèrent la méthode de la « stepwise descent » qui préconise de commencer avec toutes les variables d'entrées, pas de couche cachée c'est-à-dire avec un modèle de base linéaire. Alors, on ajoute progressivement des unités à la couche cachée, en calculant l'estimateur modifié de la variance (S ($\hat{\theta}$) / T-m), à chaque étape et aussi longtemps que cet estimateur décroît significativement. Les poids initiaux sont choisis aléatoirement ou nuls excepté le paramètre α_0 qu'on pose égal à $\overline{X}$.

La seconde question nécessite de faire le tour sur les méthodes d'optimisation puisque les méthodes de résolution précédentes ne sont plus utilisables. On a alors recours à des méthodes itératives de type "gradient" pour effectuer l'estimation des paramètres des modèles non linéaires. Ces méthodes d'optimisation sont coûteuses en temps de calcul, mais restent simples à mettre en oeuvre, et s'appliquent à toutes les fonctions f(.; θ) dérivables par rapport à θ. Ceci est en particulier le cas lorsque f (.;θ) est réalisé par un réseau de neurones. Les méthodes d'optimisation feront l'objet du sous paragraphe suivant.

2.4.4. Méthodes d'optimisation

Dans la pratique, il existe quatre types d'algorithmes d'optimisation afin de choisir les paramètres d'un réseau de neurones pour minimiser S_T (θ).Les trois premières méthodes, celle

de la plus forte descente, celle du gradient conjugué et la méthodes quasi Newtoniennes dont la plus connue est celle de BFGS, sont des méthodes d'optimisation générales.
La méthode de Levemberg Marquardt est spécialement adaptée à une fonction d'erreur qui résulte d'un critère d'erreur quadratique de la forme qu'on a nous même supposée, cette méthode et celle de BFSG sont les plus utilisées car elle présentent plusieurs avantages.

2.4.4.1. Algorithmes de descente itératifs Nous notons généralement tous les paramètres du réseau (poids de connection et biais) par un vecteur $\theta \in \Theta \subset IR^l$ .L'itération récursive de base et commune à toutes les méthodes d'apprentissage largement utilisées actuellement est un processus à deux étapes qui détermine un nouveau vecteur de paramètres θ_{k+1} en fonction du vecteur courant θ_k, à travers une première sélection d'une direction de recherche d_k puis d'un taux d'apprentissage ou taille du pas α_k

$$\theta_{k+1} = \theta_k + \alpha_k d_k \tag{2.60}$$

Une recherche à travers un espace l-dimensionnel d'un minimum est conduite en recherchant séquentiellement le long des directions individuelles $\{d_k\}$ de sorte que la distance recherchée le long de d_k soit déterminée par le taux d'apprentissage ou taille du pas α_k. Typiquement, les algorithmes de descente sont markoviens dans le sens où pour un état, l'état futur dépend seulement de son état actuel et non de la succession des états passés qui l'y ont conduit. L'état après l'itération k consiste en le triplet (θ_k, d_k, g_k) où g_k est le gradient de S_θ.
On pourra explorer le choix de la direction de recherche en considérant l'approximation de premier ordre suivante (i.e $f(x) - f(x_0) \approx f'(x_0)\ (x-x_0)$) aux valeurs successives de la fonction objectif (fonction erreur) :

$$(S_{k+1} - S_k) \approx g(\theta_k)^T(\theta_{k+1} - \theta_k) \tag{2.61}$$

Si nous souhaitons que notre algorithme itératif nous donne une descente régulière, alors nous devons réduire l'erreur à chaque étape. Pour les incréments $\theta_{k+1} - \theta_k$ qui ne sont pas aussi grands pour que l'approximation de Taylor de premier ordre de l'équation (2.61) soit valide, on voit que nous devons avoir

$$\begin{aligned} & g(\theta_k)^T(\theta_{k+1} - \theta_k) = g(\theta_k)^T(\alpha_k d_k) \\ & \alpha_k g_k{}^T d_k \prec 0 \quad (\text{condition de descente}) \end{aligned} \tag{2.62}$$

Le moyen le plus simple de satisfaire la condition de descente de l'équation (2.61) est d'avoir

$$\alpha_k \succ 0 \quad , d_k = -g_k \tag{2.63}$$

Ce choix particulier de la direction de descente est la base des algorithmes de la plus forte descente, la direction de recherche est prise dans la direction de la plus forte descente de la surface de l'erreur. Plus généralement si $\{M_k\}$ est une suite de matrices définies positives, alors $d_k = -M_k g_k$ aussi satisfait la condition de descente. La méthode de Newton choisit $M_k = H^{-1}(\theta_k)$, H étant la Hessienne qui nous est généralement inconnue. Les méthodes quasi Newtoniennes et de Levenberg-Marquardt font d'autres choix pour M_k en prenant des approximations définies positives de l'inverse da la matrice H. Un choix « optimal » pour le taux d'apprentissage α_k pour un choix donné de la direction du gradient d_k est celui qui minimise S_{k+1}, $\alpha^*_k = \arg\min_\alpha S(\theta_k + \alpha d_k)$ .Pour minimiser, nous utilisons :

$$\frac{\partial S(\theta_{k+1})}{\partial \alpha}\Big|_{\alpha=\alpha_k^*} = \frac{\partial S(\theta_k + \alpha d_k)}{\partial \alpha}\Big|_{\alpha=\alpha_k^*} = 0$$

Pour évaluer cette équation, remarquer que $\frac{\partial S(\theta_k + \alpha d_k)}{\partial \alpha} = g_{k+1}^T d_k$

Et conclure que pour le taux d'apprentissage optimal nous devons satisfaire la condition d'orthogonalité

$$g_{k+1}^T d_k = 0 \tag{2.64}$$

Le gradient de l'erreur à la fin de l'étape d'itération est orthogonal à la direction de descente le long de laquelle nous avons changé le vecteur des paramètres. Ainsi, dans le cas de la plus forte descente (2.63), les gradients successifs sont orthogonaux entre eux. Nous avons utilisé « optimal » entre parenthèses pour souligner le fait que ce choix du taux d'apprentissage est vraiment optimal seulement si nous sommes à l'étape finale de l'itération. Si, comme c'est souvent le cas, nous sommes à une étape intermédiaire de l'itération, alors le choix ci-dessus pour le taux d'apprentissage n'est pas optimal en ce qui concerne la minimisation éventuelle de l'erreur en achèvement du processus de la descente itérative. Dans notre notation,

$$g_{k+1} - g_k = \alpha_k H d_k \tag{2.65}$$

Ainsi, la condition d'optimalité pour le taux d'apprentissage α_k obtenu à partir de la condition d'orthogonalité (2.64) devient :

$$\alpha_k^* = \frac{-d_k^T g_k}{d_k^T H d_k} \tag{2.66}$$

Quand les directions de recherche sont choisies via $d_k = -M_k g_k$, avec M_k symétrique, alors le taux d'apprentissage optimal est :

$$\alpha_k^* = \frac{g_k^T M_k g_k}{g_k^T M_k H M_k g_k} \tag{2.67}$$

Dans le cas particulier de la plus forte descente pour une fonction d'erreur quadratique, M_k est l'identité et

$$\alpha_k^* = \frac{g_k^T g_k}{g_k^T H g_k} \qquad (2.68)$$

Dans la méthode de Newton, $M_k = H^{-1}(\theta_k)$ et $\alpha_k^*=1$ .Mais l'équation (2.67) concerne la Hessienne et le calcul de cette dernière prend souvent beaucoup de temps et exige de grandes capacités de stockage. Pour un réseau à p paramètres, on a ½ p (p+1) éléments !

Les méthodes de gradient Ces méthodes appelées méthodes du premier ordre, ont pour principe de modifier les paramètres de la fonction à minimiser dans la direction de la plus forte pente, donc dans la direction opposée au vecteur gradient (voir fig 2.25).

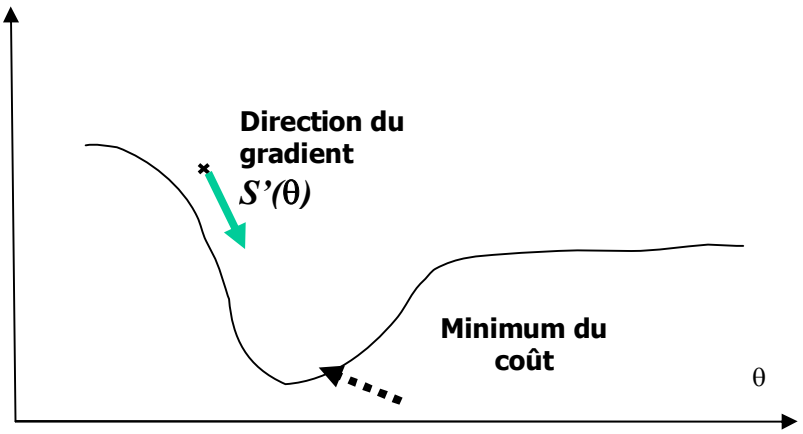


Fig 2.25 Principe de la méthode du gradient

La méthode du gradient la plus simple à mettre en oeuvre est celle telle que à l'itération k, la modification de θ est : $\theta^k = \theta^{k-1} - \eta \nabla S_T(\theta^{k-1})$

C'est-à-dire en prenant dans (2.60) d_k = - grad $S_T(\theta^{k-1})$ et $\alpha_k = \eta$. C'est la méthode du gradient totale (ou batch), dite totale car on considère le gradient de la fonction de coût globale, calculée sur toutes les données. Le pas η est une constante et la direction de descente est simplement l'opposé de celle du gradient. Cette méthode est très utilisée. Elle a pour avantages une grande facilité de mise en oeuvre et une grande robustesse. Le choix de η n'est pas critique pour la convergence. Mais même si elle est efficace, elle est loin d'un minimum, et la vitesse de convergence diminue lorsque l'on s'approche du minimum (la modification de ηest proportionnelle à grad $S_T(\theta_{k-1})$ qui tend vers 0).

En général, on utilise une autre version de cet algorithme où on modifie le vecteur des paramètres dans la direction opposée au gradient associé à un seul terme d'erreur quadratique

de la fonction coût , tiré aléatoirement suivant une loi uniforme. Pour un t ∈ {1,2,…T} choisi aléatoirement, à l'étape k,

$$\theta^{k} = \theta^{k-1} - \eta \frac{\partial(X_t - f_{\theta^{k-1}}(X_{t-1}^{(p)}))^2}{\partial\theta}$$

Cette variante de la méthode du gradient total est appelée méthode du gradient stochastique (ou on-line) (Duflo, 1990). L'inconvénient essentiel de la méthode du gradient réside dans sa lenteur de convergence et le fait que la vitesse de cette dernière dépend étroitement de la valeur du pas d'apprentissage et on ne connaît pas a priori, la valeur la plus adéquate. En effet, si celui ci est trop petit, la convergence est «lente» vers la solution ; s'il est trop grand, il y a risque d'oscillations. Les heuristiques courantes consistent à diminuer le pas d'apprentissage au fur et à mesure à la main ou en fonction de la forme de la surface d'erreur. La méthode du gradient stochastique est plus rapide que la première mais son efficacité à éviter les minimums locaux dépend aussi de la valeur du pas. Afin de remédier à cette difficulté, les méthodes dites de gradient conjugué ont été introduites. Elles permettent, dans le cas quadratique de garantir une vitesse de convergence. La procédure du gradient utilise comme information la pente mais celle de Newton utilise la pente et la courbure (la dérivée seconde), elle se révèle par conséquent optimale pour minimiser des coûts quadratiques.

2.4.4.2. Méthode de Newton

a) Développement de Taylor de la fonction erreur Les fonctions de transfert sont différentiables à n'importe quel ordre et à travers la règle de la chaîne de différentiation, ceci implique que la fonction erreur $S_T(\theta)$ définie en (2.58) est aussi différentiable à n'importe quel ordre. Ainsi, nous pouvons écrire un développement de Taylor de $S_T(\theta)$ en θ. On va considérer les algorithmes pour la minimisation de $S_T(\theta)$ en supposant que l'on peut tronquer un développement en série de Taylor en un point θ qui peut être un minimum local. Si nous notons le gradient de $S_T(\theta)$ par $\nabla S_T(\theta) = \frac{\partial S_T}{\partial \theta_i} \Big]_{\theta}$ et la matrice Hessienne de $S_T(\theta)$ par $H(\theta)=[H_{ij}(\theta)]= \nabla^2 S_T(\theta)$ où $H_{ij} = \frac{\partial^2 S_T(\theta)}{\partial\theta_i \partial\theta_j} = H_{ji}$, la série de Taylor de $S_T(\theta)$ supposée deux fois continûment différentiable en θ peut alors être donnée par :

$$S_T(\theta) = S_T(\theta_0) + \nabla S_T(\theta_0)^T(\theta-\theta_0) + \frac{1}{2}(\theta-\theta_0)^T H(\theta_0)(\theta-\theta_0) + o(\|(\theta-\theta_0)\|^2) \qquad (2.69)$$

où $o(\delta)$ désigne un terme tel que $\lim_{\delta\to 0} \frac{o(\delta)}{\delta} = 0$ Dans la plupart des analyses des algorithmes d'apprentissage, on va supposer avoir un modèle d'approximation quadratique :

$$m(\theta) = S_T(\theta_0) + \nabla S_T(\theta_0)^T(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^T H(\theta_0)(\theta - \theta_0) \quad (2.70)$$

Dans la mesure où les fonctions de transfert sont différentiables, cette approximation sera vraie seulement dans un voisinage de θ_0 assez petit.

Lemme 4.1. (minimum d'une quadratique) Le modèle m de l'équation (2.70) possède un minimum unique si et seulement si H est une matrice définie positive.

b) Méthode de Newton En prenant le gradient dans le modèle quadratique (2.70), on a

$$\nabla m = \nabla S_T(\theta_0) + H(\theta - \theta_0) \quad (2.71)$$

Posant le gradient égal à zéro et supposant que θ^* réalise le minimum, on a :

$$\theta^* = \theta_0 - H^{-1}\nabla S_T(\theta_0) \quad (2.72)$$

Le modèle m peut être ré exprimé en fonction de la valeur minimum θ^* et

$m(\theta^*) = m(\theta_0) + \frac{1}{2}\nabla S_T(\theta_0)^T H^{-1}\nabla S_T(\theta_0)$, d'après : $m(\theta) = m(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H(\theta^*)(\theta - \theta^*)$, c'est un résultat qui résulte de (2.70) en complétant le carré ou en reconnaissant que $\nabla S_T(\theta^*) = 0$.

Ainsi, à partir d'une valeur initiale quelconque du vecteur poids, on peut dans le cas quadratique passer en une seule étape à la valeur minimum quand elle existe. Ceci est connu par la méthode de Newton qui peut être appliquée dans le cas non quadratique où H est la Hessienne et que celle ci est définie positive. En d'autres termes, cette méthode préconise qu' à la kième itération, on ait : $\theta^k = \theta^{k-1} - [H(\theta^{k-1})]^{-1} \nabla S_T(\theta^{k-1})$

Le pas est constant (α_k=1), et la direction de déplacement est : $d_k = -[H(\theta^{k-1})]^{-1} \nabla S_T(\theta^{k-1})$

Si $S_T(\theta)$ est quadratique, l'algorithme converge en une itération. La méthode est donc efficace si θ est proche d'un minimum autour duquel $S_T(\theta)$ est presque quadratique et on a vu que pour que la méthode converge vers le minimum, la matrice $H(\theta)$ doit être définie positive. Cette méthode est peu employée, car elle nécessite le calcul et l'inversion de H à chaque itération, et cette dernière doit être définie positive (à chaque itération aussi). On lui préfère les méthodes économiques dites quasi Newtoniennes et celle de Levemberg Marquardt ou encore les méthodes du gradient conjugué où on calcule indirectement la Hessienne.

2.4.4.3. Les méthodes quasi-newtoniennes Les méthodes quasi-newtoniennes, ou méthodes du second ordre, ont pour principe d'utiliser l'information issue des dérivées secondes. Mais ces dernières sont estimées en général à partir des dérivées premières, pour éviter des temps de calcul trop importants.

Pour ces méthodes l'inverse de la Hessienne $H(\Theta^{k-1})]^{-1}$ est approximée par une matrice M_k définie positive, modifiée à chaque itération. La suite des matrices $\{M_k\}$ est construite de manière à converger vers l'inverse de la Hessienne lorsque la fonction $S_T(\theta)$ est quadratique, approximation qui peut être légitimement faite lorsque l'on s'approche du minimum. La modification des paramètres à chaque itération est :

$$\theta^k = \theta^{k-1} - \mu_{k-1} M^{k-1} \nabla S_T(\theta^{k-1})$$

où μ_{k-1} est le pas de déplacement qui minimise la fonction $g(\mu) = S_T(\theta^{k-1} + \mu d_{k-1})$. La direction de déplacement est $d_k = -M_{k-1}.\nabla S_T(\theta^{k-1})$.Les différentes méthodes quasi-newtoniennes différent alors par le choix des matrices M. La méthode BFGS (Broyden-Fletcher-Goldano-Shanno) développée indépendamment par Broyden (1970), Fletcher (1970), Goldfrab (1970) et Shanno (1969), dont la vitesse de convergence est beaucoup plus grande que celle de la méthode du gradient, est la plus adéquate car elle n'impose pas une forme particulière à la fonction erreur et parce qu'elle est peu sensible à l'exactitude de la minimisation unidimensionnelle (c'est-à-dire calcul de $(\mu_k)_{k>1}$) . Pour la méthode BFGS, la matrice M_k est calculée à la kième itération par ce qui suit :

$$M^k = M^{k-1} + \frac{(\theta^k - \theta^{k-1})(\theta^k - \theta^{k-1})}{(\theta^k - \theta^{k-1})(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k))} - \frac{[M^{k-1}(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k)] \times [M^{k-1}(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k))]}{(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k)) M^{k-1}(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k))}$$

Ici, $\times$ est le produit de deux vecteurs : la composante a_{ij} de la matrice $u \times v$ de deux vecteurs quelconques de même dimension u et v s'écrit $a_{ij} = u_i \times v_j$.

En pratique, on a intérêt à commencer par des itérations de l'algorithme du gradient simple, qui est efficace loin du minimum. M_0 est ensuite initialisée à la matrice identité, puis l'on commute sur une méthode quasi-newtonienne. Pour un réseau de neurones, les paramètres à déterminer sont les poids et la fonction à minimiser est la fonction de coût en sortie de réseau (en général l'erreur quadratique).Comme la méthode du gradient total, cette méthode est basée sur le gradient de la fonction de coût de l'ensemble des données.

2.4.4.4. Méthode de Levenberg –Marquardt La méthode d'optimisation de Levenberg – Marquardt (LM), Marquardt (1963) est une méthode intermédiaire entre celle très simple du gradient de la plus forte pente et une méthode quasi Newtonnienne utilisant l'inverse de la matrice Hessienne .C'est la méthode la plus utilisée dés lors que la fonction de coût est une somme d'erreurs quadratiques. Reprenons les notations de la section 2.4.2 et posons :

$\varepsilon_t = X_t - f_\theta(X_{t-1}^{(p)}), \qquad \forall t \in \{1,2,....T\}$. On a alors :

$$S_T(\theta) = \sum_{t=1}^{T} \varepsilon_t^2 \quad \text{et (: } \frac{\partial S_T}{\partial \theta_i} \quad = \quad -2 \sum_{t=1}^{T} \varepsilon_t \frac{\partial f_\theta(X_{t-1}^{(p)})}{\partial \theta_i} \tag{2.73}$$

$$\frac{\partial^2 S_T(\theta)}{\partial \theta_i \partial \theta_j} \quad = \quad 2\Big(\sum_{t=1}^{T} \frac{\partial f_\theta(X_{t-1}^{(p)})}{\partial \theta_i} \frac{\partial f_\theta(X_{t-1}^{(p)})}{\partial \theta_j} - \varepsilon_t \frac{\partial^2 f_\theta(X_{t-1}^{(p)})}{\partial \theta_i \partial \theta_j} \Big)$$

On rappelle que pour $i \in \{1,2,...I\}$, les $(\theta)_{i=1,2,.....I}$ sont les composantes du vecteur des paramètres θ. Quand on s'approche du minimum, $(\varepsilon_t)_{1\le t\le T}$ tend vers une suite de variables aléatoires indépendantes, centrées et de variance constante (bruit blanc). Le deuxième terme de (2.73) tends alors vers 0 en probabilité. Le principe de la méthode LM consiste à exploiter cette approximation. Posons :

$$\alpha_{ij} = \alpha_{ji} = \sum_{t=1}^{T} \Big(\frac{\partial f_\theta(X_{t-1}^{(p)})}{\partial \theta_i} \frac{\partial f_\theta(X_{t-1}^{(p)})}{\partial \theta_j}\Big) \approx \frac{1}{2} \frac{\partial^2 S_T(\theta)}{\partial \theta_i \partial \theta_j}$$

et

$$\beta_i = \frac{1}{2} \frac{\partial S_T(\theta)}{\partial \theta_i}$$

D'après le principe des méthodes du second ordre, on peut alors modifier le vecteur des paramètres en utilisant l'itération de Newton

$$\theta^k = \theta^{k-1} - [H(\theta^{k-1})]^{-1} \nabla S_T(\theta^{k-1}).$$

Au lieu d'estimer $H^{-1}(\theta)$ suivant la méthode décrite à la section précédente, on peut tenter de résoudre le système d'équation linéaire: $\sum_{j=1}^{t} \alpha_{ij} \delta_j = \beta_i$ où δ_i est la ième composante de $\theta^k - \theta^{k-1}$ Cette méthode a l'inconvenant d'être instable L'astuce de LM consiste alors à remplacer les $(\alpha_{ij})_{1\le i,j\le I}$ par $(\alpha'_{ij})_{1\le i,j\le I}$ tels que : $\begin{cases} \alpha'_{jj} = \alpha_{jj}(1+\lambda) & \forall j \in \{1,2,..I\} \\ \alpha'_{ij} = \alpha_{ij} & si\ i,j \in \{1,2,..I\}, i \neq j \end{cases}$ où λ est un réel positif. Quand λ est grand, la matrice $(\alpha'_{ij})_{1\le i,j\le I}$ devient fortement diagonale, et on se rapproche de la méthode du gradient de la plus forte pente : $\delta_i \approx \frac{1}{\alpha_{ii}} \beta_i$ et $\frac{1}{\alpha_{ii}}$ est vu comme le pas du gradient. Quand λ est « petit », on se rapproche d'une méthode du second ordre utilisant la matrice Hessienne H : $\theta^k - \theta^{k-1} \approx - [H(\theta^{k-1})]^{-1} \nabla S_T(\theta^{k-1})$.

Notons que cette méthode n'est plus valide si la fonction de coût n'est plus la somme des erreurs quadratiques du modèle. Dans la plupart des cas, les chercheurs et praticiens utilisent soit la méthode BFGS soit la méthode LM pour l'estimation des paramètres.

On voit dans tout ce qui précède que toutes ces méthodes requièrent un calcul efficace et répété de gradients. Les chercheurs se sont alors intéressés à des méthodes qui permettent le

calcul du gradient dans un temps réduit. Dans ce qui suit, on va présenter cet algorithme très important pour l'optimisation:

2.4.4.5. L'algorithme de rétro propagation pour l'évaluation du gradient A l'époque de Rosenblatt entre 1950 et 1960, le problème de recherche des poids qui ajustent le mieux les données d'apprentissage fut impossible à résoudre et le développement des réseaux de neurones devenait limité. Minsky et Papert ont manifesté une vigoureuse limitation des réseaux de neurones amenant ces derniers à un manque d'intérêt.
L'intérêt actuel très fort à l'envers des perceptrons multicouches a commencé dans les années 80, après que plusieurs groupes de recherche ont indépendamment proposé un algorithme très simple, principalement [Rumelhart et al, (1986)] et [Lecun et al, 1985)] qui est celui de la rétro propagation du gradient devenu très célèbre, et qui est connu sous l'appellation anglaise « back propagation ». Grâce à cet algorithme, les réseaux de neurones sont revenus sur la scène. Il est basé sur un calcul astucieux des dérivées de fonctions composées et fournit une méthode efficace pour l'évaluation du vecteur gradient. Cet algorithme ne calcule pas directement le gradient mais utilise la règle de la chaîne de différentiation en organisant efficacement le calcul de celui ci pour des réseaux de neurones ayant plus d'une couche cachée. Cet algorithme est un outil efficace pour calculer le gradient de la fonction coût (ou erreur). Son principe est de minimiser une fonction d'erreur, par un calcul exemple par exemple puis sommation des résultats. Il s'agit ensuite de calculer la contribution à cette erreur de chacun des poids. C'est cette étape qui est difficile. En effet, chacun des poids influe sur le neurone correspondant, mais, la modification pour ce neurone va influer sur tous les neurones des couches suivantes.

a) Cas d'un perceptron multicouches à une seule couche cachée et à une seule sortie
Soit un perceptron multicouches défini par une architecture à p entrées x_i, une seule couche cachée à n cellules et une sortie y ; les poids sont notés $[\beta ji]_{i=1\ldots p,\ j=1\ldots n}$ et $(\alpha j)_{j=1\ldots n}$. On pose $W_1= [\beta ji]$ et $W_2=[\alpha j]$. L'erreur du perceptron multicouches sur un échantillon d'apprentissage d'exemples (de taille T) $(\vec{x}_t, y_t^{des})_{t=1\ldots T}$ où l'exemple $\vec{x}_l$ est de dimension p est définie par :

$S(\theta) = \sum_{t=1}^{T} S_t(\theta) = \sum_{t=1}^{T} (y_t^{des} - y_t)^2$ où y_t^{des} est la valeur désirée correspondant à l'entrée $\vec{x}_t$.

Calcul des sorties du réseau en propageant les valeurs de x de couche en couche
Les entrées sont les x_i i=1:p , en supposant que nous avons une entrée $x_0=1$, on pose

$a_j^{(1)} = \sum_{i=1}^{p} \beta_{ji} x_i + \beta_{j0} = \sum_{i=0}^{p} \beta_{ji} x_i$ $\quad x_j^{(1)} = \psi\left(a_j^{(1)}\right)$. La sortie est $a_{st}^{(2)} = \sum_{j=0}^{n} \alpha_j x_j^{(1)}$, qui est au fait y (en supposant aussi que pour la couche cachée on a une cellule $x_0^{(1)} = 1$).

Couche de sortie : Calcul de $\frac{\partial S_t}{\partial \alpha_j}$ (pour un exemple fixé)

$$\frac{\partial S_t}{\partial \alpha_j} = \frac{\partial S_t}{\partial a_s^{(2)}} \frac{\partial a_s^{(2)}}{\partial \alpha_j}$$

$$S_t(\theta) = (X_t^{des} - a_s^{(2)})^2$$

$$-2\left(X_t^{des} - a_s^{(2)}\right)$$

$$a_s^{(2)} = \sum_{j=0}^{n} \alpha_j x_j^{(1)} \rightarrow x_j^{(1)}$$

$$Err_t \equiv \frac{\partial S_t}{\partial a_s^{(2)}} = -2\left(X_t^{des} - a_s^{(2)}\right) \quad \text{d'où} \quad \mathbf{\frac{\partial S_t}{\partial \alpha_j} = Err_t . x_j^{(1)}}$$

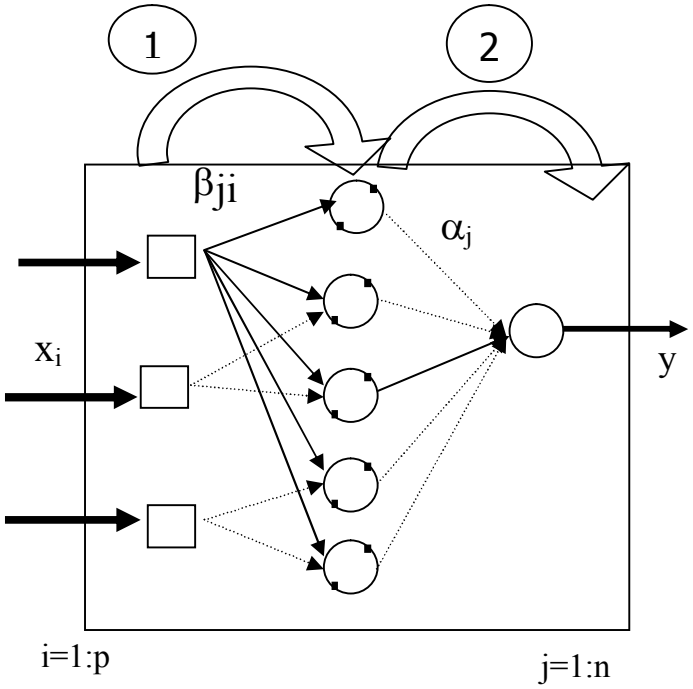


Fig 2.26 Propagation des valeurs de couche en couche

Couche cachée : Calcul de $\frac{\partial S_t}{\partial \beta_{ji}}$ (pour un exemple fixé)

$$\frac{\partial S_t}{\partial \beta_{ji}} = \frac{\partial S_t}{\partial x_j^{(1)}} \frac{\partial x_j^{(1)}}{\partial a_j^{(1)}} \frac{\partial a_j^{(1)}}{\partial \beta_{ji}}$$

$$\frac{\partial S_t}{\partial x_j^{(1)}} = \frac{\partial S_t}{\partial a_s^{(2)}} \frac{\partial a_s^{(2)}}{\partial x_j^{(1)}}$$

$$\frac{\partial S_t}{\partial x_j^{(1)}} = Err_t \alpha_j$$

$$x_j^{(1)} = \psi\left(a_j^{(1)}\right)$$

$$\psi'\left(a_j^{(1)}\right)$$

$$a_j^{(1)} = \sum_{i=0}^{p} \beta_{ji} x_i$$

$$x_i$$

On pose $Err_j \equiv \frac{\partial S_t}{\partial a_j^{(1)}} = Err_t \ \alpha_j \ \psi'\left(a_j^{(1)}\right)$ alors $\frac{\partial S_t}{\partial \beta_{ji}} = Err_j . x_i$

Prenons l'exemple suivant : l'entrée x = [0.5,1] avec comme valeur désirée y_{des} =0.5. Les poids sont donnés par W_1=[0.5, 0.5; 0.5, 0.5] (pas de biais) et W_2=[1,1], on a alors $a^{(1)}$= [0.75,0.75], si on prend comme fonction d'activation la fonction ψ(x) = thx qui vérifie df/dx = 1- f^2(x),on a alors $x^{(1)}$= [0.6351,0.6351] , $a^{(2)}$=1.2703 =y, ce qui donne :Err = 0.7703, $GradW_2$ = [0.4893 , 0.4893], Err_j = [0.6208, 0.6208] et $GradW_1$ =[0.3104,0.6208].

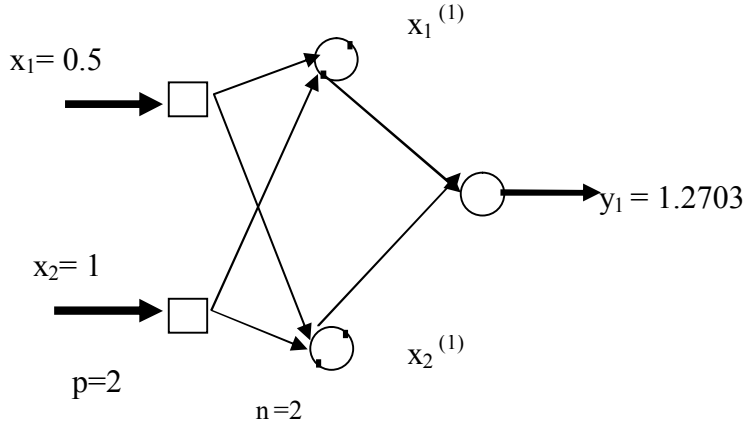


Fig 2.27 Réseau de l'exemple

Si on veut par exemple appliquer l'algorithme du gradient où la mise à jour de w_{ji} et α_j se fait via $W_{new} = W_{old} - \eta \nabla S_T$ Où W_{new} est le nouveau vecteur poids et W_{old} l'ancien vecteur poids, η une constante appelée pas d'apprentissage, alors si on prend η =0.5, les nouvelles valeurs des poids sont w_1 =[0.3448 0.3448 ; 0.1896 0.1896] et w_2 =[0.7554 0.9142 ; 0.7554 0.9142].La nouvelle valeur de y sur la base de ces nouveaux poids est alors y = 0.5242, l'évolution de y en fonction du nombre d'itérations est donnée par le graphe suivant (fig.2.28) où on remarque que si ce dernier croit, y s'approche de la valeur désirée 0.5.

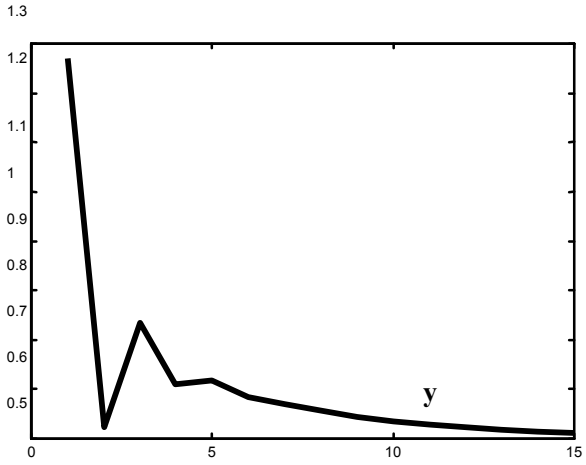


Fig 2.28 Evolution de y

b) Cas d'un perceptron multicouches à plusieurs couches cachées et à une seule sortie

Nous utilisons les notations suivantes (voir Fig 2.29) **:**

- Chaque cellule est définie par un indice,
- y_t^{des} est la sortie désirée sur l'entrée $\vec{x}_t$
- β_{ij} est le poids synaptique associé au lien entre la cellule j vers la cellule i, ce qui implique qu'elles se trouvent sur deux couches successives par définition de l'architecture,
- x_{ij} est l'entrée associée au lien entre la cellule j vers la cellule i,
- Pred (i) est l'ensemble des cellules dont la sortie est une entrée de la cellule i; ceci implique que la cellule n'est pas une cellule d'entrée et que tous les éléments de Pred(i) appartiennent à la couche précédente de celle à laquelle appartient la cellule i,
- y_i l'entrée totale de la cellule i, soit $y_i = \sum_{j \in Pred(i)} \beta ij\ xij$,
- o_i est la sortie de la cellule i, soit $o_i = \psi(y_i)$,
- Succ (i) est l'ensemble des cellules qui prennent comme entrée la sortie de la cellule i, ceci implique la cellule n'est pas une cellule de sortie et que tous les éléments de Succ(i) appartiennent à la couche suivante de celle à laquelle appartient la cellule i.

On va évaluer $\frac{\partial S_t}{\partial \beta_{ij}}$ **.**On a : $$\frac{\partial S_t}{\partial \beta_{ij}} = \frac{\partial S_t}{\partial y_i}\frac{\partial y_i}{\partial \beta_{ij}} = \frac{\partial S_t}{\partial y_i} x_{ij} \qquad (2.74)$$

Il nous suffit donc de calculer $\frac{\partial S_t}{\partial y_i}$ **.** Pour cela, nous allons distinguer deux cas : le cas où la cellule i est une cellule de sortie, et le cas où c'est une cellule interne.

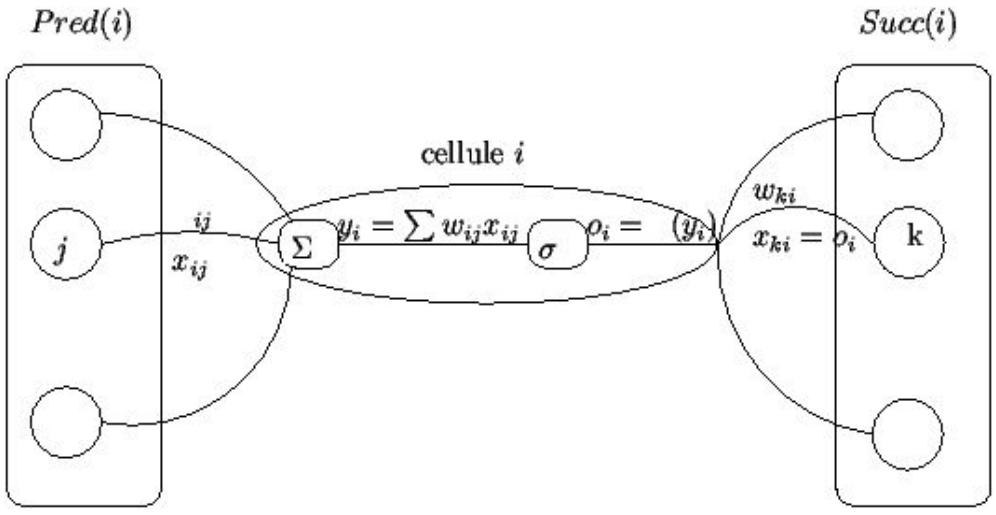


Fig 2.29 Cas d'un perceptron multicouches à plusieurs couches cachées et à une seule sortie

La cellule i est la cellule de sortie

Dans ce cas, la quantité y_i ne peut influencer la sortie du réseau que par le calcul de o_i. Nous avons donc

$$\frac{\partial S_t}{\partial y_i} = \frac{\partial S_t}{\partial o_i}\frac{\partial o_i}{\partial y_i} \qquad (2.75)$$

Nous allons maintenant calculer chacune des deux dérivées partielles apparaissant dans cette équation. Pour la première de ces deux dérivées, nous avons :

$o_i = y_t$ et $\frac{\partial S_t}{\partial o_i} = \frac{\partial}{\partial o_i}(y_t^{des} - y_t)^2$, $o_i = \psi(y_i) = \psi(\sum_{j \in Pred(i)} \beta ij\ xij)$,

Seul le terme correspondant à t=i a une dérivée non nulle ce qui nous donne finalement

$$\frac{\partial S_t}{\partial o_i} = -2\,(y_t^{des} - y_t)^2. \qquad (2.76)$$

Pour la seconde des deux dérivées de l'équation (2.74), en utilisant la définition du calcul de la sortie d'une cellule élémentaire et celle du calcul de la dérivée de la fonction sigmoïde donnée au paragraphe 2.2 , nous avons : $\frac{\partial o_i}{\partial y_i} = \frac{\partial \psi(y_i)}{\partial y_i} = \Psi(y_i)(1 - \Psi(y_i)) = o_i(1 - o_i)$ (2.77)

En substituant les résultats de (2.75) et (2.76) dans l'équation (2.74) nous obtenons :

$$\frac{\partial S_t}{\partial y_i} = -2\,(y_t^{des} - y_t)^2\, o_i(1 - o_i) \qquad (2.78)$$

La cellule i est une cellule interne

Dans ce cas, y_i va influencer le réseau par tous les calculs des cellules de l'ensemble Succ(i).

Nous avons alors : $\frac{\partial S_t}{\partial y_i} = \sum_{k \in Succ(i)} \frac{\partial S_t}{\partial y_k}\frac{\partial y_k}{\partial y_i} = \sum_{k \in Succ(i)} \frac{\partial S_t}{\partial y_k}\frac{\partial y_k}{\partial o_i}\frac{\partial o_i}{\partial y_i} = \sum_{k \in Succ(i)} \frac{\partial S_t}{\partial y_k}\beta_{ki}\, o_i(1 - o_i)$

Soit encore

$$\frac{\partial S_t}{\partial y_i} = o_i(1 - o_i)\sum_{k \in Succ(i)} \frac{\partial S_t}{\partial y_k}\beta_{ki} \qquad (2.79)$$

Grâce aux équations (2.78) et (2.79), nous pouvons calculer toutes les dérivées partielles $\frac{\partial S_t}{\partial y_i}$ pour toute cellule *i*. Le calcul devra être fait pour les cellules de sortie puis des cellules de l'avant-dernière couche jusqu'aux cellules de la première couche. C'est pour cette raison que l'on parle de « rétro propagation».

Grâce à l'équation (2.74), nous pouvons calculer toutes les dérivées partielles $\frac{\partial S_t}{\partial \beta_{ij}}$.

Pour la méthode du gradient, on modifie les poids synaptiques par $\Delta\beta_{ij} = -\varepsilon\,\frac{\partial S(\beta)}{\partial \beta_{ij}}$

On va dans ce qui suit définir l'algorithme de rétro propagation du gradient.

c) Algorithme de rétro propagation du gradient

Pour écrire l'algorithme, nous allons simplifier quelques notations. Nous appelons δ_i la quantité - $\frac{\partial S_t}{\partial y_i}$. En utilisant les équations (2.78) et (2.79), nous obtenons les formules suivantes pour une cellule i de sortie, nous avons : $\delta_i = 2\, o_i(1-o_i)\; o_i\,(1-o_i)$

(2.80)

Pour une cellule i interne, nous avons : $\delta_i = o_i(1-o_i)\sum_{k\in Succ(i)} \delta_k \beta_{ki}$ (2.81)

La modification du poids β_{ij} est alors définie par : $\Delta\beta_{ij} = \varepsilon\; x_{ij}\; \delta_i$ (2.82)

En résumé, l'algorithme suit les étapes suivantes :

(1). Initialiser les poids du réseau

(2). Présenter premièrement les vecteurs d'entrée à partir des données d'apprentissage

(3). Envoyer les vecteurs d'entrée à travers le réseau pour obtenir une sortie

(4). Calculer un signal d'erreur entre la sortie réelle et la sortie désirée

(5). Envoyer le signal d'erreur en arrière à travers le réseau

(6). Corriger les poids pour minimiser l'erreur

(7). Répéter les étapes 2-6 avec le prochain vecteur d'entrée jusqu'à ce que l'erreur soit suffisamment petite ou jusqu'à ce qu'un nombre d'itération maximal fixé soit atteint.

Remarque

Cet algorithme n'est pas un algorithme d'apprentissage, mais un moyen de calculer le gradient de la fonction de coût.

2.4.4.6. Problèmes de l'estimation des paramètres

On résume dans ce qui suit les problèmes pratiques et théoriques liés à la modélisation neuronale lorsqu'on estime les paramètres, sans être exhaustif :

Les minima locaux. Pendant l'apprentissage, puisque le modèle est non linéaire par rapport aux paramètres, la solution trouvée peut être un minimum local. Converger vers le minimum global est un problème difficile dans le cadre d'algorithme déterministe tels que celui du gradient ou des méthodes quasi newtoniennes.

Le sur apprentissage. Du fait des capacités d'approximation universelle des modèles neuronaux, l'apprentissage peut mener à un sur ajustement (overfitting en anglais) de la série. Si un modèle a un grand nombre de paramètres alors qu'il modélise un problème de faible complexité, ce problème a lieu. Il y a alors une mauvaise adéquation entre le modèle et le problème de prévision réel. La fig 2.30 montre l'effet du sur ajustement lors de l'estimation des paramètres. En particulier, on remarque à partir d'un certain nombre d'itérations de la phase d'optimisation, une croissance de la somme des erreurs quadratiques sur des valeurs n'ayant pas servi à l'apprentissage.

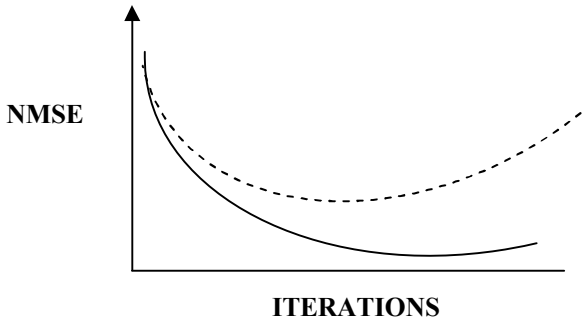


Fig 2.30 En abscisse le nombre d'iterations de la methode d'optimisation utilisée. Le trait plein correspond à la somme des erreurs quadratiques normalisée calculée sur la base d'apprentissage .Le trait en pointillé correspond à ce même critère calculé sur des valeurs de la série non utilisée pour l'apprentissage (base de test).

Dans la partie suivante, on propose une application pratique comme exemple d'optimisation, où on va procéder à la prévision d'une série temporelle en utilisant une méthode classique, en l'occurrence la méthode de Box et Jenkins et le nouvel outil présenté, celui neuronal.

3. APPLICATION PRATIQUE

3.1. Introduction

Les méthodes classiques sont incapables de bien décrire le comportement non linéaire de certaines variables économiques et financières. L'approche neuronale peut résoudre le problème de non linéarité en fournissant à l'utilisateur une grande liberté de conception de modèles et une flexibilité de choix de paramètres. En 1987 Lapedes et Farber réalisent le premier travail qui montra la possibilité d'identifier et de prédire le futur des séries temporelles à l'aide des perceptrons multicouches. Cet article a lancé plusieurs applications à des données réelles, parmi les auteurs on peut citer H.White (1988) qui a étudié le cas de la prévision des retours de stock pour IBM, cet effort a été suivi par Sharda et Patil (1990), qui ont conduit des compétitions de prévision entre les réseaux de neurones et les techniques traditionnelles, en utilisant 75 séries de différentes natures, ils ont aboutit à des résultats qui favorisent les réseaux de neurones : sur 75 séries testées, les réseaux de neurones ont été plus performants pour 39 séries, et ils ont été moins performants pour 36 séries. Plus tard des auteurs comme Tang, Fishwick, Chatfield et Faraway (1996) ont voulu, avec précaution, établir un lien entre les modèles classiques et les modèles connexionnistes, surtout dans l'identification des paramètres, et de l'architecture pour un modèle. Mais on reproche aux réseaux de neurones le fait qu'ils fournissent des réponses, mais pas des explications, il s'agit d'une "boîte noire", parce qu'il est complètement impossible à inspecter, c'est comme si on voulait examiner le cerveau de quelqu'un pour savoir ce qu'il pense: on ne peut inspecter et visualiser que les prédictions faites. D'autre part, les méthodes d'analyse classique sont critiquées car elles reposent sur des hypothèses imposant des restrictions, comme celle de linéarité et l'hypothèse de stationnarité des données étudiées. La démarche du prévisionniste restera la même, il doit passer par l'analyse, l'identification, l'estimation, la validation et enfin la prévision, sauf que les techniques utilisées diffèrent d'une approche à une autre dans quelques étapes. Dans ce qui suit, nous allons prendre comme méthode de prévision l'apprentissage supervisé qui est la caractéristique la plus importante des réseaux de neurones. Il est équivalent à l'estimation des paramètres du modèle neuronal et qui n'est que la convergence d'un algorithme itératif vers un point stable. Pour ce qui est de la validation,

l'analyse des résidus reste toujours un point commun entre les différentes approches utilisées. Cependant, le problème majeur des réseaux de neurones est la détermination de l'architecture du modèle à estimer. Pour résoudre ce problème, l'analyste peut utiliser les compétences traditionnelles en modélisation telles que la méthodologie Box-Jenkins pour sélectionner un bon modèle neuronal.

Nous allons nous intéresser à travers des données réelles à la prévision d'une série chronologique par la méthode de Box-Jenkins et celle des réseaux de neurones artificiels. Nous allons comparer les performances de ces deux approches sur les mêmes données.

3.2. Prévision par les perceptrons multicouches

3.2.1. Prétraitements

D'après Vares et Versino en 1990, il n'est pas nécessaire de stationnariser la série avant de chercher à la prévoir en utilisant les réseaux de neurones artificiels qui sont des modèles non linéaires. Mais on peut effectuer des prétraitements dans le but de supprimer de la série tout ce qui est modélisable par des techniques simples, car il paraît inutile de demander au réseau de prévoir des aspects tout à fait modélisables de la série temporelle, ce genre de prétraitement permet de réduire la complexité du réseau et accélérer sa convergence.

3.2.1.1. La Normalisation La normalisation des données est le prétraitement le plus important pour les réseaux de neurones. On trouve dans la littérature plusieurs types de normalisation.

Nous citons les deux méthodes les plus courantes, la première est exprimée par

$$x_1 = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

avec x valeur de la série temporelle à normaliser, x_{min} valeur minimale de l'ensemble de données utilisées pour la normalisation, et x_{max} est la valeur maximale. La deuxième méthode de normalisation des données est exprimée par

$$x_1 = \frac{x - \mu_x}{\sigma_x}$$

Où μ_x est la valeur moyenne des données utilisées pour la prévision, et σ_x l'écart type.

3.2.1.2. Traitement logarithmique Pour de nombreuses séries, il est conseillé d'opérer à un prétraitement logarithmique, il suffit d'associer à chaque valeur de la série temporelle x_t une valeur x_0 tel que : $x_1 = \ln x_t$:

3.2.2. La Non linéarité

Il est important d'élaborer une analyse empirique des données parce qu'elle permet de déterminer d'une manière efficace une distribution adaptée aux données. La forme de l'histogramme donne une idée importante sur la fonction de répartition des données. L'hypothèse de normalité sera avancée si la forme de l'histogramme est une cloche symétrique. D'après Azencott et Dacunha-Castelle en 1994, si une famille d'observations suit une loi gaussienne, alors les techniques de prévision linéaires sont convenables. Après le test de l'hypothèse de normalité des observations, les réseaux de neurones grâce à leur apport de non linéarité peuvent résoudre le problème et apporter un plus dans la prévision. L'objectif essentiel de l'utilisation des réseaux de neurones artificiels n'est pas la modélisation elle même mais plutôt la prévision des valeurs futures d'une série temporelle, donc on peut se confier à un modèle non paramétrique de type boîte noire.

3.2.3. Construction d'un modèle avec les réseaux de neurones

La conception d'un modèle neuronal consiste à choisir un nombre de paramètres avant toute expérimentation du modèle. D'abord il faut choisir le nombre de couches, généralement pour les problèmes de prévision on conçoit un modèle à trois couches: la couche d'entrée, la couche cachée et enfin celle de sortie. Ensuite on définit le nombre de neurones pour chaque couche, ce dernier sera optimisé lors de la sélection du modèle. Par la suite il faut choisir l'algorithme d'apprentissage approprié ainsi que les paramètres qui lui sont relatifs tels que le taux d'apprentissage et le critère d'arrêt (un nombre d'itérations maximal ou un seuil accepté pour la fonction coût). On choisit comme fonctions de transfert, celles sigmoïdes vues au chapitre 2. Il existe deux approches différentes de modélisation des séries temporelles avec les réseaux de neurones artificiels, la première est l'approche de la prévision multi-pas en avant, la deuxième est celle de la prévision à un pas de temps (one step ahead forecasting).Le premier type de modèles est caractérisé par un nombre d'unités de sortie supérieur ou égale à 2 tandis que la prévision à un pas de temps, qui est le modèle le plus utilisé dans la littérature, utilises le principe suivant :

A partir des valeurs $X_{t\text{-}lag\ 1}$, $X_{t\text{-}lag2}$ …. $X_{t\text{-}\ lag\ max}$, lag_{max} étant le retard le plus élevé de neurones d'entrées, on veut estimer la valeur de X_t: Par exemple, dans la phase de l'apprentissage une architecture de la fig 3.1 peut être utilisée. Après avoir estimé le modèle, c'est-à-dire abouti à des poids de connexions optimaux pour minimiser l'erreur commise par le réseau, (qui revient à minimiser la somme des carrés des erreurs de prévision à l'intérieur de l'échantillon), la phase de l'utilisation proprement dite du réseau servira à calculer des prévisions hors échantillon. Puisque notre modèle contient une seule unité au niveau de la couche de sortie, cela ne veut pas dire qu'on est restreint à un seul pas de prévision hors échantillon, il est possible de raccorder une simple récurrence de la couche de sortie vers la couche d'entrée, si on voulait prévoir d'autres valeurs dans le futur. La Figure 3.1 donne un exemple. Cette récurrence n'affectera pas le modèle initial parce qu'elle est introduite seulement dans la phase d'utilisation du réseau c'est à dire la prévision hors échantillon.

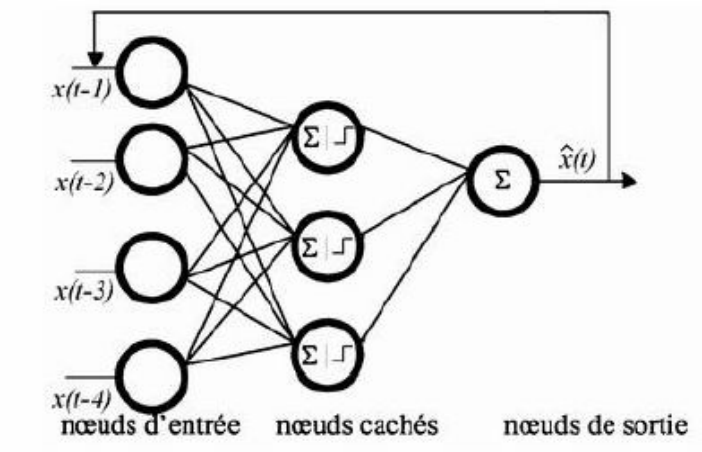


Fig 3.1 Exemple d'un modèle pour la prévision à un pas de temps en avant.

3.2.3.1. Choix d'une architecture L'optimisation de l'architecture a pour but de réduire le nombre de synapses (poids) et de neurones afin de réduire la complexité du réseau, améliorer les temps de calcul et maintenir les capacités de généralisation. En effet, si on utilise par exemple un nombre trop faible de neurones on obtiendra des mauvaises performances en apprentissage et en généralisation, et d'un autre coté si on utilise un nombre très grand de neurones, il se provoquera une limite de la capacité de généralisation, ce que l'on appelle "l'apprentissage par coeur". Concernant l'optimisation de l'architecture du réseau, deux principales approches ont été proposées dans la littérature, à savoir l'approche par sélection (approche destructive) et l'approche incrémentale(approche constructive).

L'approche par sélection L'approche par sélection ou approche destructive consiste à commencer par la construction d'un réseau complexe qui contient un grand nombre de neurones, puis essayer de réduire le nombre de neurones inutiles et supprimer les connexions redondantes pendant ou à la fin de l'apprentissage.

L'approche incrémentale Dans l'approche incrémentale ou approche constructive, et à l'inverse de l'approche précédente, on commence par un réseau le plus simple possible, puis on y ajoute des neurones ou des couches, jusqu'à avoir une architecture optimale. On peut citer la méthode de Fahlman en 1989 qui commence par un réseau initialisé à un seul neurone. Une fois le neurone bien entraîné, une nouvelle unité sera ajoutée et ainsi de suite. L'inconvénient de cette approche est qu'elle induit à un réseau ayant un nombre de couches élevé.

Analogie avec les méthodes classiques Le choix d'une architecture du réseau est un problème fondamental. Les techniques statistiques classiques telles que la modélisation de Box-Jenkins disposent de plusieurs méthodes fiables qui aident à identifier les ordres de modèles, par exemple pour un ARMA(p; q) p et q peuvent être identifiés à partir du corrélogramme d'autocorrélation (ACF) et celui de l'autocorrélation partielle(PACF). Dans l'approche connexionniste le choix de l'architecture (nombre de neurones d'entrée, et les retards qu'ils présentent, le choix de nombre d'unités cachées et autres paramètres) est plus délicat. Dans la plupart des travaux empiriques, les auteurs essaient plusieurs combinaisons de paramètres aléatoirement jusqu'à ce qu'ils arrivent à trouver un modèle performant, d'autres auteurs

3.2.4. Apprentissage et Généralisation

La généralisation est la capacité d'un réseau entraîné sur une base d'apprentissage à répondre correctement à des données non présentées lors de l'apprentissage.

Le sur ajustement est l'ajustement trop important des données conduisant à une mauvaise généralisation (overfitting en anglais). Le sur apprentissage est le fait de sur apprendre les données de la base d'apprentissage (bruit inclus), ce qui conduit à un sur ajustement. La base de généralisation ou base de test est la base de données formée d'observations non utilisées lors de l'apprentissage du réseau et totalement indépendantes de la structure et des poids du réseau. Cette base sert à estimer la capacité de généralisation du réseau de neurones et dans

notre cas, estime l'efficacité de notre prédiction par rapport à des situations futures. Comment éviter le sur apprentissage ?

Technique Early Stopping Un troisième ensemble des données (ensemble de validation) est donc nécessaire pour pouvoir comparer deux prédictions issues d'architectures différentes mais construites sur le même ensemble « d'apprentissage ».Dans Bishop (1995) « Neural Networks for Pattern Recognition » de nombreuses solutions sont proposées pour comparer et évaluer différents fonctions de prédiction issues d'architectures neuronales différentes. Dans la stratégie Early Stopping, on divise l'ensemble d'apprentissage Le en deux ensembles: l'ensemble d'entraînement Tr qui permet de calculer les pondérations θ du réseau, l'ensemble de validation Vl qui permet d'arrêter le processus d'apprentissage (voir Fig 3.2). La convergence est déclarée quand la fonction de coût, évaluée sur Tr croit sur Vl. L'ensemble test Ts mesure la qualité de la prédiction choisie. Il sert à calculer une erreur indépendante des données qui ont servi à faire (et à arrêter) l'apprentissage. Par exemple, pour un ensemble de 1550, l'ensemble de données peut être divisé en trois parties: l'ensemble d'apprentissage Le (1000) et l'ensemble test Ts (550). L'ensemble d'apprentissage est alors divisé en deux : l'ensemble d'entraînement Tr (700) et l'ensemble de validation Vl (300).Ainsi, on vient de voir un moyen d'arrêter l'apprentissage avant le sur-apprentissage en arrêtant l'algorithme avant d'avoir atteint le minimum.

3.2.5. Evaluation de la performance et sélection d'un modèle

L'approche statistique usuelle pour la sélection du modèle est d'estimer les erreurs de généralisation de différents modèles candidats en sélectionnant celui pour lequel on a le minimum de ces estimateurs. Plusieurs estimateurs complexes ont été développés parmi lesquels on a retenu deux approches :

3.2.5.1. Validation croisée Une approche efficace consiste à diviser l'ensemble des données en deux moitiés (les entrées sont alors choisies au hasard et rangées dans chaque moitié). On construit le prédicteur à l'aide de la première moitié des données, puis on estime l'erreur de prédiction à l'aide de la seconde. On recommence cette opération en échangeant le rôle des deux moitiés.

En moyennant, les deux résultats, on diminue la variance de l'erreur de prédiction estimée. L'inconvénient de cette méthode est que l'on obtient ainsi une estimation de l'erreur de

prédiction n'utilisant que la moitié des données et le fait qu'il faut faire deux apprentissages au lieu d'un seul.

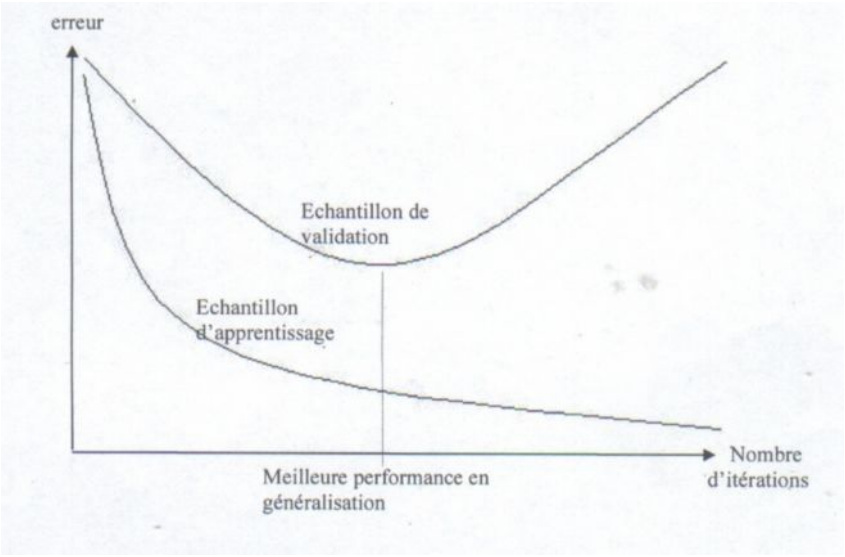


Fig 3.2 On s'arrête quand l'erreur remonte "trop" sur l'ensemble de validation

3.2.5.2. Utilisation des critères AIC et BIC Rappelons les critères d'information d'Akaïke $AIC(P) = N\,Log\frac{E}{N} + 2P$ et le critère d'information Bayesien $AIC(P) = N\,Log\frac{E}{N} + P + P\,Log\,N$ où P représente le nombre de paramètres estimés, N le nombre d'observations utilisées pour ajuster le modèle, et où E est la somme des carrés des résidus. Le nombre de paramètres P à estimer dans le modèle correspond au nombre total de poids des connexions entre neurones. P est bien sur beaucoup plus grand que celui utilisé pour les méthodes traditionnelles. Pour les réseaux de neurones à trois couches (à une seule couche cachée) et à une seule unité de sortie, P est donné par P = (p + 2) n + 1 où p est le nombre de variables d'entrées en excluant la constante (cette entrée est utilisé pour les biais de chaque couche), et n désigne le nombre de neurones cachés. Comme P est grand, on pourrait signaler qu'il existe un danger que l'algorithme d'apprentissage surexploite les données et réalise un ajustement artificiellement bon mais ne donnant pas de meilleures prévisions. C'est pourquoi il est très important d'utiliser les critères AIC et BIC: ces deux critères pénalisent l'ajustement de paramètres supplémentaires.

3.3. Etude empirique

3.3.1. Introduction

Les données historiques qui ont servi de base à la prévision sont constituées d'une chronique réelle représentant la série ventes électricité pour des clients ordinaires (BT ou de basse tension) enregistrées de Janvier 1984 à Septembre2005 (261 observations, l'unité étant le Kilowattheure (KWH)) obtenues au niveau du siège SONELGAZ Blida qui a sous son actif cinq wilayas Blida, Tizi Ouzou, Djelfa, Bouira, et Médéa. L'observation de ces données montre que les ventes électricité sont des quantités chiffrées qui varient dans le temps. Ce constat nous amène à faire appel à la théorie sur les processus aléatoires et sur les techniques d'analyse des séries chronologiques. Soit alors $(X_t)_{t\in T}$ le processus traduisant les ventes électricité. On s'intéresse alors à ce processus. Pour chaque wilaya citée ci-dessus, on dispose d'une série mensuelle : la série ventes Electricité VE qui représente au fait X_t , la réalisation du processus c'est-à-dire les chiffres observés des ventes à l'instant t. T : Base des temps : T={janvier 1984,…, Septembre2005 }.On a choisi la région de Djelfa et sur cette base, dans ce qui suit, la série des ventes d'électricité dans la wilaya de Djelfa sera notée VED. L'électricité est une source d'énergie qui a vu une demande croissante et la production dans ce secteur obéit à des impératifs économiques, démographiques et météo logiques. L'état devant ce fait a investi en réalisant de nouvelles centrales ou en optant pour l'expansion des anciennes. La wilaya de Djelfa qui a connu une croissance et un développement considérables les dernières années, a enregistré une croissance de consommation en électricité.

Dans ce travail, il est question de faire une prévision dite de court terme, les méthodes utilisées dans ces prévisions sont des techniques d'auto projection à partir de l'analyse de séries temporelles. Les prévisions sont calculées à partir de valeurs mensuelles. Des chiffres historiques mensuels sont recueillis sur un certain nombre d'années pour faire une prévision à un horizon d'un an au maximum, étant donnée la rigueur historique qu'implique le temps. Tout d'abord, il faut remarquer qu'il s'agit d'un modèle à une seule variable : le temps. C'est le temps qui permet de déterminer l'évolution de la consommation d'électricité à court terme. Ensuite, il faut assumer que le passé conditionne l'avenir. C'est pourquoi la prévision est limitée à un horizon d'un an. Les méthodes de prévision à court terme prennent un aspect particulier car en général on ne fait pas intervenir de variables extérieurs et on explique la série par ses valeurs antérieures. Les techniques d'auto projection comprennent un grand nombre de techniques : Le lissage exponentielle (Brown, 1962) et plus particulièrement la

méthode de Holt-Winters qui constitue un outil très utilisé pour ce qui est de la prévision. Il existe également un type de techniques plus sophistiquées, ce sont les modèles ARIMA de Box et Jenkins (1970) et qui représentent l'idéologie dominante actuelle. Ce sont des modèles très intéressants, car ils sont particulièrement indiqués pour les cas où la série temporelle présente une tendance et un caractère saisonnier.

Dans ce travail, nous nous sommes intéressés particulièrement à l'analyse Box-Jenkins et aux réseaux de neurones artificiels. Nous allons comparer les performances des deux approches indépendamment sur les mêmes données.

La figure 3.3 suivante nous fournit une représentation de l'échantillon à étudier sous forme d'histogramme afin de décrire ses propriétés globales.

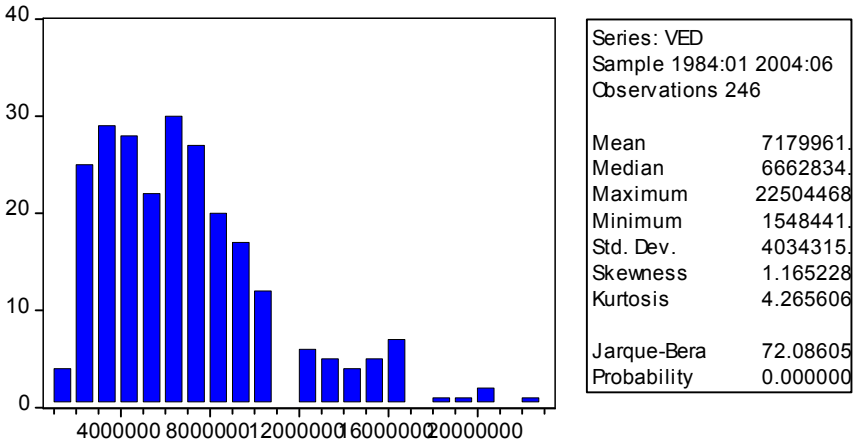


Fig 3.3 Statistiques descriptives de l'échantillon

Nous remarquons que la distribution est asymétrique étalée à droite. De plus, elle est leptokurtique. La p-valeur de probabilité associée au test de Jarque-Bera confirme la non admission de l'hypothèse nulle de normalité.

3.3.2. Représentation Linéaire et Traitement par la méthode de Box et Jenkins

Nous allons adopter comme échantillon qui sera utilisé dans l'estimation du modèle, les données entre Janvier 1984 et Juin 2004, les quinze observations qui restent (de Juillet 2004 à Septembre 2005 seront utilisées pour évalue r le pouvoir prédictif issu du modèle qui sera estimé à partir des observations antérieures.

Pour chaque année, on calcule la moyenne des observations, on a 20 années complètes et 6 mois pour l'année 2004 donc 20 moyennes sur 12 mois et 1 sur 6 mois, ce qui donne le

tableau A.4.et les deux graphes A.5 et A.6 de l'annexe A. Ces deux graphes nous donnent un aperçu sur la nature de la série. Ainsi, nous constatons que la moyenne et la variance varient avec le temps**.** Le graphe ci-dessous (Fig 3.4) de la série brute VED montre une tendance faible au début puis croissante ainsi que des fluctuations régulières : la série ne peut être stationnaire, ce qui est confirmé par les corrélogrammes (Fig. A.7 de l'annexe A) qui présentent des pics significatifs aux retards 12, 24 et 36 signalant l'existence d'une composante saisonnière, mais surtout par les auto corrélations qui sont toutes significatives et qui décroissent très lentement (impliquant une tendance en moyenne), ce qui laisse supposer une non stationnarité.

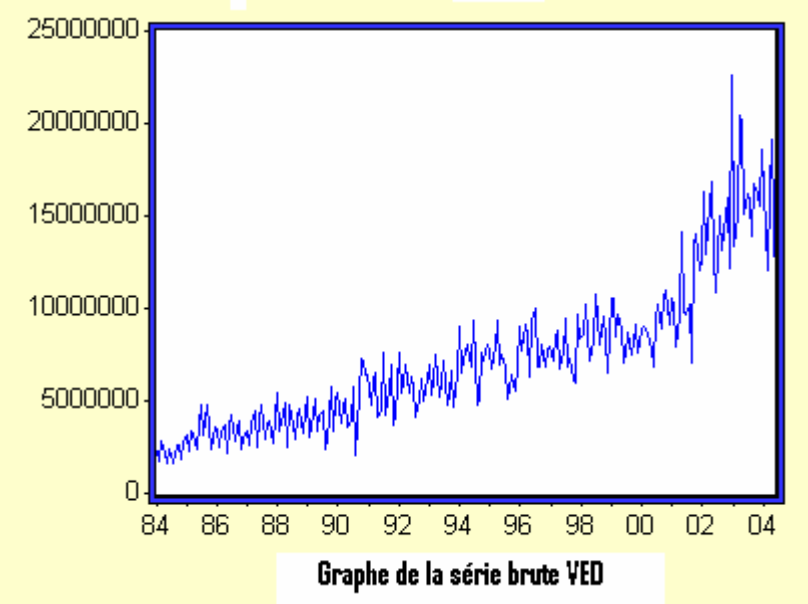


Fig 3.4 Graphe de la série VED

Test de saisonnalité Le test classique en cette matière est le test de Fisher.

-test de saisonnalité

H_0 **:** « Pas d'influence du facteur mois » contre H_1 : « Il y a influence du facteur mois ».

On considère n: Le nombre d'années = 22 ; p: Le nombre d'observations dans l'année=12 ; X_{ij} la valeur de la série pour la $i^{ème}$ année et la $j^{ème}$ période (on a 261observations) ; la moyenne générale $\overline{X}_{..}$ = 7607358, la moyenne de l'année i $\overline{X}_{i.}$ i=1…22 et la moyenne de la période j, $\overline{X}_{.j}$,j=1…12. Ces trois moyennes ont été calculées. Les variances année var_A, période var_p et celle résiduelle var_R sont définies respectivement par :

$$var_A = \frac{p\sum_{i=1}^{n}(\overline{X}_{i.} - \overline{X}_{..})^2}{n-1} \qquad var_p = \frac{n\sum_{j=1}^{p}(\overline{X}_{.j} - \overline{X}_{..})^2}{p-1} \quad \text{et} \quad var_R = \frac{\sum_{i=1}^{n}\sum_{j=1}^{p}(X_{ij} - \overline{X}_{i.} - \overline{X}_{.j} - \overline{X}_{..})^2}{(n-1)(p-1)}$$

Ces trois variances ont été calculées ainsi que la valeur calculée $F_0 = \frac{var_p}{var_R}$ que l'on a comparée à la valeur tabulée $F^{\alpha}_{v_1 v_2}$ avec v_1= (p-1)=11, v_2= (n-1)(p-1) =11×21=231 degrés de liberté, on a

trouvé la valeur suivante $F_0 = \frac{var_p}{var_R} = F_{0\ calculée} = 27.78$, alors que $F_{tab} = F^{\alpha}(11,231) = 1.82$

$F_{calculée} > F_{tab} \Rightarrow$ la série VED est saisonnière.

Pour ôter le facteur saisonnalité qui est de 12 (1 an), on applique un filtre différence saisonnière (∇_{12}=1-B^{12}) à la série VED, on obtient la série SVED (voir Fig 3.5). Le graphe de SVED semble indiquer que la série en différence saisonnière est stationnaire (la tendance parait être éliminée), comme on le voit à travers le corrélogramme de SVED qui ne présente aucune structure particulière (voir Fig A.8 de l'annexe A), on va confirmer ceci par de le test de racine unité ou test de racine unitaire de Dickey Fuller augmenté (ADF)

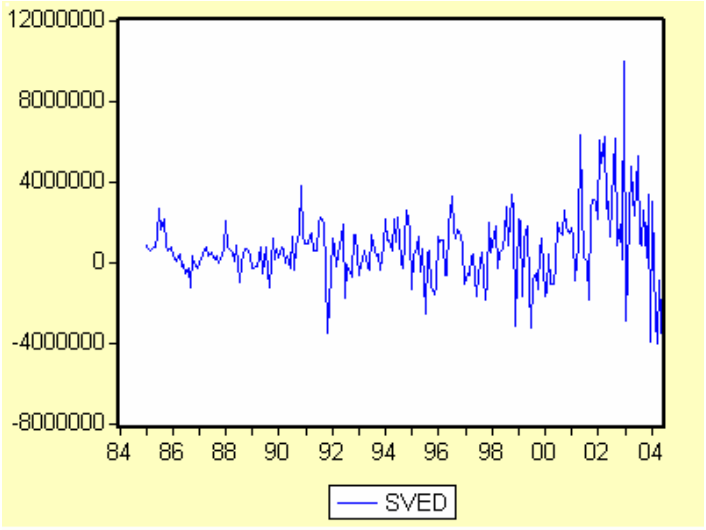


Fig 3.5 Graphe de SVED = (1-B^{12}) VED

Résultats du test de stationnarité sur la série transformée

L'application du test de racine unité sur la série SVED (voir annexe A.2) fait ressortir que la valeur observée de la statistique de Student égale à **-3.797434** est inférieure aux valeurs

critiques relatives à **1%**, **5%** et **10%** qui sont respectivement -2.5748-1.9411 et -1.6164**,** donc on rejette H_0 : « Il existe une racine unitaire ».

Conclusion :

La série SVED est stationnaire, on passe à la modélisation de la série par la méthode Box & Jenkins.

Enfin, l'analyse des coefficients d'auto corrélation et d'auto corrélation partielle (Fig 3.9) de cette série terminale SVED fait apparaître une décroissance de $\hat{\tau}_k$ ($\hat{\tau}_k$ étant l'auto corrélation partielle) comprise dans une enveloppe du type E (définie au chapitre1 en 1.3.3.2**).**

Identification et estimation du modèle

On doit ajuster à notre série un modèle SARIMA (p, d, q) (P, D, Q) avec s=12. A partir des fonctions d'auto corrélation empiriques ρ_k et d'auto corrélation partielle empiriques α_k de SVED, on peut trouver P et Q en observant $(\rho_{12k})_k$ et $(\alpha_{12k})_{k,k=1,2\ldots}$.On voit que seul ρ_{12} est significativement non nul, donc Q=0 ou 1 et $\alpha_{12} \neq 0$, donc P=0 ou 1. On n'a pas différencié VED, donc d=0 et on a appliqué l'opérateur de différentiation saisonnière une fois, donc D=1. Au vu du corrélogramme de SVED on peut penser à prendre q=1,2,…7 et le corrélogramme partiel de SVED nous suggère de prendre p=1, 2, 3, 4.
L'étape d'identification conduit alors à retenir plusieurs modèles candidats pour la série SVED. Ce sont les modèles SARIMA (p, 0; q) (P, 1; Q) p et q telles que p∈ {0,1, 2, 3,4}, q∈{0,1, 2, 3,4, 5, 6,7}, P∈ {0,1} et Q∈ {0,1. La phase d'estimation utilise des techniques classiques de statistique telles que l'approche du maximum de vraisemblance ou la technique des moindres carrés. Nous trouvons plusieurs logiciels spécialisés qui nous aident à réaliser ce travail d'estimation en donnant directement les valeurs estimées et toutes les statistiques nécessaires aux différents tests, ces modèles ont été tous estimés par l'intermédiaire du logiciel Eviews et ont donné les résultats suivants :
Onze (11) parmi eux ont été rejetés et ce, souvent à cause de la non significativité des paramètres du modèle, la singularité de la matrice d'estimation et la non inversibilité du processus moyenne mobile ou de la non stationnarité du processus autorégressif (inverses des racines unitaires à l'extérieur du cercle unité). Dans le tableau A.9 de l'annexe A, nous

récoltons les résultats pour seulement les modèles dont les coefficients sont significatifs. Ces modèles candidats sont en nombre de 41, les résultats contiennent les critères d'information: le critère d'Akaïke (AIC), le critère d'information Bayesien BIC, la somme des carrés des résidus SSE, la statistique R^2 et la celle de Durbin-Watson.

Le choix du modèle optimal

On en conclut que le meilleur modèle est le modèle 38 qui est le SARIMA (4; 0; 3) (0;1;1) avec $\varphi_1 = \varphi_2 = 0$, la sélection est faite grâce au critère d'Akaïke qui a la plus petite valeur par rapport aux autres modèles, ainsi que la somme des carrés des résidus(RSS) qui est la plus faible, mais aussi grâce à R^2 qui est la plus grande valeur pour ce modèle. Le modèle sélectionné SARIMA
(4; 0; 3) (0;1;1) (où la constante est significative) est exprimé par l'expression suivante :

$$\mathbf{(1-\varphi_3 B^3 -\varphi_4 B^4)\ (1-B)^0\ (1-B^{12})\ VED_t = c + (1-sma(1)B^{12})[1-ma(1)B-ma(2)B^2-ma(3)B^3)]\varepsilon_t}$$

Où B est l'opérateur retard. Son estimation est donnée dans le tableau A10 de l'annexe A.

Validation du modèle

Test sur les résidus de Box-Lung

Il convient maintenant d'analyser les résidus à partir de leur corrélogramme (Fig. A.12).
On teste l'hypothèse nulle : Les auto corrélations au pas K ne sont pas significatives **:**
H_0 : « $\rho_1 = \rho_2 = \ldots = \rho_K = 0$ » contre H_1 : « $\exists\ \rho_j$ j=1,...K telle que $\rho_j \neq 0$ » Si $Q < \chi^2_{K-p-q-P-Q}$, on accepte l'hypothèse H_0 que les résidus sont non corrélés, K= nombre de retards choisis p,q, P et Q étant les paramètres significatifs du SARIMA . On voit que ρ_K présente deux pics pour K=13 et K= 22 : pour K =13 Q-Stat =11.812 et $\chi^2_{13-6} = \chi^2_7 = 14.1$ donc Q-Stat < χ^2_7 et le pic en 13 n'est pas significatif mais en 22, Q-Stat = 26.863, $\chi^2_{22-6} = \chi^2_{16} = 26.3$, donc Q-Stat > χ^2_{16} , on n'accepte pas l'hypothèse H_0, ce qui implique que les résidus ne forment pas un bruit blanc : on cherche un autre modèle.
Le modèle 37 qui correspond à un modèle SARIMA (3; 0; 3) (0, 1,1) avec $\varphi_1 = \varphi_2 = 0$ est le meilleur parmi les modèles restants, avec la plus petite valeur du BIC. Il s'écrit : $(1-\varphi_3 B^3)\ (1-B)^0\ (1-B^{12})\ VED_t = c + (1-sma(1)B^{12})[1-ma(1)B-ma(2)B^2-ma(3)B^3)]\varepsilon_t$.

Il y a un pic en k=22, et pour k=22, la Q-stat = 27.029, et $\chi^2_{22-5} = \chi^2_{17} = 27.6$ (on a 5 coefficients significatifs) donc Q-Stat < χ^2_{17} donc ce pic n'est pas significatif et on peut confirmer que les résidus forment un bruit blanc. L'estimation de ce modèle est donnée dans le tableau A11 de l'annexe A et la fig. 3.6 ci-dessous représente simultanément les valeurs de la série lissée et réelle ainsi que les résidus.

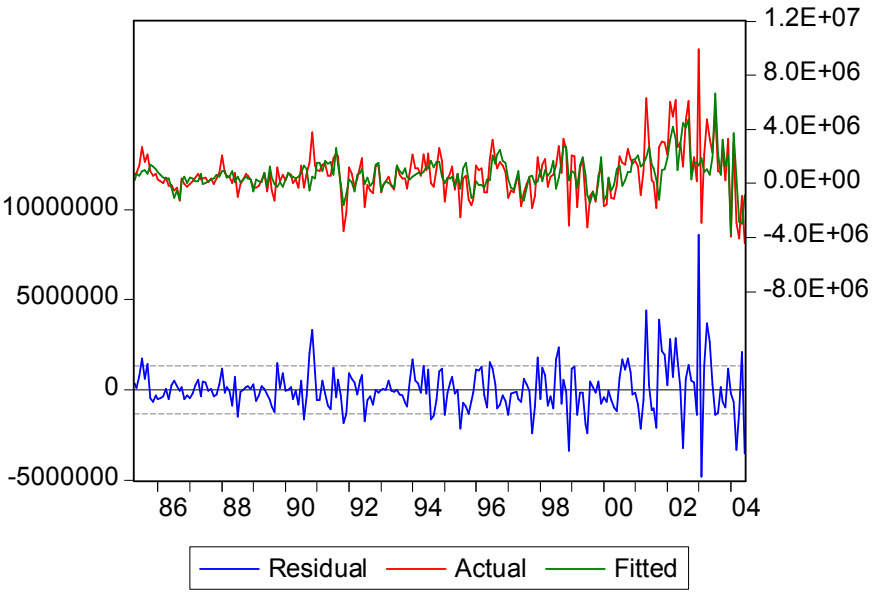


Fig.3.6 Estimation du modèle

Test de normalité de Jaque et Béra

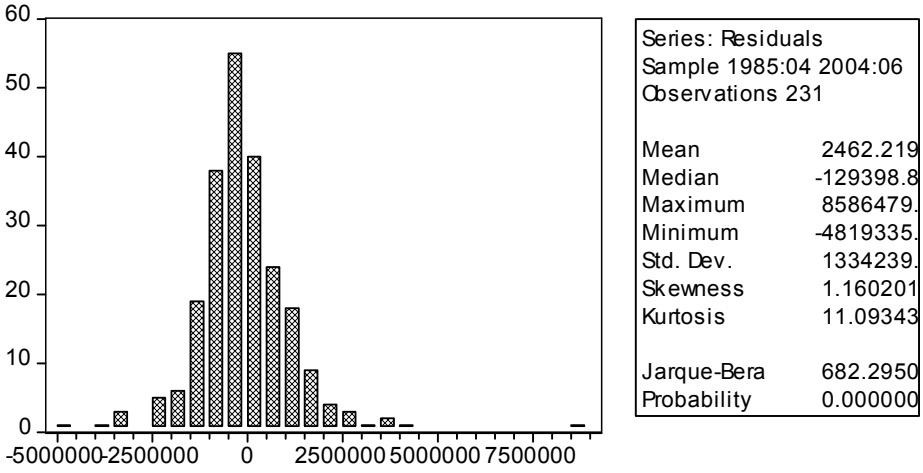


Fig.3.7 Résultats du test de Jaque et Béra

Ni la skewness, ni la kurtosis ni la statistique de Jaque et Bera ne reflètent la normalité des résidus, ceux-ci ne sont donc pas gaussiens.

Test d'hétéroscédasticité

Tel que le montre le corrélogramme des résidus carrés (voir Fig.A.13), le pic en 1 est significatif donc les résidus au carré sont hétéroscédastiques ce qui se traduit peut être **par effet ARCH, on va alors effectuer le test ARCH. Les résultats de ce test le confirment :**

F-statistic	15.13117	Probability		0.000132
Obs*R-squared	14.31395	Probability		0.000155
Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	1.35E+12	3.79E+11	3.550785	0.0005
RESID^2(-1)	0.251491	0.064653	3.889880	0.0001

Tab.3.1 Résultats du test ARCH

La probabilité de F-statistic 0.000132 < 0.05 et le coefficient et la constante sont significativement non nuls, donc les erreurs subissent un effet ARCH de l'ordre 1. La modélisation SARIMA (3; 0; 3) (0;1;1) avec erreurs GARCH (0,1).a donné :

	Coefficient	Std. Error	z-Statistic	Prob.
C	686976.8	234938.2	2.924075	0.0035
AR(3)	0.897602	0.070632	12.70807	0.0000
MA(1)	0.359158	0.097048	3.700846	0.0002
MA(2)	**0.146605**	**0.087254**	**1.680197**	**0.0929**
MA(3)	-0.529676	0.092343	-5.735955	0.0000
SMA(12)	-0.851597	0.052729	-16.15034	0.0000
Variance Equation				
C	2.08E+12	3.01E+11	6.909167	0.0000
ARCH(1)	**0.192838**	**0.127093**	**1.517303**	**0.1292**

Tab 3.2 Estimation du modèle SARIMA (3; 0; 3) (0;1;1) avec erreurs GARCH (0,1)

Nous avons été contraints d'envisager une autre modélisation des erreurs, ainsi avec des erreurs GARCH (1,0), GARCH (2,0), GARCH (1,1), GARCH (1,2) et GARCH (2,2), le processus moyenne mobile n'est pas inversible et avec des erreurs GARCH (0,1) GARCH (0,2) les cœfficients MA (2), ARCH (1) et ARCH (2) sont significativement nuls. On s'est alors penché sur le modèle EGARCH (p, q) (exponential GARCH) qui est un modèle log-linéaire introduit par Nelson (1991) et qui s'écrit :

$$\mathrm{Ln}\,\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \left(\phi z_{t-i} + \gamma \left[\, |z_{t-i}| - \mathrm{E}|z_{t-i}| \right) \right) + \sum_{j=1}^{p} \beta_j \mathrm{Ln}\,\sigma_{t-j}^2 \qquad \text{où} \qquad z_{t-i} = \frac{\varepsilon_{t-i}}{\sigma_{t-i}}$$

Pour un modèle EGARCH (1,1) par exemple,

$$\text{Ln}\,\sigma_t^2 = C + \alpha\left|\frac{\varepsilon_{t-1}}{\sigma_{t-1}}\right| + \gamma\frac{\varepsilon_{t-1}}{\sigma_{t-1}} + \beta\sigma_{t-1}^2 \qquad (3.1)$$

Notre modélisation SARIMA (3; 0; 3) (0,1,1) avec erreurs EGARCH (0,1) a donné les résultats suivants:

	Coefficient	Std. Error	z-Statistic	Prob.
C	968015.2	158683.8	6.100279	0.0000
AR(3)	0.895762	0.027979	32.01499	0.0000
MA(1)	0.412186	0.044075	9.352009	0.0000
MA(2)	0.178714	0.037860	4.720370	0.0000
MA(3)	-0.513216	0.031873	-16.10201	0.0000
SMA(12)	-0.876328	0.016057	-54.57539	0.0000
Variance Equation				
C	27.44622	0.110074	249.3424	0.0000
|RES|/SQR[GARCH](1)	0.793573	0.086561	9.167798	0.0000
RES/SQR[GARCH](1)	-0.140616	0.070306	-2.000050	0.0455
R-squared	0.408573	Mean dependent var		686936.3
Adjusted R-squared	0.387260	S.D. dependent var		1782002.
S.E. of regression	1394911.	**Akaike info criterion**		**30.93854**
Sum squared resid	4.32E+14	**Schwarz criterion**		**31.07266**
Log likelihood	-3564.401	F-statistic		19.17038
Durbin-Watson stat	2.258177	Prob(F-statistic)		0.000000

Tab 3.3 Estimation du modèle élu : SARIMA (3; 0; 3) (0,1,1) avec $\varphi_1 = \varphi_2 = 0$ et erreurs EGARCH (0,1)

Pour ce modèle, on a :

$$\text{Ln}\,\sigma_t^2 = C + \alpha\left|\frac{\varepsilon_{t-1}}{\sigma_{t-1}}\right| + \gamma\frac{\varepsilon_{t-1}}{\sigma_{t-1}} = 27.44622 + 0.793573\left|\frac{\varepsilon_{t-1}}{\sigma_{t-1}}\right| - 0.140616\frac{\varepsilon_{t-1}}{\sigma_{t-1}}$$

On remarque que ce modèle affiche les meilleures valeurs du BIC (31.07266) et du AIC (30.934).

Conclusion :

On adopte le modèle SARIMA (3; 0; 3) (0;1;1) avec une modélisation EGARCH (0,1) de la variance conditionnelle des erreurs.

Prévision

Il existe plusieurs critères pour mesurer la performance de la prévision. Ci-dessous nous citons les critères classiques de pertinence de prévisions, que nous allons utiliser dans la suite. Notons les observations (Xi) i=1,...., n et les prévisions ($\hat{X}_i$) i=1,...., n.

Nous pouvons calculer : L'erreur quadratique moyenne. L'erreur quadratique moyenne (Mean Square Error : MSE) mesure la moyenne des carrés des écarts entre les valeurs observées de la série et leurs valeurs prédites, c'est le critère le plus utilisé car il est facile à manipuler et il permet de pénaliser les grandes erreurs plus que les petites en les élevant au carré :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{X}_i - X_i\right)^2$$

Racine de l'erreur quadratique moyenne

La racine de l'erreur quadratique moyenne (Root Mean Square Error : RMSE) est utilisée aussi pour comparer des modèles différents. Avec ce critère une valeur atypique de la série a une grande influence sur l'erreur, ce qui pose des problèmes en cas de valeurs aberrantes (outlier).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{X}_i - X_i\right)^2}$$

Variance relative moyenne

La variance relative moyenne, connue sous l'abréviation ARV (Average Relative variance) est le carré du critère utilisé par Lapedes et Farber (1987).

$$ARV = \frac{\sum_{i=1}^{n}\left(X_i - \hat{X}_i\right)^2}{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}$$

où $\overline{X}$ est la moyenne estimée sur l'ensemble de données utilisées pour la prévision. On peut constater que si à chaque instant i la prévision $\hat{X}_i$ est égale à la moyenne estimée $\overline{X}$, l'ARV sera de 1. D'autre part, si la prévision est parfaite c'est à dire $X_i = \hat{X}_i$, l'
ARV sera égale à 0.

La prévision consiste à prévoir les valeurs futures VED_{t+l}, l = (1, 2, 3, ...) de la série à partir de ses valeurs observées jusqu'au temps t. $VEDF_t$ (l) désigne la prédiction de la série au temps t + l, qui est en général différente de la valeur réelle VED_{t+l} notée REAL Eviews a été utilisé dans son module forecasting pour obtenir ces valeurs qui rappelons le sont les valeurs de VED de Juillet 2004 à Septembre 2005, c'est-à-dire 15 valeurs. La série VEDF représente les valeurs de VED prédites. Les résultats obtenus sont alors donnés dans le tableau A.14 de l'annexe A et dans le tableau Tab.3.4 ci dessous. La moyenne des 15 observations réelles est $\overline{REAL}$ = 14616669,9.

Ci-dessous (Fig 3.8) on représente la série des 15 prévisions. Nous pouvons constater une fiabilité correcte du modèle, puisque les prévisions se trouvent à l'intérieur de l'intervalle de confiance.

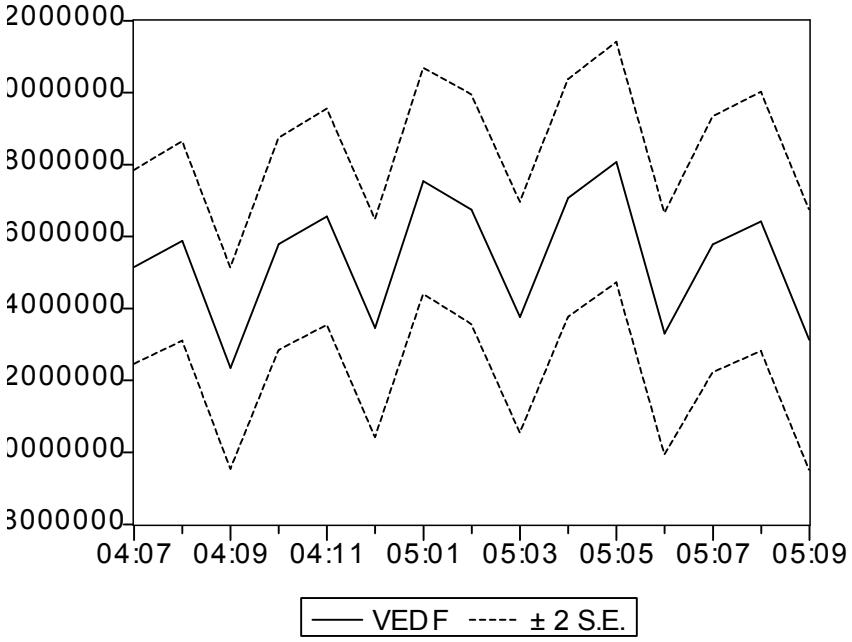


Fig 3.8 la série des 15 prévisions

SSE ou $\sum_{i=1}^{15}(REAL_i - VEDF_i)^2$	**3,7654E+13**
Racine de SSE (RSSE)	**6136285,5214**
MSE	**2510266666666,66**
RMSE	**1584382,11**
ARV= $\sum_{i=1}^{15}(REAL_i - VEDF_i)^2 \ / \ \sum_{i=1}^{15}(REAL_i - \overline{REAL})^2$	**0,502435188**

Tab.3.4 la performance de la prévision à l'aide du modèle retenu

Conclusion

Avec la modélisation de Box-Jenkins, on a stationnarisé la série temporelle et on a trouvé un effet GARCH prononcé qui correspond à un processus non linéaires en variance, on n'est pas sur que cette modélisation est capable de décrire convenablement le comportement non linéaire de notre série et vu cet inconvénient, nous présentons dans ce qui suit l'approche connexionniste qui peut résoudre le problème de non linéarité en fournissant à l'utilisateur une grande liberté de conception de modèles et une flexibilité de choix de paramètres.

3.3.3. Traitement par la méthode neuronale

3.3.3.1. Notation

On note dans ce qui suit par RN (j_1 , :::::; j_k; h) le réseau de neurones avec les délais j_1; :::::; j_k , si X_{t-j1} , X_{t-j2},… X_{t-jk} sont les entrées et X_t, est la sortie et h désigne le nombre de neurones de la couche cachée. Ainsi la figure 3.9 représente un modèle RN (1, 2, 3, 4; 3). Aussi, RN (l-h-s) désignera dans la suite un réseau de neurones à l entrées, une couche cachée à h unités et une couche de sortie à s unités (dans notre cas, s=1), le réseau de 3.8 peut alors s'écrire RN (4-3-1).

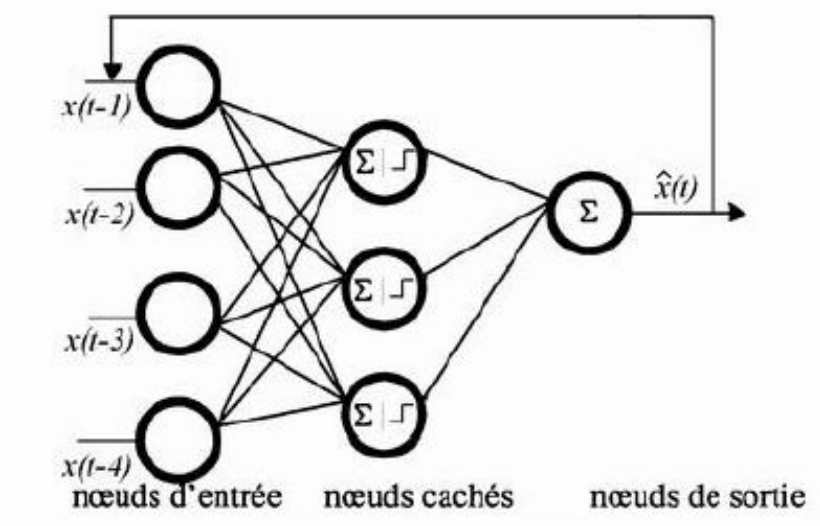


Fig 3.9 Réseau RN (4-3-1)

3.3.3.2. Mise en œuvre

La programmation des réseaux de neurones est assez fastidieuse vu le nombre de termes à manipuler et la complexité des algorithmes d'optimisation utilisés ; l'utilisation des logiciels est dés lors plus que nécessaire.

Néanmoins, la difficulté rencontrée a été de trouver un logiciel permettant la prévision des séries temporelles malgré qu'il en existe plusieurs malheureusement inaccessibles. Après un temps assez important, nous avons obtenu un logiciel très puissant qui nous a permis de faire la prévision, c'est le logiciel Alyuda NeuroIntelligence. Ce logiciel très puissant permet d'effectuer une à une dans l'ordre les opérations suivantes :

Analyse, prétraitement des données, construction de réseau, apprentissage, test et enfin requête

Toutes ces étapes sont largement expliquées dans l'annexe B. Il est à noter que la phase de requête ou r réponse du réseau est équivalente à celle de la prévision.

Comme cela a été noté ci-dessus, nous pouvons nous inspirer de la modélisation classique des séries temporelles pour choisir les paramètres du réseau de neurones, en se basant par exemple sur les travaux empiriques de Zaiyoung Tang et PaulA. Fishwick qui considèrent que le modèle airline défini par Box-Jenkins est bien similaire à un modèle neuronal de prévision à un pas de temps. Héctor Allende et Claudio Moraga et Rodrigo Salas ont aussi aboutit après étude empirique des données internationales"air line data" que le modèle RN (1;12;13; 1) est le meilleur approprié suite à une analogie avec le modèle airline classique$(1-B)\ (1-B^{12})\ X_t = (1-\theta_1 B)\ (1-\theta_{12} B^{12})\ \varepsilon_t$, ce qui équivaut à $X_t - X_{t-1} - X_{t-12} + X_{t-13} = \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_{12} \varepsilon_{t-12} + \theta_1 \theta_{12} \varepsilon_{t-13}$, duquel on peut comprendre le choix des retards au niveau de l'architecture neuronale.

Adaptons cette méthode, c'est-à-dire on prend comme réseau de neurones un réseau avec des retards de VED_t similaires à ceux du SARIMA trouvé, c'est à dire dans notre cas le modèle SARIMA (3; 0; 3) (0;1;1) sans restriction sur les paramètres, il s'écrit :

$(1-\varphi_1 B- \varphi_2 B^2 - \varphi_3 B^3)\ (1-B)^0\ (1-B^{12})\ VED_t = 1-sma(1)B^{12})[1-ma(1)B-ma(2)B^2-ma(3)B^3)]\varepsilon_t$ (I)

$(1-\varphi_1 B-\varphi_2 B^2 - \varphi_3 B^3 - B^{12}+ \varphi_1 B^{13} +\varphi_2 B^{14} +\varphi_3 B^{15})\ VED_t = \theta\ (\varepsilon_t)$

où $\theta\ (\varepsilon_t)$ est le deuxième membre de cette équation, indépendant de VED_t. (I) s'écrit alors:

$VED_t - \varphi_1 VED_{t-1} - \varphi_2 VED_{t-2} - \varphi_3 VED_{t-3} - VED_{t-12} + \varphi_1 VED_{t-13} + \varphi_2 VED_{t-14} + \varphi_3 VED_{t-15} = \theta\ (\varepsilon_t)$

On voit que les délais liés à VED_t ou les retards concernés par le modèle sont les délais :

t-1, t-2,t-3, t-12,t-13 ,t-14 et t-15. On est tenté par une modélisation neuronale de la forme RN (1, 2,3,12,13,14,15 ;h), c'est-à-dire les entrées sont des vecteurs à sept (07) composantes et qui sont VED_{t-1} ,VED_{t-2} ,VED_{t-3} ,VED_{t-12} , VED_{t-13} ,VED_{t-14} et enfin VED_{t-15} .et le nombre d'unités de la couche cachée est h. Par rapport au logiciel utilisé, on doit former les suites VED_{t-k} k=1,2,3,12,13,14,15 à partir le la série VED_t ; la contrainte t > 15 nous impose de rétrécir la plage des valeurs de VED_{t-k} par rapport à celles de VED_t , ainsi on aura 246-15 =231 observations pour chaque input X_{tk} k=1,2,3,12,13,14,15 où on a posé $X_{tk}= VED_{t-k}$ .Ainsi, on se retrouve avec des séries de longueur 231 telles que :

La série $X_{t1} = (VED_{t-1})$ a pour valeurs VED_{15} , VED_{16},…… VED_{245}
La série $X_{t2} = (VED_{t-2})$ a pour valeurs VED_{14} , VED_{15},…… VED_{244}
La série $X_{t3} = (VED_{t-3})$ a pour valeurs VED_{13} , VED_{14},…… VED_{243}
La série $X_{t12} = (VED_{t-12})$ a pour valeurs VED_4 , VED_5,…… VED_{234}
La série $X_{t13} = (VED_{t-13})$ a pour valeurs VED_3 , VED_4,…… VED_{243}
La série $X_{t14} = (VED_{t-14})$ a pour valeurs VED_2 , VED_3,…… VED_{232}
La série $X_{t15} = (VED_{t-15})$ a pour valeurs VED_1 , VED_2,…… VED_{231}

On forme ces séries sous forme de colonnes sur Excel et on reporte les huit (08) colonnes (sept colonnes d'input et la colonne target ou cible) vers Alyuda qui va traiter ces données par étapes comme c'est décrit à l'annexe B..

La phase d'analyse

8 colonnes et 231 lignes ont été analysées, 8 colonnes et 221 lignes ont été acceptées pour l'apprentissage du réseau de neurones. 8 colonnes numériques X_{t15}, X_{t14}, X_{t13}, $X_{t12,}$ X_{t3}, X_{t2}, X_{t1}, X_t est la colonne de sortie. 10 lignes désactivées. La partition des données a été effectuée selon la méthode de partition aléatoire et on a eu :

151 enregistrements pour l'ensemble d'apprentissage (68,33%).
35 enregistrements pour l'ensemble de Validation set (15,84%)
35 enregistrements pour l'ensemble Test (15,84%).
On a relevé une anomalie dans les données:13 valeurs isolées (outliers).

La phase de prétraitement des données

C'est la normalisation des données, donc après le prétraitement des données, l'échelle des sept (07) colonnes d'entrées va se trouver dans l'intervalle [-1, 1] car la fonction d'activation de la couche cachée est la tangente hyperbolique. Cette normalisation est un prétraitement obligatoire pour les réseaux utilisant une fonction de transfert de type sigmoïde au niveau de la couche de sortie, et comme nous avons une fonction linéaire au niveau de la couche de sortie (c'est l'identité), nous ne serons pas obligés de changer l'échelle des données de la colonne de sortie.

La phase de Construction du réseau

Automatiquement le logiciel a sélectionné le premier réseau de neurones RN (7-3-1), 7 entrées (les 07 X_{tk} k=1, 2, 3, 12, 13, 14,15), 1 couche cachée formée de 3 neurones et 1 sortie car par défaut, le logiciel choisit comme nombre de neurones de la couche cachée la moitié du nombre d'entrées. On commence avec cette architecture RN (7-3-1) et avec la méthode de recherche heuristique en utilisant le critère de sélection (appelée critère de Fitness) qui est le BIC (baysien information criterion), on a trouvé plusieurs architectures, il a fallu exécuter ou plutôt entraîner plusieurs modèles plusieurs fois dans le but de minimiser la fitness et à chaque fois on obtenait un ensemble de modèles avec différents critères et enfin, la dernière phase qui est la recherche exhaustive nous a permis d'arrêter 08 architectures qui ont été vérifiées et l'architecture qui a affiché la meilleure valeur du BIC est l'architecture [7-28-1]. Le tableau Tab.A.15 de l'annexe A représente une comparaison des architectures trouvées. Remarquer que ce modèle affiche les meilleures valeurs de l'erreur d'apprentissage, la corrélation et du R-Squared qui sont successivement 811104,5, 0, 958506 et 0,918727. Par contre, le nombre de paramètres ou poids est très grand, 253. Ci-dessous, le graphe représentant l'évolution de l'erreur.

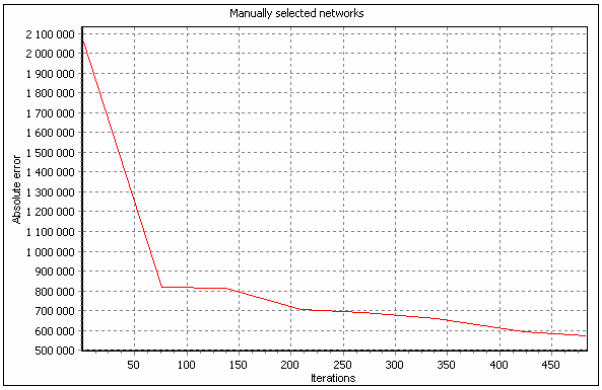


Fig 3.10 Evolution de l'erreur absolue pour le modèle [7-28-1]

La phase d'apprentissage

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement. Il existe principalement deux types d'algorithmes permettant d'appliquer la rétro propagation du gradient : la méthode du gradient total (ou batch) et la méthode du gradient stochastique (ou on-line). Dans la première méthode les gradients d'erreurs dus à la présentation de chaque exemple sont accumulés au cours du passage de la base. La somme des gradients d'erreur est ensuite employée pour modifier les poids. Dans la deuxième méthode les poids sont mis à jour après la présentation de chaque exemple.

La deuxième méthode présente l'avantage d'être plus rapide que la première, elle est par contre plus sensible aux minimums locaux. Le logiciel fait appel à quelques algorithmes cités au chapitre 4.On a retenu pour ce travail quatre algorithmes et qui sont :

- L'algorithme de Quick Propagation
- L'algorithme On line Back Propagation
- L'algorithme Batch Back Propagation
- L'algorithme de Levenberg-Marquardt.

Pour le réseau de neurones [7-28-1], on a eu les résultats suivants.

1. Résultats avec l'algorithme quick propagation

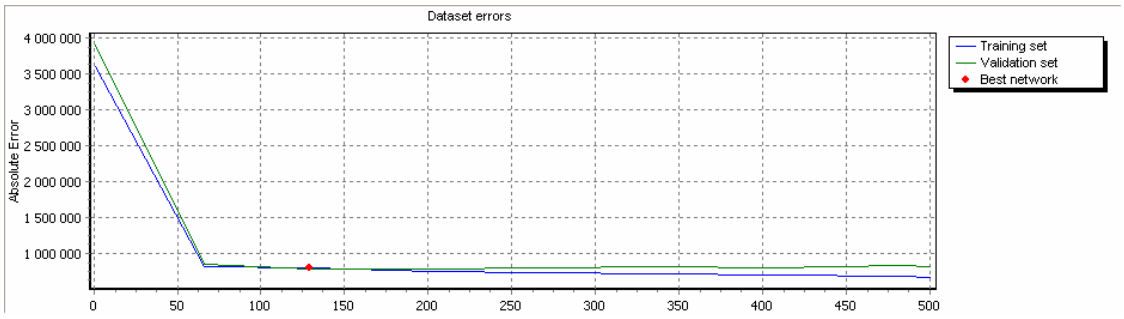


Fig 3.11 Le graphe des erreurs sur les ensembles des données (apprentissage et validation)

Le graphe des erreurs sur les ensembles de données représente l'erreur absolue moyenne de l'ensemble des données en fonction de l'ensemble des itérations, c'est à dire comment l'erreur absolue est réduite d'une itération à une autre. On retrouve la technique « Early Stopping » on

s'arrête quand l'erreur remonte "trop" sur l'ensemble de validation. L'erreur du réseau est de 0,00955. Le nombre d'itérations est de 501. Le temps passé est de 00:00:01 et la raison de l'arrêt de l'apprentissage est que toutes les itérations étaient effectuées. Ci-dessous, on a le tableau donnant l'Input Importance qui montre l'importance relative de chaque colonne entrée. Ce tableau permet de comprendre les colonnes les plus importantes qui ont eu le plus d'influence sur le réseau. On voit que les entrées relatives à t-1, t-3 et t-12 sont les plus importantes. C'est bon vu la saisonnalité de la série et le fait que les valeurs de X_t sont liées aux valeurs de rang t-1 et t-3.

Input column name	Importance, %
X_{t-15}	**8,237652**
X_{t-14}	**2,483975**
X_{t-13}	**3,416372**
X_{t-12}	**37,984207**
X_{t-3}	**18,73261**
X_{t-2}	**2,568165**
X_{t-1}	**26,577019**

Tab 3.5 Input importance avec l'algorithme quick propagation

La phase de test

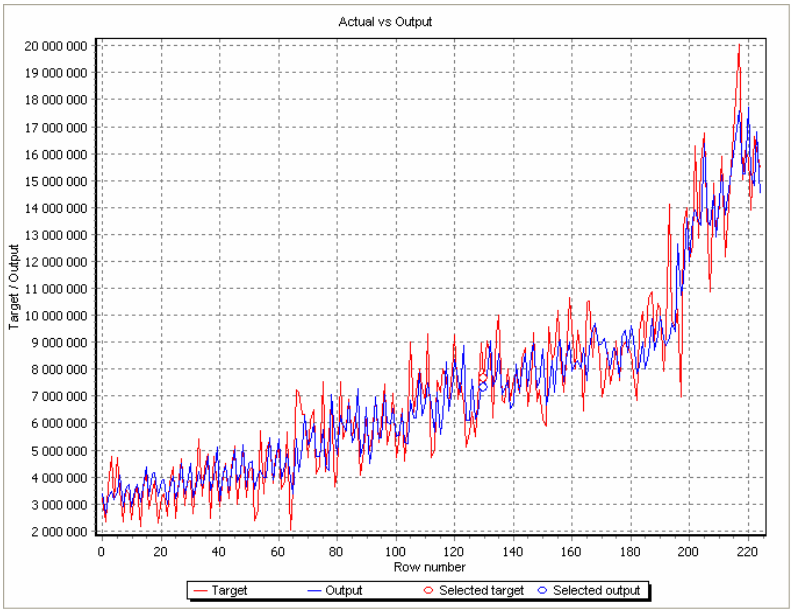


Fig 3.12 Les valeurs actuelles vs les valeurs de sortie

On peut y comparer les valeurs actuelles vs les valeurs de sortie en termes de graphes (voir Fig.3.12 ci-dessus). On obtint en premier un tableau donnant pour chaque ligne, les paramètres suivants : Target, Output, AE et ARE. Target ou cible est la vraie valeur ou valeur actuelle, output ou sortie est la valeur calculée par le réseau ou dans notre cas la valeur prévue. AE= Absolute Error = valeur absolue de (sortie - cible), enfin ARE =Absolute Relative Error = (sortie - cible) / |cible|. Cette dernière valeur de l'erreur donne une idée précise sur la qualité du réseau, elle est exprimée en pourcentages.

Le tableau ci-dessous représente les moyennes des valeurs cible, de sortie, leurs valeurs min et max ainsi que des deux mesures de l'erreur (AE et ARE) sur l'ensemble des valeurs disponibles.

Cette modélisation a donné comme valeurs de la corrélation et de la R-squared successivement 0,949245 et 0,887036.

	Target	**Output**	**AE**	**ARE**
Mean:	7089920,972851	7042971,600623	799855,985893	0,126144
Std Dev:	3459982,016559	3243313,586247	740616,125389	0,121435
Min:	2033468	2651155,183112	9297,399064	0,001539
Max:	20054399	17703504,189895	4980111,731155	0,933435

Tab 3.6 Résultats de la phase de test

Ce sont de bons résultats, vu que les erreurs sont très faibles.

La phase de requête

Dans cette phase, on fait entrer de nouvelles données (hors échantillon, utilisant de nouveaux exemples) et on demande au réseau de répondre.

Nous avons 15 données à prévoir, ce sont les observations VED_t t=247….261. D'après les entrées disponibles et le principe de prévision choisi qui est celui de « one step ahead forecasting method », pour avoir VED_{247}, on doit utiliser les observations 246, 245,244, 235, 234,233 et 232. Pour avoir VED_{248}, on doit utiliser les observations 246,245,236,235,234, 233 et 247, cette dernière sera la valeur forecasted obtenue lors de la première opération, et Ainsi, à chaque fois on utilise, quand elle n'est pas disponible, les valeurs de VED_t prédites à des étapes qui ont précédé. On obtient quinze vecteurs à sept composantes chacun et à chaque

présentation de l'exemple, on calcule la sortie du réseau, bien sur par rapport au dernier réseau choisi [7-28-1]. Ci-dessous les résultats dans le tableau 3.7:

Observation	Prévision VEDF Réseau de neurones	Réalisation REAL
247	14714184,819381	14077116
248	17243373,250138	14542408
249	15018316,584952	10835791
250	16009532,470265	12002959
251	16450362,177761	15084737
252	15497529,833926	12810994
253	16679661,01997	19182011
254	16087237,150207	18411711
255	14542687,191272	13629048
256	16746134,444095	15619071
257	17553452,155579	15545358
258	13742246,750478	12198798
259	15627610,907751	15792681
260	17878837,79086	16288953
261	15072113,136649	13228413

Tab 3.7 Prévision hors échantillon

2. Résultats avec l'algorithme On line Back Propagation

L'erreur du réseau est de 0, 16105. Le nombre d'itérations est de 501. Le temps passé est de 00:00:05et la raison de l'arrêt de l'apprentissage est que toutes les itérations étaient effectuées.

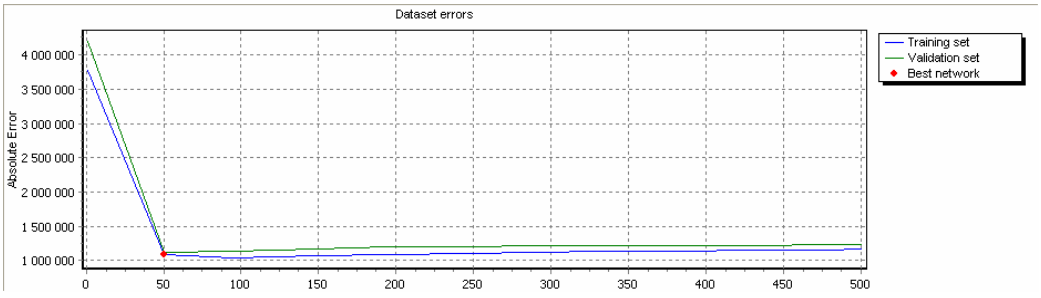


Fig 3.13 Le graphe des erreurs sur les ensembles des données (apprentissage et validation)

Input column name	Importance, %
X_{t-15}	18,541637
X_{t-14}	0,766291
X_{t-13}	0,766346
X_{t-12}	47,500097
X_{t-3}	5,95112
X_{t-2}	9,389227
X_{t-1}	17,085282

Tab 3.8 Input importance avec l'algorithme On line Back Propagation

Remarquer l'importance des entrées X_{t-1}, X_{t-12} et de X_{t-15}

La phase de test

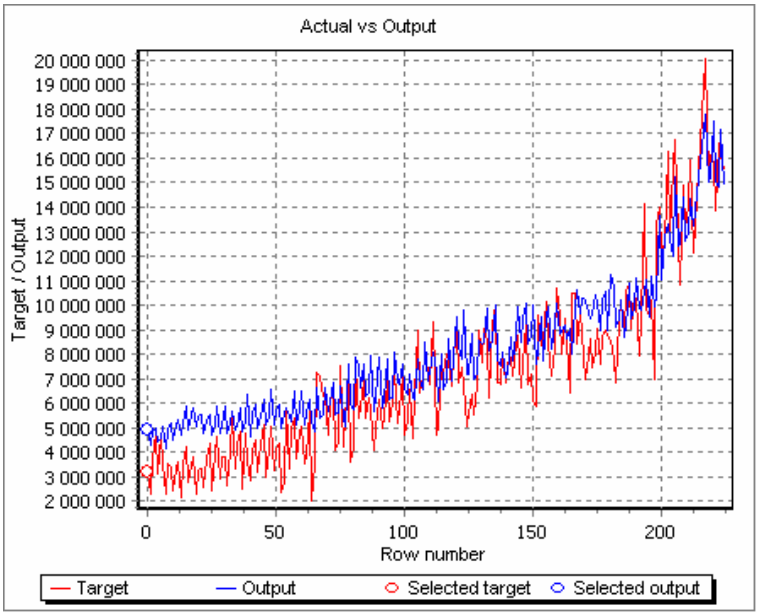


Fig 3.14 Les valeurs actuelles vs les valeurs de sortie

	Target	Output	AE	ARE
Mean:	**7089920,972851**	**7932333,264835**	**1272695,119458**	**0,256373**
Std Dev:	**3459982,016559**	**2787780,270423**	**823582,395628**	**0,257268**
Min:	**2033468**	**4165232,54741**	**11345,467153**	**0,001602**
Max:	**20054399**	**17765517,24816**	**3886200,460057**	**1,598865**

Tab 3.9 Résultats de la phase de test

La corrélation : 0, 941086 et la R-squared : 0,704307.

La phase de requête

Observation	Réseau de neurones	Valeur réelle
247	17044184,523057	14077116
248	19124703,706157	14542408
249	16108675,535847	10835791
250	16808339,975821	12002959
251	17193721,689832	15084737
252	16427967,33742	12810994
253	17723032,00634	19182011
254	17232052,896709	18411711
255	15892506,441587	13629048
256	17802034,120618	15619071
257	17997280,194914	15545358
258	14462888,14991	12198798
259	17268902,97962	15792681
260	18735977,478613	16288953
261	15619265,839871	13228413

Tab 3.10 Prévision hors échantillon

3. Résultats avec l'algorithme Batch Back Propagation

Fig 3.15 Graphe des erreurs sur les ensembles des données (apprentissage et validation)

Input column name	Importance, %
X_{t-15}	19,444953
X_{t-14}	0,116997
X_{t-13}	0,295284
X_{t-12}	36,576898
X_{t-3}	25,432676
X_{t-2}	0,690953
X_{t-1}	17,442239

Tab 3.11 Input importance avec l'algorithme Batch Back Propagation

L'erreur du réseau est de 0,017058. Le nombre d'itérations est de 501. Le temps passé est de 00:00:01 et la raison de l'arrêt de l'apprentissage est que toutes les itérations étaient effectuées.

D'après le tableau Tab 3.11 ci-dessus, et comme précédemment, on remarque l'importance des entrées X_{t-1}, X_{t-3} X_{t-12} et X_{t-15}.

La phase de test

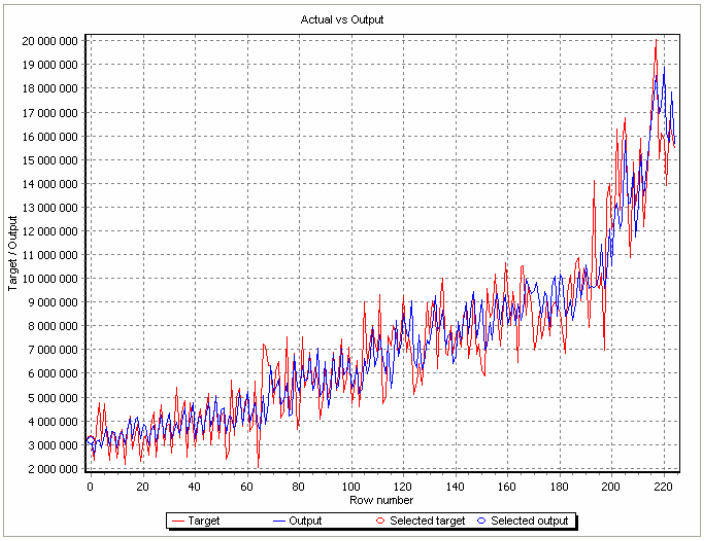


Fig 3.16 Les valeurs actuelles vs les valeurs de sortie

z	Target	Output	AE	ARE
Mean:	**7089920,972851**	**7094085,883169**	**859454,383812**	**0,131335**
Std Dev:	**3459982,016559**	**3347681,198716**	**769583,227519**	**0,117953**
Min:	**2033468**	**2616969,002332**	**2144,146109**	**0,000607**
Max:	**20054399**	**18870417,023559**	**4511387,608836**	**0,843997**

Tab 3.12 Résultats de la phase de test

La corrélation : 0,943093 et la R-squared : 0, 881242

La phase de requête

Observation	Sortie du Réseau de neurones	Valeur actuelle
247	17988268,902064	14077116
248	19764405,67688	14542408
249	16627985,008748	10835791
250	18966450,555925	12002959
251	19565050,738267	15084737
252	18315466,412715	12810994
253	20150551,800827	19182011
254	19823990,995782	18411711
255	18037407,428451	13629048
256	20081271,457143	15619071
257	20737764,983258	15545358
258	16968959,022827	12198798
259	19870317,469543	15792681
260	21519672,498421	16288953
261	18306017,365208	13228413

Tab 3.13 Prévision hors échantillon

4. Résultats avec l'algorithme de Levenberg-Marquardt

L'erreur du réseau est de 9,32E-12. Le nombre d'itérations est de 501. Le temps passé est de 00:15:55et la raison de l'arrêt de l'apprentissage est que toutes les itérations étaient effectuées.

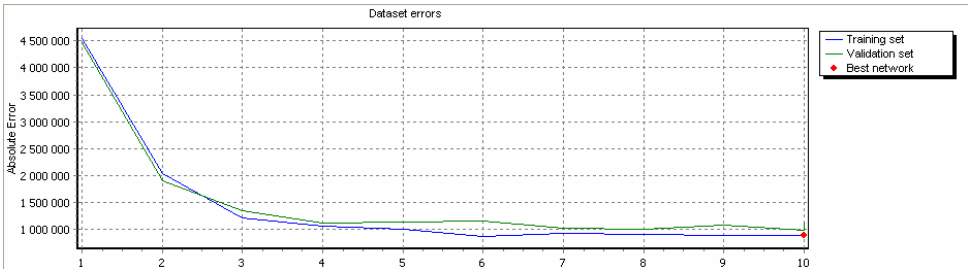


Fig 3.17 Graphe des erreurs sur les ensembles des données (apprentissage et validation)

Input column name	Importance, %
X_{t-15}	14,251499
X_{t-14}	1,718434
X_{t-13}	4,600935
X_{t-12}	4,260434
X_{t-3}	8,708014
X_{t-2}	19,844837
X_{t-1}	46,615846

Tab 3.14 Input importance avec l'algorithme de Levenberg-Marquardt

La phase de Test

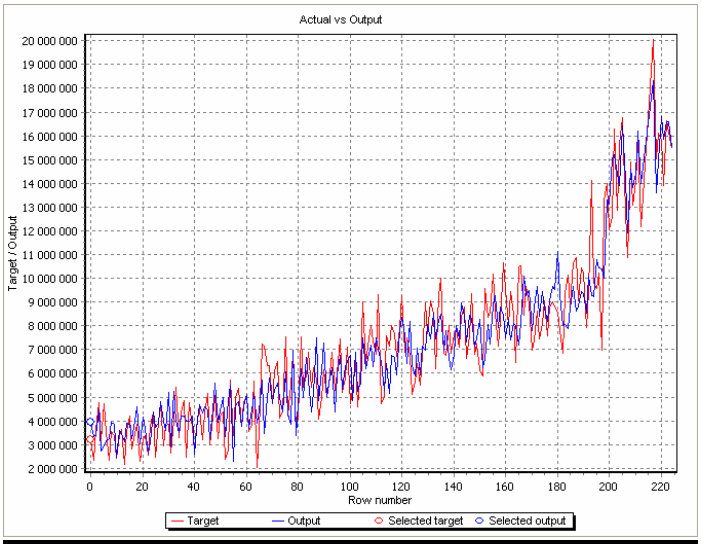


Fig 3.18 Les valeurs actuelles vs les valeurs de sortie

	Target	Output	AE	ARE
Mean:	**7089920,972851**	**7047846,855217**	**890284,194628**	**0,141775**
Std Dev:	**3459982,016559**	**3332484,923679**	**746164,654483**	**0,129001**
Min:	**2033468**	**2308118,623403**	**1572,085578**	**0,000386**
Max:	**20054399**	**18328265,599548**	**4834473,288099**	**0,915584**

Tab 3.15 Résultats de la phase de test

La corrélation est de 0,942268 et la R-squared : 0,878495.

La phase de requête

N°Obs.	Prévision VEDF avec Réseau de neurones	Réalisation REAL
247	**13921284,06**	**14077116**
248	**16150215,974585**	**14542408**
249	**11380058,440656**	**10835791**
250	**14690744,286607**	**12002959**
251	**16666538,875853**	**15084737**
252	**15376245,926663**	**12810994**
253	**17504255,720011**	**19182011**
254	**16750873,483987**	**18411711**
255	**14267524,852189**	**13629048**
256	**15592692,339887**	**15619071**
257	**18611127,350528**	**15545358**
258	**11893835,29182**	**12198798**
259	**14495381,1854**	**15792681**
260	**15995614,211658**	**16288953**
261	**13130400,439938**	**13228413**

Tab 3.16 Prévision hors échantillon

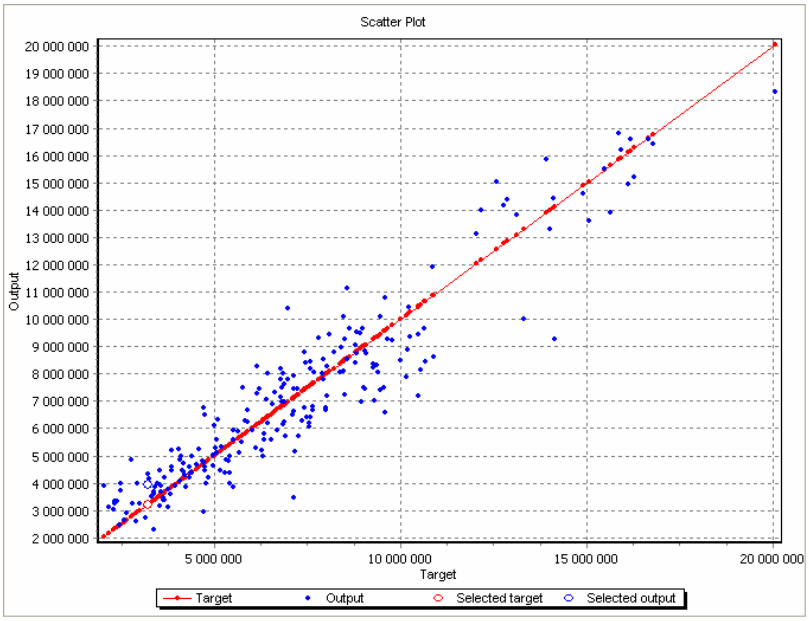


Fig 3.19 Graphe de la dispersion de l'erreur

Le graphe ci dessus appelé « scatter plot » ou graphe de la dispersion représente la dispersion des valeurs de sortie ou forecasted values (valeurs prévues) par rapport aux valeurs target ou cibles.

Commentaires des résultats

Le tableau suivant donne les valeurs de l'erreur du réseau, la corrélation et la R-squared et ce, pour les quatre méthodes d'apprentissage utilisées :

Méthode	**L'erreur du réseau**	**La corrélation**	**la R-squared**
Quick Propagation	0,00955.	0,949245	0,887036
On line Back Propagation	0, 16105	0,941086	0,704307
Batch Back Propagation	0,017058	0,943093	881242
Levenberg-Marquardt	9,32E-12	0,942268	0,878495

Tab 3.17 Erreur du réseau, corrélation et R-squared pour chaque méthode

Les quatre valeurs de la corrélation et de la R-squared sont bonnes car elles s'approchent de 1.Le résultat d'apprentissage retenu est celui obtenu avec la méthode de Levenberg-Marquardt vu que la valeur de l'erreur du réseau avec cette méthode est la plus faible. On retient comme résultats de la requête (prévision) ceux obtenus avec cette méthode (voir Tab A.16 de l'annexe A).

Les résultats concernant les erreurs de prévision sont données ci-dessous dans le tableau Tab.3.18

SSE ou $\sum_{i=1}^{15}(REAL_i - VEDF_i)^2$	**3,6465 E+13**
Racine de SSE (RSSE)	**6038625,6714**
MSE	**2431 E+9**
RMSE	**1559166,4439**
ARV= $\sum_{i=1}^{15}(REAL_i - VEDF_i)^2 \ / \ \sum_{i=1}^{15}(REAL_i - \overline{REAL})^2$	**0,486569793**

Tab.3.18 La performance de la prévision à l'aide du modèle neuronal [7-28-1]

3.3.3.3. Comparaison du pouvoir prédictif de la méthode de Box-Jenkins à celui du réseau de neurones

Dans cette partie, on va comparer la prévision obtenue avec le modèle neuronal (après apprentissage du réseau [7-28-1] avec la méthode Levenberg-Marquardt) à celle obtenue avec la modélisation SARIMA. Le tableau 3.19 regroupe tous ces résultats. Rappelons que :

Variance Relative Moyenne = ARV= $\sum_{i=1}^{15}(REAL_i - VEDF_i)^2 \ / \ \sum_{i=1}^{15}(REAL_i - \overline{REAL})^2$

$\overline{REAL}$ = 14616669,9. et que $\sum_{i=1}^{15}(REAL_i - \overline{REAL})^2$ = 7,4943 E+13, d'où les résultats du tableau Tab. 3.19. Il en résulte que le modèle de réseau de neurones [7-28-1] est plus performant que le modèle SARIMA (3; 0; 3) (0, 1,1) avec erreurs EGARCH (0,1).

Finalement on peut dire que les modélisations classiques offrent une source d'inspiration pour le choix d'architecture des réseaux de neurones (surtout le choix du nombre de neurones de la couche d'entrée et les retards qu'ils représentent), ce choix qui a resté une lacune, et qui a été

considéré parmi les désavantages des réseaux de neurones dans l'étape de l'identification de l'architecture optimale.

Critère de comparaison	SARIMA (3; 0; 3) (0, 1 ,1) avec erreurs EGARCH (0,1)	Réseau de neurones [7-28-1]
Somme des carrés des résidus (SSE)	3,7654E+13	3,6465 E+13
Racine de SSE (RSSE)	6136285,5214	6038625,6714
MSE L'erreur quadratique moyenne	2510266666666,66	2431 E+9
RMSE Racine de l'erreur quadratique moyenne	1584382,11	1559166,4439
Variance Relative Moyenne ARV=	0,502435188	0,486569793

Tab. 3.19
Comparaison du pouvoir prédictif des réseaux de neurones et la méthodologie Box-Jenkins.

CONCLUSION

Après l'analyse des résultats obtenus, nous pouvons affirmer que les modèles neuronaux représentent l'avantage de pouvoir prévoir une série temporelle et que le réseau apprend et modélise lui-même sans avoir besoin qu'on lui indique des hypothèses ou des restrictions. On obtient en outre une meilleure performance par rapport aux modèles classiques.
Cet avantage ne nous permet pas de cacher les inconvénients des réseaux de neurones qui se présentent dans le problème de la "boîte noire" : un réseau de ne fournit pas d'explication à son problème. Parmi les autres limites des réseaux de neurones, c'est l'absence d'une méthode systématique d'identification de l'architecture initiale du réseau, cet inconvénient comme nous l'avons déjà mentionné, peut être remédié par une inspiration auprès des méthodes classiques, au niveau du choix des variables dépendantes qui alimentent le vecteur d'entrée du réseau. C'est la méthode adaptée dans ce travail.

Une des perspectives serait d'étudier le cas d'une modélisation AR(X) (autorégressif à variable exogène) où on considérerait les variables indépendantes qui influent sur la consommation d'électricité comme la température, par exemple et de comparer cette approche avec celle de neurones où on prendrait comme entrées ces variables exogènes en plus des retards de la variable à prévoir. Le problème de récolte des données se pose alors où on ne dispose pas actuellement d'une banque de données reflétant l'historique complète (de 2004 à 2005) de toutes les variables exogènes.

Ci-dessous (fig.3.20) une comparaison entre les deux approches étudiées dans ce travail.

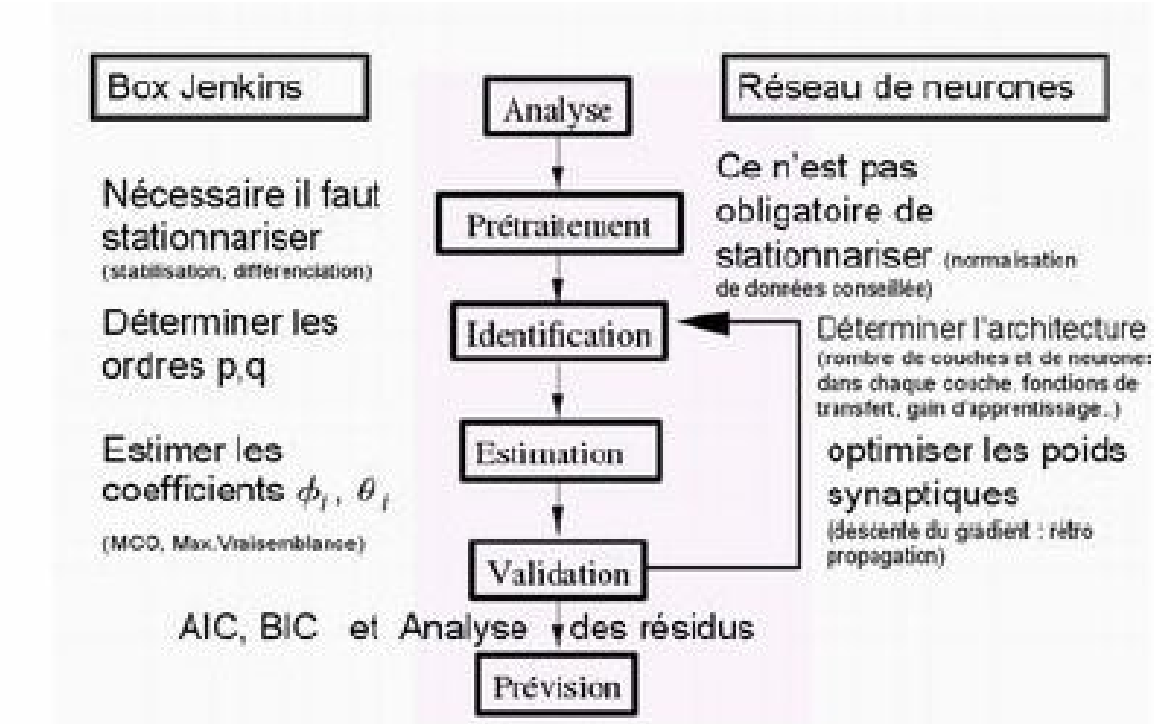


Fig 3.20 Comparaison entre l'approche de Box et Jenkins et celle de neurones

ANNEXES

ANNEXE A. TABLEAUX, FIGURES ET RESULTATS DES PHASES D'ESTIMATION

Dans ce qui suit, nous allons présenter annexes certains rappels ainsi que quelques résultats de calcul obtenus dans chacune des deux phases d'estimation, celle de SARIMA et celle des réseaux de neurones respectivement.

Modèle	FAC	FAP
AR (p)	fonction exponentielle et /ou une sinusoïde amortie incluse dans l'enveloppe E.	Nulle pour k>p
MA (q)	Nulle pour k> q	fonction exponentielle et /ou une sinusoïde amortie incluse dans l'enveloppe E.
ARMA (p,q)	mélange de fonctions exponentielles et sinusoïdales amorties incluse dans l'enveloppe E. pour k> q-p	Comportement général inclus dans l'enveloppe E pour k> p-q

Tab.A.1 Caractéristiques des processus ARMA

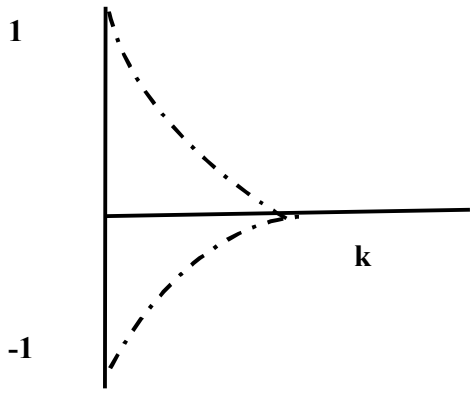


Fig A.2 tracé de l'enveloppe E

A.1 La résolution du problème du XOR

On utilise trois cellules d'entrée au lieu de deux et on ajoute une troisième couche (interne) appelée couche cachée. L'ensemble des poids indiqués dans le tableau A.4 ci-dessous résout le problème du XOR (cette solution n'est pas unique). $\theta_{11} = 1$, $\theta_{21} = 1$, $\theta_{12} = 1$ $\theta_{22} = 1$, $\theta_{10} = -0.5$, $\theta_{20} = -1.5$, $\alpha_1 = 1$, $\alpha_2 = -2$, $\alpha_0 = -0.5$, $\theta ij_{\ i=1,2 \text{ et } j=1,2}$ représente le poids qui relie la cellule de la couche cachée i à celle d'entrée j et α_k k=1,2 est le poids reliant la cellule cachée k à la sortie. θ_{k0} représente le biais (ou seuil), c'est le poids d'une entrée supposée égale à 1 reliée à la cellule cachée k, k=1,2. Nous avons un réseau à deux entrées avec une entrée supplémentaire égale à l'unité avec le poids θ_{k0} (biais ou seuil) et une cachée formée de deux cellules, en plus d'une cellule cachée seuil dont le poids de connexion à la sortie est α_0 Si nous associons la fonction seuil à ce réseau alors $\psi(x) = 1$ si $x > 0$ et $\psi(x) = 0$ si $x \leq 0$, alors

$\psi(-0.5)=0$ (à 0 0 → 0)
$\psi(1)=1$ (à 1 0 → 1)
$\psi(1)=1$ (à 0 1 → 1)
$\psi(0)=0$ (à 1 1 → 0)

Entrée	Activation de la cellule interne (ou cachée)		Activation de la cellule de sortie
(0, 0 ,1)	1ère cellule cachée	2ère cellule cachée	(0×1)+ (0×-2) + (1×-0.5)= - 0.5
	(0×1)+ (0×1)+(1×-0.5)	(0×1)+ (0×1)+(1×-1.5)	
0, 1 ,1)	1ère cellule cachée	2ère cellule cachée	(0.5×1)+ (-0.5×-2) + (1×-0.5)= 1
	(0×1)+ (1×1)+(1×-0.5)	(0×1)+ (1×1)+(1×-1.5)	
(1, 0,1)	1ère cellule cachée	2ère cellule cachée	(1×0.5)+ (-0.5×-2) + (1×-0.5)=1
	(1×1)+ (0×1)+(1×-0.5)	(1×1)+ (0×1)+(1×-1.5)	
(1, 1 ,1)	1ère cellule cachée	2ère cellule cachée	(1×1.5)+ (0.5×-2) + (1×-0.5)=0
	1×1)+ (1×1)+(1×-0.5)	(1×1)+ (1×1)+(1×-1.5)	

Tab A.3 L'ensemble des poids qui résout le problème du XOR

Année	Moyenne de l'année
1	2205149
2	3320134
3	3202237
4	3530282
5	3984904
6	4031480
7	4858771
8	5341709
9	5736221
10	5934657,
11	7210450
12	6837839
13	7974898
14	7613285
15	8624964
16	8595835
17	9058716
18	10676176
19	13991651
20	16704070
21	14368615

Année	Variance de l'année
1	231664E6
2	671034E6
3	416450E6
4	538590E6
5	774007E6
6	114888E7
7	251039E7
8	169084E7
9	836436E6
10	886690E6
11	187246E7
12	132182E7
13	141223E7
14	146136E7
15	144248E7
16	139288E7
17	127854E7
18	515921E7
19	353262E7
20	784766E7
21	808000E7

Tab A.4 Evolution de la moyenne et de la variance de la série VED

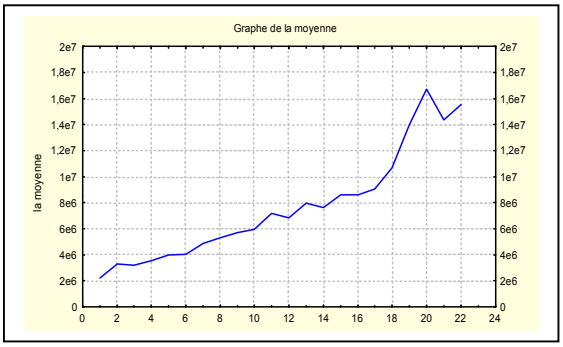


Fig A.5 Evolution de la moyenne de VED

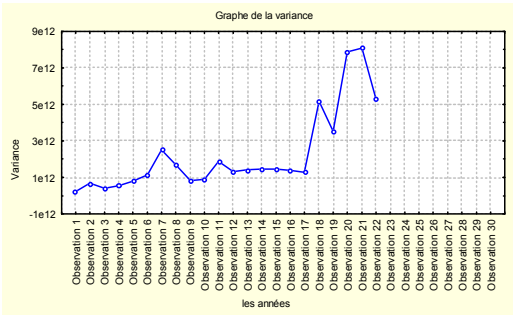
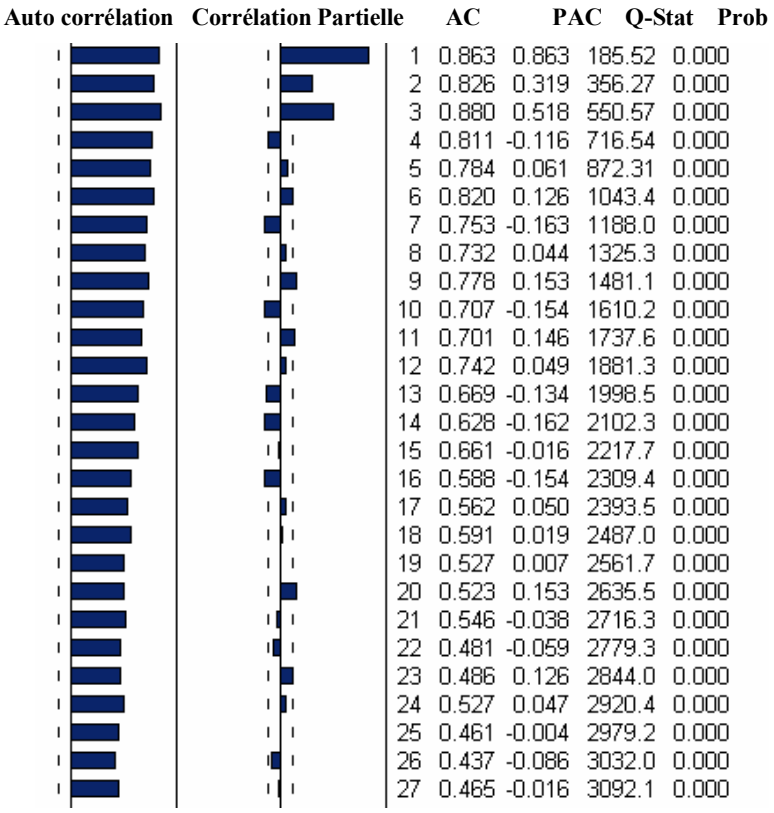
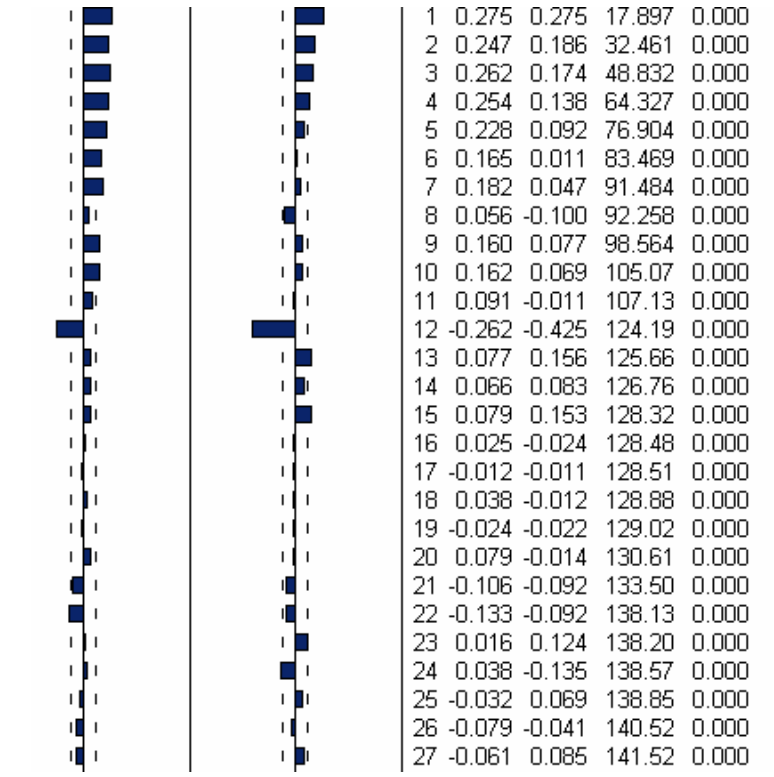


Fig A.6 Evolution de la variance de VED



	AC	PAC	Q-Stat	Prob
1	0.863	0.863	185.52	0.000
2	0.826	0.319	356.27	0.000
3	0.880	0.518	550.57	0.000
4	0.811	-0.116	716.54	0.000
5	0.784	0.061	872.31	0.000
6	0.820	0.126	1043.4	0.000
7	0.753	-0.163	1188.0	0.000
8	0.732	0.044	1325.3	0.000
9	0.778	0.153	1481.1	0.000
10	0.707	-0.154	1610.2	0.000
11	0.701	0.146	1737.6	0.000
12	0.742	0.049	1881.3	0.000
13	0.669	-0.134	1998.5	0.000
14	0.628	-0.162	2102.3	0.000
15	0.661	-0.016	2217.7	0.000
16	0.588	-0.154	2309.4	0.000
17	0.562	0.050	2393.5	0.000
18	0.591	0.019	2487.0	0.000
19	0.527	0.007	2561.7	0.000
20	0.523	0.153	2635.5	0.000
21	0.546	-0.038	2716.3	0.000
22	0.481	-0.059	2779.3	0.000
23	0.486	0.126	2844.0	0.000
24	0.527	0.047	2920.4	0.000
25	0.461	-0.004	2979.2	0.000
26	0.437	-0.086	3032.0	0.000
27	0.465	-0.016	3092.1	0.000

Fig. A.7 Corrélogramme et corrélogramme partiel de la série VED



	AC	PAC	Q-Stat	Prob
1	0.275	0.275	17.897	0.000
2	0.247	0.186	32.461	0.000
3	0.262	0.174	48.832	0.000
4	0.254	0.138	64.327	0.000
5	0.228	0.092	76.904	0.000
6	0.165	0.011	83.469	0.000
7	0.182	0.047	91.484	0.000
8	0.056	-0.100	92.258	0.000
9	0.160	0.077	98.564	0.000
10	0.162	0.069	105.07	0.000
11	0.091	-0.011	107.13	0.000
12	-0.262	-0.425	124.19	0.000
13	0.077	0.156	125.66	0.000
14	0.066	0.083	126.76	0.000
15	0.079	0.153	128.32	0.000
16	0.025	-0.024	128.48	0.000
17	-0.012	-0.011	128.51	0.000
18	0.038	-0.012	128.88	0.000
19	-0.024	-0.022	129.02	0.000
20	0.079	-0.014	130.61	0.000
21	-0.106	-0.092	133.50	0.000
22	-0.133	-0.092	138.13	0.000
23	0.016	0.124	138.20	0.000
24	0.038	-0.135	138.57	0.000
25	-0.032	0.069	138.85	0.000
26	-0.079	-0.041	140.52	0.000
27	-0.061	0.085	141.52	0.000

Fig A.8 Corrélogramme et Corrélogramme partiel de la série SVED

A.2 Test de stationnarité sur la série SVED
Pour level et lagged 0, on a eu les résultats :

On applique le test de la racine unitaire pour voir si la série n'est pas affectée d'une tendance et dans le cas positif déterminer la nature de cette tendance. On teste le modèle $(1-\phi_1 B)(X_t - c - bt) = \varepsilon_t$ (modèle autorégressif d'ordre1 avec tendance, on obtient :

ADF Test Statistic	**-11.36811**	**1% Critical Value***		**-4.0005**
		5% Critical Value		**-3.4303**
		10% Critical Value		**-3.1384**
Variable	Coefficient	Std. Error	t-Statistic	Prob.
SVED(-1)	-0.742630	0.065326	-11.36811	0.0000
C	59791.29	240262.4	0.248858	0.8037
@TREND(1984:01)	**3441.540**	**1688.375**	**2.038374**	**0.0427**

D'après les valeurs critiques de la tendance, calculées par Dickey Fuller ; la t-statistic de la tendance **2.038374** est à comparer aux valeurs 3.49, 2,79 et 2,38 et on voit la non significativité de la tendance, on passe à l'estimation du modèle 2 .

ADF Test Statistic	**-11.11317**	**1% Critical Value***		**-3.4599**
		5% Critical Value		**-2.8740**
		10% Critical Value		**-2.5734**
Variable	Coefficient	Std. Error	t-Statistic	Prob.
SVED(-1)	-0.714944	0.064333	-11.11317	0.0000
C	**484123.9**	**120776.5**	**4.008427**	**0.0001**

La constante est significative puisque sa t-statistic **4.008427** est à comparer aux valeurs 3.19, 2.53 et 2,16 et on voit que toutes lui sont inférieures, on passe alors au test de racine unitaire et on a voit bien qu'on rejette l'hypothèse H_0, la t-statistic **-11.11317** étant inférieure aux valeurs critiques -3,4599, -2,8740 et -2,5734.

Pour le problème du choix du nombre p de retards dans le test ADF, l'une des méthodes adaptées est de choisir p correspondant à la dernière auto corrélation partielle de ΔX_t significativement différente de zéro. Dans notre cas, on prend la série DSVED = Δ SVED et on observe son corrélogramme, on a pour k=1,2,3 et 11 les PACF sont significativement non nulles :donc on prend p=11.

ADF Test Statistic	-4.520929	1% Critical Value*		-4.0024
		5% Critical Value		-3.4312
		10% Critical Value		-3.1389
Variable	Coefficient	Std. Error	t-Statistic	Prob.
SVED(-1)	-0.549917	0.121638	-4.520929	0.0000
D(SVED(-1))	-0.297030	0.124997	-2.376303	0.0184
D(SVED(-2))	-0.158436	0.125649	-1.260948	0.2087
D(SVED(-3))	0.029340	0.124525	0.235616	0.8140
D(SVED(-4))	0.130499	0.121668	1.072583	0.2847

D(SVED(-5))	0.226042	0.118912	1.900919	0.0587
D(SVED(-6))	0.274296	0.117615	2.332145	0.0206
D(SVED(-7))	0.343337	0.113179	3.033575	0.0027
D(SVED(-8))	0.287404	0.108777	2.642146	0.0089
D(SVED(-9))	0.405528	0.099275	4.084872	0.0001
D(SVED(-10))	0.471873	0.084758	5.567285	0.0000
D(SVED(-11))	0.548128	0.066877	8.196069	0.0000
C	-6328.515	228919.0	-0.027645	0.9780
@TREND(1984:01)	**2571.194**	**1764.322**	**1.457327**	**0.1465**

Le trend n'est pas significatif on passe au modèle 2

ADF Test Statistic	**-4.372490**	**1% Critical Value***	**-3.4612**
		5% Critical Value	**-2.8746**
		10% Critical Value	**-2.5737**

Variable	Coefficient	Std. Error	t-Statistic	Prob.
SVED(-1)	-0.460536	0.105326	-4.372490	0.0000
D(SVED(-1))	-0.379490	0.111756	-3.395698	0.0008
D(SVED(-2))	-0.234257	0.114678	-2.042728	0.0423
D(SVED(-3))	-0.037154	0.116176	-0.319807	0.7494
D(SVED(-4))	0.074017	0.115639	0.640064	0.5228
D(SVED(-5))	0.175400	0.114026	1.538238	0.1255
D(SVED(-6))	0.231283	0.114157	2.025999	0.0440
D(SVED(-7))	0.305654	0.110481	2.766562	0.0062
D(SVED(-8))	0.255848	0.106886	2.393652	0.0176
D(SVED(-9))	0.381451	0.098154	3.886252	0.0001
D(SVED(-10))	0.455768	0.084260	5.409050	0.0000
D(SVED(-11))	0.540927	0.066873	8.088849	0.0000
C	**273939.4**	**124503.9**	**2.200248**	**0.0289**

Si on voit seulement la valeur critique de la constante à 10 % qui est de 2,16, la valeur de la t-statistic **2.200248** lui est supérieure et donc la constante est significative mais si on voit les deux autres valeurs à 1 % et à 5 % (respectivement 3.19 et 2.53), cette t-statistic leur est inférieure. Optons pour cette dernière conclusion c'est-à-dire que la constante n'est pas significative et donc on passe à l' étape 3 (estimation du modèle sans trend et sans constante).

ADF Test Statistic	**-3.797434**	**1% Critical Value***	**-2.5748**
		5% Critical Value	**-1.9411**
		10% Critical Value	**-1.6164**

Variable	Coefficient	Std. Error	t-Statistic	Prob.
SVED(-1)	-0.317347	0.083569	-3.797434	0.0002
D(SVED(-1))	-0.510370	0.095471	-5.345795	0.0000
D(SVED(-2))	-0.354575	0.101718	-3.485842	0.0006
D(SVED(-3))	-0.145594	0.106161	-1.371447	0.1717
D(SVED(-4))	-0.021324	0.108190	-0.197094	0.8439
D(SVED(-5))	0.088486	0.107939	0.819776	0.4133
D(SVED(-6))	0.154687	0.109710	1.409968	0.1600
D(SVED(-7))	0.238976	0.107211	2.229019	0.0269
D(SVED(-8))	0.199063	0.104668	1.901860	0.0586
D(SVED(-9))	0.338157	0.097037	3.484833	0.0006
D(SVED(-10))	0.426403	0.083954	5.079008	0.0000
D(SVED(-11))	0.525655	0.067118	7.831862	0.0000

D'après ces résultats, on voit que la statistique calculée pour ce modèle **-3.797434** est inférieure aux valeurs critiques relatives à **1%**, **5%** et **10%**, donc on rejette H_0 : « Il existe une racine unitaire ».

Modèle	R^2	RSS	BIC	AIC	DW
1.	0.270242	5.33E+14	31.36658	31.32215	2.217854
2.	0.226521	5.59E+14	31.49011	31.42880	1.752952
3.	0.197649	5.80E+14	31.50242	31.45643	1.809816
4.	0.206185	5.80E+14	31.44623	31.40193	1.820468
5.	0.400506	4.33^{E}+14	31.26443	31.18755	1.987936
6.	0.160103	6.13E+14	31.50715	31.46271	2.047891
7.	0.394800	4.42E+14	31.20282	31.14358	2.000335
8.	0.382688	4.46E+14	31.26930	31.20779	2.033843
9.	0.234528	5.59E+14	31.43318	31.37412	1.873961
10.	0.232483	5.55E+14	31.48237	31.42106	1.855767
11.	0.294165	5.16E+14	31.37539	31.30156	1.891995
12.	0.293926	5.10E+14	31.42326	31.34663	1.889975
13.	0.321237	4.96E+14	31.35959	31.27099	1.915675
14.	0.315539	4.95E+14	31.41651	31.32455	1.910954
15.	0.345383	4.73E+14	31.42060	31.29798	1.921411
16.	0.338512	4.83E+14	31.35712	31.25376	1.933567
17.	0.341153	4.76E+14	31.40271	31.29542	1.926710
18.	0.365023	4.59E+14	31.41448	31.27653	1.908456
19.	0.359838	4.68E+14	31.34767	31.22954	1.927299
20.	0.354602	4.67E+14	31.40642	31.28380	1.932764
21.	0.379488	4.49E+14	31.41577	31.26250	1.957751
22.	0.375338	4.56E+14	31.34647	31.21357	1.982653
23.	0.368414	4.57E+14	31.40912	31.27118	1.972182
24.	0.328794	4.85E+14	31.38208	31.30495	2.143021
25.	0.318037	4.98E+14	31.32686	31.26744	2.137146
26.	0.321402	4.90E+14	31.36852	31.30682	2.140550
27.	0.410000	4.26E+14	31.27764	31.18509	2.048301
28.	0.410531	4.31E+14	31.20459	31.13031	2.072608
29.	0.398449	4.35E+14	31.27252	31.19539	2.077328
30.	0.168620	6.07E+14	31.54845	31.47417	1.932771
31.	0.417292	4.26E+14	31.21653	31.12739	1.904976
32.	0.382109	4.46E+14	31.30376	31.22638	2.046917
33.	0.389956	4.46E+14	31.22005	31.16044	2.144090
34.	0.411315	4.30E+14	31.20797	31.13346	1.931006
35.	0.393593	4.38^{E}+14	31.28500	31.20762	1.989780
36.	0.422725	4.22E+14	31.21196	31.12255	2.000074

37.	0.439401	4.09E+14	31.18265	31.09323	1.965043
38.	**0.449548**	**4.02E+14**	**31.19286**	**31.08822**	**2.038109**
39.	0.229750	5.63E+14	31.57612	31.44159	1.921425
40.	0.345383	4.73E+14	31.42060	31.29798	1.921411
41.	0.244086	5.52E+14	31.53369	31.41411	2.066222

Tab. A.9 Comparaison des modèles candidats

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	735154.5	178142.4	4.126780	0.0001
AR(3)	0.961168	0.027708	34.68960	0.0000
AR(4)	-0.068617	0.028062	-2.445161	0.0153
MA(1)	0.287570	0.059336	4.846510	0.0000
MA(2)	0.167837	0.054236	3.094572	0.0022
MA(3)	-0.680565	0.058673	-11.59927	0.0000
SMA(12)	-0.788427	0.031316	-25.17677	0.0000

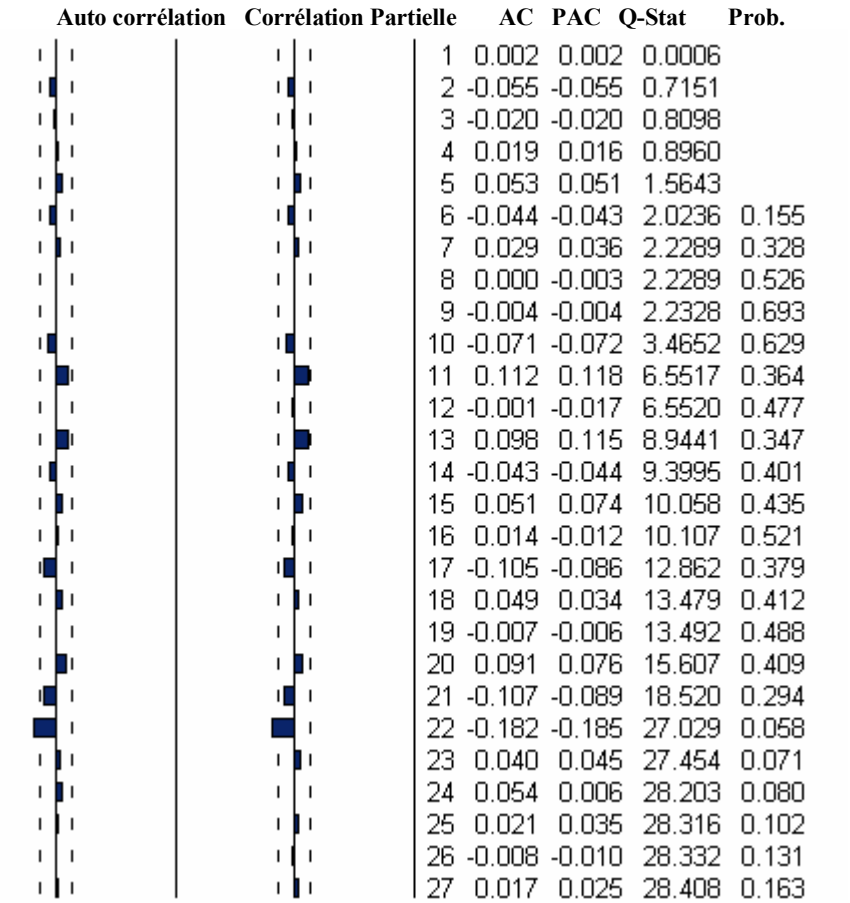
R-squared	0.449548	Mean dependent var	687219.3
Adjusted R-squared	0.434737	S.D. dependent var	1785884.
S.E. of regression	1342698.	Akaike info criterion	31.08822
Sum squared resid	4.02E+14	Schwarz criterion	31.19286
Log likelihood	-3568.146	F-statistic	30.35356
Durbin-Watson stat	2.038109	Prob(F-statistic)	0.000000

Tab A.10 Estimation du modèle SARIMA(4; 0; 3) (0;1;1)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	739860.7	232784.9	3.178301	0.0017
AR (3)	0.926588	0.039942	23.19849	0.0000
MA (1)	0.240919	0.058464	4.120817	0.0001
MA (2)	0.139510	0.057870	2.410759	0.0167
MA (3)	-0.586883	0.064980	-9.031768	0.0000
SMA (12)	-0.837855	0.030569	-27.40859	0.0000

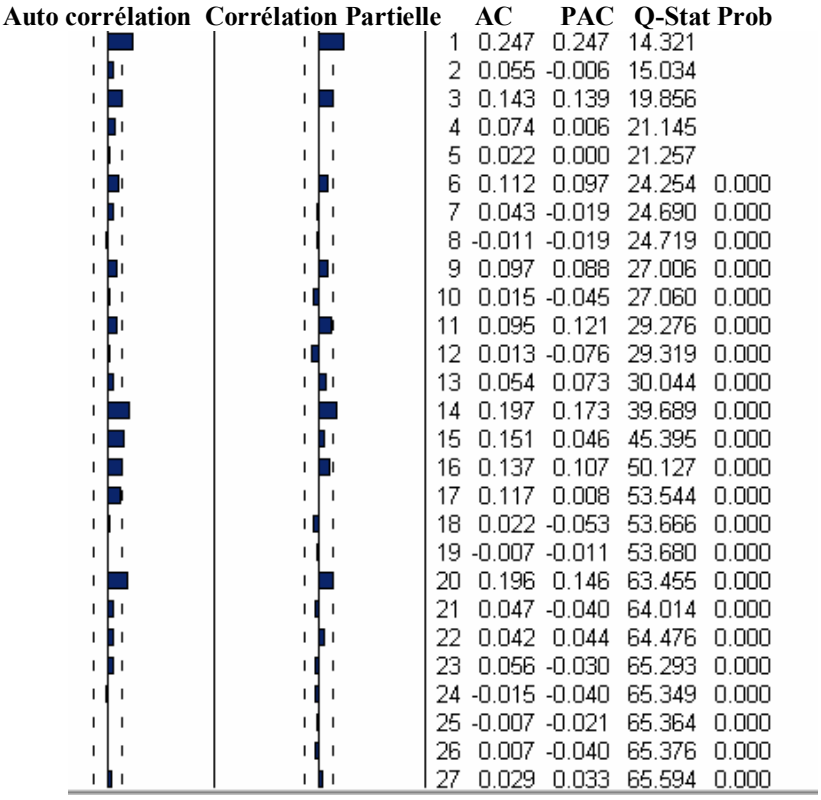
R-squared	0.439401	Mean dependent var	686936.3
Adjusted R-squared	0.426943	S.D. dependent var	1782002.
S.E. of regression	1348985.	Akaike info criterion	31.09323
Sum squared resid	4.09E+14	Schwarz criterion	31.18265
Log likelihood	-3585.269	F-statistic	35.27128
Durbin-Watson stat	1.965043	Prob(F-statistic)	0.000000

Tab A.11 Estimation du modèle élu : SARIMA (3; 0; 3) (0,1,1) avec $\varphi_1 = \varphi_2 = 0$



Auto corrélation	Corrélation Partielle		AC	PAC	Q-Stat	Prob.
		1	0.002	0.002	0.0006	
		2	-0.055	-0.055	0.7151	
		3	-0.020	-0.020	0.8098	
		4	0.019	0.016	0.8960	
		5	0.053	0.051	1.5643	
		6	-0.044	-0.043	2.0236	0.155
		7	0.029	0.036	2.2289	0.328
		8	0.000	-0.003	2.2289	0.526
		9	-0.004	-0.004	2.2328	0.693
		10	-0.071	-0.072	3.4652	0.629
		11	0.112	0.118	6.5517	0.364
		12	-0.001	-0.017	6.5520	0.477
		13	0.098	0.115	8.9441	0.347
		14	-0.043	-0.044	9.3995	0.401
		15	0.051	0.074	10.058	0.435
		16	0.014	-0.012	10.107	0.521
		17	-0.105	-0.086	12.862	0.379
		18	0.049	0.034	13.479	0.412
		19	-0.007	-0.006	13.492	0.488
		20	0.091	0.076	15.607	0.409
		21	-0.107	-0.089	18.520	0.294
		22	-0.182	-0.185	27.029	0.058
		23	0.040	0.045	27.454	0.071
		24	0.054	0.006	28.203	0.080
		25	0.021	0.035	28.316	0.102
		26	-0.008	-0.010	28.332	0.131
		27	0.017	0.025	28.408	0.163

Fig A.12 Corrélogramme et corrélogramme partiel des résidus



Auto corrélation	Corrélation Partielle		AC	PAC	Q-Stat	Prob
		1	0.247	0.247	14.321	
		2	0.055	-0.006	15.034	
		3	0.143	0.139	19.856	
		4	0.074	0.006	21.145	
		5	0.022	0.000	21.257	
		6	0.112	0.097	24.254	0.000
		7	0.043	-0.019	24.690	0.000
		8	-0.011	-0.019	24.719	0.000
		9	0.097	0.088	27.006	0.000
		10	0.015	-0.045	27.060	0.000
		11	0.095	0.121	29.276	0.000
		12	0.013	-0.076	29.319	0.000
		13	0.054	0.073	30.044	0.000
		14	0.197	0.173	39.689	0.000
		15	0.151	0.046	45.395	0.000
		16	0.137	0.107	50.127	0.000
		17	0.117	0.008	53.544	0.000
		18	0.022	-0.053	53.666	0.000
		19	-0.007	-0.011	53.680	0.000
		20	0.196	0.146	63.455	0.000
		21	0.047	-0.040	64.014	0.000
		22	0.042	0.044	64.476	0.000
		23	0.056	-0.030	65.293	0.000
		24	-0.015	-0.040	65.349	0.000
		25	-0.007	-0.021	65.364	0.000
		26	0.007	-0.040	65.376	0.000
		27	0.029	0.033	65.594	0.000

Fig A.13 Corrélogramme et corrélogramme partiel des résidus carrés

N°Obs.	Année Et mois	Prévision VEDF	Réalisation REAL	REAL - VEDF	(REAL-VEDF)^2	REAL- $\overline{REAL}$
247	2004:07	14632848.2678	14077116	555732	308838055824	-539553,9
248	2004:08	15812839.1539	14542408	1270431	1613994925761	-74261,9
249	2004:09	12540831.6735	10835791	1705040	2907161401600	-3780878,9
250	2004:10	15293538.1807	12002959	3290579	10827910155241	-2613710,9
251	2004:11	16553069.2797	15084737	1468332	2155998862224	468067,1
252	2004:12	13786331.2714	12810994	975337	951282263569	-1805675,9
253	2005:01	16964191.6743	19182011	-2217820	4918725552400	4565341,1
254	2005:02	16880881.7089	18411711	-1530830	2343440488900	3795041,1
255	2005:03	14324189.2526	13629048	695141	483221009881	-987621,9
256	2005:04	16778045.925	15619071	1158974	1343220732676	1002401,1
257	2005:05	17965788.7171	15545358	2420430	5858481384900	928688,1
258	2005:06	14029066.4984	12198798	1830268	3349880951824	-2417871,9
259	2005:07	15807220.7377	15792681	14539	211382521	1176011,1
260	2005:08	16615017.4347	16288953	326064	106317732096	1672283,1
261	2005:09	13925139.0454	13228413	696726	485427119076	-1388256,9

Tab A.14 Prévision des 15 valeurs à l'aide du modèle retenu avec la modélisation par Box et Jenkins

Architecture	# of Weights	Fitness	Inverse Test error	Akaike's criterion	R-squared	Correlation	Train error	Stop reason
[7-3-1]	28	0,000736	795749,875	0,000736	0,905396	0,951538	845614,1875	All iterations done
[7-28-1]	**253**	**0,000555**	**831982,4375**	**0,000555**	**0,918727**	**0,958506**	**811104,5**	**All iterations done**
[7-18-1]	163	0,000615	855308,1875	0,000615	0,912415	0,955387	833363,8125	All iterations done
[7-12-1]	109	0,000659	827191	0,000659	0,912673	0,955664	827534,25	All iterations done
[7-8-1]	73	0,000691	793955,5	0,000691	0,908486	0,953256	837031,4375	All iterations done
[7-6-1]	55	0,000709	839052,9375	0,000709	0,915217	0,956675	825930,1875	All iterations done
[7-4-1]	37	0,000729	830511,1875	0,000729	0,914315	0,956199	814618	All iterations done
[7-5-1]	46	0,000719	806327,5	0,000719	0,912107	0,95519	825044,375	All iterations done

Tableau A.15 Comparaison des architectures trouvées à la phase de construction de réseau.

N°Obs.	Année Et mois	Prévision VEDF avec Réseau de neurones	Réalisation REAL	(REAL-VEDF)^2	REAL- $\overline{REAL}$
247	2004:07	13921284,06	14077116	24283593524,1636	291118411005,21
248	2004:08	16150215,974585	14542408	2585046483139,120005922225	5514829791,61
249	2004:09	11380058,440656	10835791	296227046958,232481710336	14295045256465,21
250	2004:10	14690744,286607	12002959	7224189746901,073133572449	6831484668778,81
251	2004:11	16666538,875853	15084737	2502097174452,069624477609	219086810102,41
252	2004:12	15376245,926663	12810994	6580517447248,233530315569	3260465455840,81
253	2005:01	17504255,720011	19182011	2814862779530,967783840121	20842339359349,21
254	2005:02	16750873,483987	18411711	2758381254596,232031416169	14402336950689,21
255	2005:03	14267524,852189	13629048	407652690781,174154091721	975397017359,61
256	2005:04	15592692,339887	15619071	695833709,357177172769	1004807965281,21
257	2005:05	18611127,350528	15545358	9398941710636,874933878784	862461587081,61
258	2005:06	11893835,29182	12198798	93002253380,4798389124	5846104524809,61
259	2005:07	14495381,1854	15792681	1682986808961,19437316	1383002107323,21
260	2005:08	15995614,211658	16288953	86047644745,952675108964	2796530766545,61
261	2005:09	13130400,439938	13228413	9606461929,907157443844	1927257220397,61

Tab A.16 Prévision des 15 valeurs à l'aide du modèle neuronal (Apprentissage avec la méthode de Levenberg-Marquardt)

ANNEXE 2. FONCTIONNALITES DU LOGICIEL ALYUDA NEUROINTELLIGENCE

Dans ce qui va suivre, on expliquera brièvement les fonctionnalités du logiciel Alyuda Neurointelligence utilisé le long de ce travail pour le traitement par la méthode neuronale.

Pour traiter des données avec les réseaux de neurones, ce logiciel comprend plusieurs étapes:

La phase d'Analyse

- Les données sont classées par type en colonnes
- On détecte les anomalies liées aux données
- Partitionner les données en trois ensembles : ensemble d'apprentissage, ensemble de validation et ensemble test
- On définit la colonne cible (target column**)**

La phase de prétraitement des données

C'est une phase très importante dans laquelle le logiciel à cause des plages de valeurs que prend une fonction d'activation, change l'échelle des entrés et/ou des sorties du réseau. Plus précisément, après avoir chargé les données numériques brutes sous forme de colonnes, ces dernières sont automatiquement prétraitées durant cette phase. Par défaut, les valeurs numériques sont normalisées en utilisant les formules suivantes :

- SF = (SRmax-SRmin)/(Xmax-Xmin)
- Xp = SRmin + (X-Xmin)* SF où
- X : est la valeur actuelle d'une colonne numérique.
- Xmin : minimum que la valeur actuelle d'une colonne peut prendre
- Xmax : maximum que la valeur actuelle d'une colonne peut prendre
- SRmin : limite inférieure de l'intervalle de l'échelle
- SRmax : limite supérieure de l'intervalle de l'échelle
- SF : coefficient de l'échelle
- Xp : la valeur prétraitée ou valeur de X après prétraitement

Pour les colonnes d'entrées (input), la plage de l'échelle est [-1..1]. Pour la colonne cible, la plage de l'échelle dépend de la fonction d'activation choisie et dans notre cas, elle est linéaire donc les valeurs restent inchangées et ne sont pas prétraitées. La colonne cible sera prétraitée automatiquement si au cours de l'étude, on change de fonction d'activation. Par défaut, Xmin et Xmax sont prises de l'ensemble des entrées. Si on sait que les données futures pour la prévision seront inférieures au le minimum et supérieures au maximum présentés dans l'ensemble des entrées, on peut changer les valeurs minimum et maximum utilisées dans l'échelle. On peut aussi réserver une place (a headroom) dans le calcul des échelles pour les entrées qui formeront notre requête qui peuvent excéder les valeurs MIN et MAX de la colonne des entrées. Il représente le pourcentage de la plage à réserver. La spécification du headroom est ajoutée aux limites sup et inf de l'intervalle. Par exemple, si le MIN est 0 et le MAX est 10, alors le Headroom de 10% fait qu'il est permis pendant les calculs à l'échelle des entrées de varier de -1 à 1 (10% de l'intervalle est ajouté au dessus et au dessous).

La phase de Construction de réseau

Aide à construire l'architecture la plus efficace de réseau de neurones. On peut y exécuter des algorithmes de recherche d'architecture, gérer l'ensemble du réseau, analyser les graphes d'apprentissage et obtenir les statistiques pour les réseaux candidats.
Deux méthodes sont disponibles afin de faire cette recherché:

- **La recherché heuristique** mais cette méthode ne fonctionne que pour un réseau à trois couches (1 seule couche cachée).
- **La recherche Exhaustive** effectue une recherche exhaustive parmi toutes les topologies possibles qu'on spécifie; la méthode fonctionne pour les réseaux de neurones qui ont jusqu'à 5 couches cachées.

La recherché de la meilleure architecture peut prendre un temps considérable en fonction des paramètres de la méthode de recherché, le nombre d'entrées, la quantité de données et bien sur de la puissance de l'ordinateur utilisé. Notons que par défaut, Neuro Intelligence propose une topologie avec une couche cachée et avec le nombre d'unités cachées égal à la moitié du nombre d'entrées (3 dans notre cas). Bien sur, ce n'est pas toujours recommandé d'utiliser cette règle et on recommande d'exécuter une recherche en expérimentant plusieurs architectures. Si on ne dispose pas d'information complémentaire sur le niveau de la complexité du problème, commencer avec une couche cachée et poser le nombre d'unités cachées égal à la moitié du nombre d'entrées. Ajouter alors graduellement des unités cachées.

Le nombre maximum d'unités caches excède rarement quatre fois le nombre d'entrées. On recommande strictement d'entraîner ou réapprendre chaque l'architecture au moins 3 fois (recommandé jusqu'à 10 fois) avec des différents poids initiaux et de ne garder que le meilleur pour la comparaison avec les autres architectures On recommande aussi de tester avec des partitions de données différentes et ne garder que le meilleur résultat pour une comparaison future avec d'autres architectures. Le critère de Fitness spécifie quel paramètre du réseau serait utilise pour distinguer le meilleur réseau. Les critères suivants sont disponibles :

- **L'erreur Test**: plus l'erreur est petite, plus le réseau est meilleur. Ce paramètre est calculé comme l'inverse de l'erreur moyenne absolue du réseau sur l'ensemble test.
- **Le critère d'Akaike (le AIC)** : plus le AIC est petit, plus le réseau est meilleur.
- **R-squared**: plus cette valeur est proche de 1 plus le réseau est meilleur
- **Corrélation**: plus cette valeur est proche de 1 plus le réseau est meilleur.

Dans le module "design an architecture"il y a deux options : Evaluation qui définit comment évaluer les architectures candidates. Le nombre retrains"spécifie combien d'exécutions (ou de réentraînements du réseau) avec différents poids initiaux e seront effectuées pour chaque configuration d'architecture. Par exemple, si on pose retrains=3 par configuration, chaque architecture sera ré entraînée 3 fois et le meilleur résultat parmi ces 3 réentraînements sera sélectionné pour comparer avec les résultats d'autres topologies.

La phase d'apprentissage

Avec NeuroIntelligence sept (07) algorithmes d'apprentissage sont disponibles:

1. Quick propagation
2. Conjugate Gradient Descent
3. Quasi-Newton,
4. Limited Memory Quasi-Newton,
5. Levenberg-Marquardt,
6. Incremental back propagation,
7. Batch back propagation.

Il n'existe pas un unique meilleur algorithme d'apprentissage pour les réseaux de neurones. Tout dépend du problème à traiter et en général, les règles suivantes, très simples ont prouvé leur efficacité pour la plupart des cas pratiques rencontrés.

1. Si on a un réseau de neurones avec un petit nombre de poids (en général jusqu'à 300), l'algorithme de Levenberg-Marquardt est souvent très performant par sa vitesse et par le fait qu'il trouve les meilleurs optima par rapport aux autres algorithmes. Mais sa capacité mémoire est proportionnelle à la racine carrée du nombre de poids. En plus, Levenberg-Marquardt est limité par le fait qu'il est spécialement conçu pour minimiser la somme des erreurs au carré et ne peut être utilise pour d'autres types d'erreurs de réseau.
2. Avec un nombre modéré de poids, on peut utiliser l'algorithme de Quasi-Newton et celui de Limited Memory Quasi-Newton qui sont efficaces. Mais là aussi la capacité mémoire doit être proportionnelle à la racine carrée du nombre de poids
3. Avec un grand nombre de poids, on recommande d'utiliser l'algorithme Conjugate Gradient Descent qui a presque la même vitesse de convergence que celle des méthodes de second ordre, en évitant le calcul et le stockage de la matrice Hessienne. Sa capacité mémoire est proportionnelle à la racine carrée du nombre de poids.
4. les algorithmes de Conjugate Gradient Descent et Quick Propagation sont à usage général pour l'apprentissage et ce, par choix.
5. On peut choisir les algorithmes incrémental et batch Back propagation pour les réseaux de n'importe quel taille. L'algorithme de Back propagation est l'algorithme le plus populaire pour l'apprentissage des perceptrons muticouches et est souvent utilisé par les chercheurs et les praticiens. Les principaux défauts du back propagation sont: une convergence lente, le besoin de régler le taux d'apprentissage et les momentum, et un forte probabilité d'obtenir les minima locaux. Incrémental back propagation peut être efficace pour de larges ensembles de données si on sélectionne correctement le taux d'apprentissage et le momentum. Généralement, il est plus performant que l'algorithme batch back propagation.

Dans cette phase, le logiciel donne une représentation visuelle du processus d'apprentissage, on peut y régler l'apprentissage par des graphes, l'histogramme de la distribution de l'erreur, l'histogramme de distribution des poids et les détails de l'apprentissage en temps réel.
L'histogramme de la distribution de l'erreur montre combien d'enregistrements d'ensembles d'apprentissage ont des valeurs d'erreurs différentes. Typiquement, Durant l'apprentissage le nombre des grandes erreurs devrait décroître et le nombre des petites erreurs devrait croître**.** L'histogramme de distribution des poids montre le nombre de poids de diverses tailles**.** A la

fin de l'apprentissage, on peut aussi inspecter combien de poids sont près de la valeur zéro ou ont de grande valeurs et faire quelques conclusions sur l'architecture sélectionnée.
Pour ce travail, les algorithmes d'apprentissage choisis étaient : Levenberg-Marquardt, Quick Propagation et batch Back propagation.

La phase de Test

Aide à analyser la performance du réseau entraîné. On peut y comparer les valeurs actuelles vs les valeurs de sortie en termes de graphes ou matrice.

La phase de requête

Permet de demander au réseau de neurones entraîné d'effectuer une requête en introduisant de nouvelles données. Les résultats sont alors représentés dans une table et dans un graphe. Notre requête sera de prévoir 15 valeurs de la série de consommation d'électricité VED.

BIBLIOGRAPHIE

Attali J., Pagès G. (1995a): Approximation of functions by perceptrons, a new approach, neural processing letters, 22, 5-19.

Attali J., Pagès G. (1995b) : Fonctions de Lyapounov et loi des grands nombres pour les fonctions non bornées d'une chaîne de Markov stable, Preprint.

Azencott R., Girard Y. Astier R. Baudin M. Girard B. Jakubowicz P. Martin M. (1991) : Mandrake, un logiciel expert en analyse des séries temporelles, Harcourt Brace and World, Paris.

Azoff E. M. (1994): Neural network time series ferecasting of financial markets, Wiley Finance edition, New York.

Bishop C. M. (1995): Neural networks for pattern recognition. Oxford, Clarendon Press.

Biganzoli E., Boracchi P., Mariani L., Marubini E. (1998): Feed Forward Neural Networks for the analysis of censored survival data: a partial logistic regression approach' Statistics in Medicine, 17, 1169-1186

Box G. E. P., Jenkins G. M. (1976): Time series analysis; forecasting and control. San Francisco, Holden-Day.

Breidt F.J., Davis R.A., Lii K.S., Rosenblatt M. (1991): Maximum likelihood estimation for non causal autoregressive processes, J. Multivariate Analysis 36, 175-198.

Brown S.F., Branford A.J., Moran (1997): On the use of Artificial Neural networks for the Analysis of Survival Data, IEEE Transactions on Neural Networks, 8, 1071-1077

Chuc H. (1994): Neural network system for forecasting method selection, Decision Support Systems, 12, 13-22.

Ciampi A., Lechevallier Y. (2002): Application des réseaux de neurones aux données censurées, Mac Gill University, Canada et INRIA, France.

Cottrel M., Girard B., Girard Y., Mangeas M., Muller C. (1995): Neural modeling for time series, a statistical stepwise method for weight elimination, IEEE transaction on neural networks, 6, 1335-1364.

Crone, S. F. (2002): Training Artificial Neural Networks using Asymmetric Cost Functions. Computational Intelligence for the E-Age. Singapore, IEEE: pp. 2374-2380.

Deutch M., Granger O.W., Terasvirta. J. (1994): The combination of forecasting using changing weights, International journal of forecasting, l10, 1, June, 47-57.

Doukhan P. (1994a): Mixing properties and examples, vol.85 of L.N.S, Springer Verlag, Berlin.

Doukhan P. (1994b): Mixing, properties and examples, Springer Verlag, L.N.S.

Doukhan P., Tsybakov A., (1993): Non parametric recursive estimation non linear ARX-models, problems of information transmission, 29, 4: 318-327.

Doukhan P., Ghindes M., (1992) : Etude des processus $X_n = f(X_{n-1}) + \varepsilon_t$, thèse de 3^{ième} cycle université Paris XI.

Dreyfus G., Samuelides M., Martinez J.M., Gordon M.B., Badran F., Thiria S., Laurent H. : Réseaux de neurones. Méthodologie et applications. Eyrolles.

Duflo M. (1996) : Algorithmes stochastiques mathématiques et applications, Springer Verlag, Berlin.

Duflo M., Senoussi R., Touati A. (1990) : Sur la loi des grands nombres pour les martingales vectorielles et l'estimateur des moindres carrés d'un module de régression. Annales I. H .P, 26, 549-566.

Duflo M. (1990) : Méthodes récursives aléatoires, Masson, Paris.

Granger C. W.J. (1994): Forecasting in economics. In Weigend A. S., Gershenfeld N. A. editors, Time series prediction: Forecasting the future and understanding the past, 529-538, Reading, MA. Addison-Wesley.
Faraway J., Chatfield C. (1998): Time Series Forecasting with Neural Networks: A Comparative Study Using the Airline Data, Applied Statistics, Vol. 47, No. 2 (1998), pp. 231-250.
Fildes R. (2002): Forecasting competitions, their role in improving forecasting practice and research. A companion to economic forecasting, Malden, Mass. [u.a.], Blackwell.
Guégan D. (1994) : Séries chronologiques non linéaires à temps discret, volume statistique mathématique et probabilités. Economica.
Guyon X. (1992) : Champs aléatoires sur un réseau, Modélisation statistique et applications, Masson.
Guyon X. (1995): Random fields on a network modeling, statistics and applications Springer Verlag, Berlin.
Hannan E., Kavalieris L. (1986): Regression, autoregressive models, J. Time Series analysis, 7, 27-49.
Hertz J., Krogh A., Palmer R.G. (1991): Introduction to the theory of neural computation, Addison-Wesley, Reading, MA.
Hornik K., Stinchcombe M., White H. (1989): Multilayer feedforward networks are universal approximators, neural networks, 2, 359-366.
ItoY. (1991): Representation of function by superpositions of step or sigmoid function and their applications to neural networks theory, neural networks, 4, 385-394.
Jones D. (1978): Non linear autoregressive processess, Journal of the Royal Society, A, 360, 71-95.
Kaastra I., Boyd M. (1996):"Designing a neural network for forecasting financial and economic time series." Neurocomputing 10(3): 215-236.
Klimbo L., Nelson P. (1978): On conditional least squares estimation for stochastic processes, Annales Statistiques, 6 , 629-642.
Lai T., Wei C. (1983): Asymptotic properties of general autoregressive models and strong consistency of least squares estimates and their applications, J. Multivariate Analysis, 13, 1-23.
Lai T. (1994): Asymptotic properties of non linear least squares estimates in stochastic regression models, Ann.Statistis. 22, 1917-1930.
Lapedes A., Farber R. (1987): Non linear signal processing using neural networks, prediction and system modelling, Technical Report, Los Alamos LA-UR 87-2662. Los Alamos National Laboratory, Los Alamos? NM.
Lapedes A., Farber R. (1988): How neural nets work, .technical report, Los Alamos National Laboratory, NN875545 January 1988.
LeCun Y. (1989): Generalisation and network design strategies. In Pfeifer R., Schreter Z., Fogelman F., Steels L. editors, Connectionnism in Perspective, Amsterdam. North Holland.
LeCun Y., Simard P.Y., Pearlmutter B. (1993): Automatic Learning Rate Maximisation by On-Line Estimation of the Hessian's Eigenvectors Advances in Neural Information Processing Systems 5, 1993, p. 156-157.
Leshno M., Lin V.Y., Pinkus A., Schoken S. (1993): Multilayer feed forward networks with non polynomial activation function can approximate any function, Neural Networks, 6, 861-867.
Mangeas M., Yao J. (1996) : Sur l'estimateur des moindres carrés d'un modèle autorégressif non linéaire, Technical Report 53, SAMOS, université Paris I.

McCulloch W. S., Pitts W. (1943): A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133.
Meyn S., Tweedie R. (1993): Markov chains and stochastic stability, Springer Verlag, Berlin.
Minsky M., Papert S. (1969): Perceptrons, M I T Press, Cambridge (E. U)
Mokkadem A. (1987a) : Sur un modèle autorégressif non linéaire: ergodicité et ergodicité géométrique, J. Time Series Analysis, 8, 2 ,195-204.
Robinson P. (1977): The estimation of non linear moving average models, Processes and their applications, 1, 81-90.
Rosenblatt F. (1958), "The perceptron: a probabilistic model for information storage and organization in the brain", - repris dans J.A. Anderson & E. Rosenfeld (1988), Neurocomputing. Foundations of Research, MIT Press.
Rosenblatt F. (1962): Principles of Neurodynamics, Spartam, New York.
Tong H. (1990): Non-linear time series, a dynamical system approach, Oxford University Press.
Roynette B. (1993) : Vitesse d'approximation d'une fonction par un réseau de neurones , pré-publication de l'université Nancy 1.
Rumelhart D., Hinton G., Williams R. (1986): Learning representations by backpropagating errors, Nature, 323, 533-536.
Senoussi R. (1990) : Statistique asymptotique presque sure de modèles convexes, Annales I. H. P, probabilités et statistiques, 26, 19-44.
Tang, Z., Fishwick P. A., (1993): "Feed-forward Neural Networks as Models for Time Series Forecasting." ORSA Journal on Computing 5(4): 374-386.
Turing A. (1945): ProposaIs for Development in the Mathematics Division of an Automatic Computing Engine (A C.E.), Rep. E.882. Executive Committee, National Physical Laboratory, Teddington, Middlesex (1945).
Turing A. (1947): Intelligent Machinery. [Réédition : Machine Intelligence, (B. Meltzer et D. Michie, eds), Vol. 5, p. 3-23. Edinburgh Univ. Press, Edinburgh, 1969].
Von Neumann J. (1945): First Draft of a Report on the EDVAC, Contract No.W-670-0RD-4926. Moore School of Electrical Engineering, Univ. of Pennsylvania, Philadelphia, Pennsylvania.
Weigend A.S., Huberman B.A., Rumelhart D. E. (1990): Predicting the future: a connexionnist approach, international journal of neural systems, 1, 193-209.
White H. (1992): Artificial neural networks, approximations and learning theory, Blackwell.