UNIVERSITÉ PARIS I
PANTHÉON-SORBONNE
U.F.R. DE MATHÉMATIQUES ET INFORMATIQUE

THÈSE

présenté par

Morgan MANGEAS

en vue de l'obtention du titre de

Docteur en Sciences

arrété du 5 juillet 1984

spécialité : **Mathématiques**

Propriétés Statistiques des Modèles Paramétriques Non-linéaires de Prévision de Séries Temporelles

Application aux Réseaux de Neurones à Propagation Directe

jury :	Marie	COTTRELL	*Examinateur*
	Xavier	GUYON	*Examinateur*
	Michael	JORDAN	*Examinateur*
	Christian	JUTTEN	*Rapporteur*
	Corinne	MULLER	*Examinateur*
	Jean-Pierre	RAOULT	*Rapporteur*

Remerciements

Me voici à l'aube de conclure un travail de trois années de thèse et de seize mois de service national. Ces années ont été pour moi heureuses et m'ont permis de compléter ma formation universitaire, au rythme des mes rencontres, de mes lectures et des aléas de la recherche. A la place de l'habituelle et formelle page de remerciements, permettez-moi de retracer le chemin parcouru, et de remercier au passage les personnes qui m'ont aidé et soutenu tout au long de cette thèse.

Mes premières pensées vont à ma famille, à mes parents, à ma sœur Jessica, qui m'ont soutenu et encouragé tout au long de mes études. Il m'est appréciable de pouvoir compter sur eux, et ce climat de confiance a permis au rêveur que je suis de me concentrer sur mes études et mon travail de thèse.

Après le D.E.A., le professeur Marie Cottrell me mit en contact avec Corinne Muller à propos d'un stage à la Direction des Études et Recherches (DER) d'EDF sur la prévision de consommation électrique. Ce sont ces deux personnes qui, par leur patience, leur disponibilité et leur constant souci de faire avancer nos recherches m'ont poussé à donner le meilleur de moi-même. A l'époque, Gérard Hatabian, alors chef du Groupe *Statistiques Optimisation et Aide à la Décision* (SOAD) de la DER m'accorda sa confiance et donna son aval pour le financement d'une thèse et me permit, lui puis Hervé Augendre, de travailler dans d'excellentes conditions, tant sur le plan matériel que relationnel. Du côté universitaire, l'équipe s'étoffa de deux personnes hors du commun, les professeurs Yvonne et Bernard Girard, qui m'indiquèrent les premières voies de recherches. Leurs compétences dans le domaine de la prévision de séries temporelles et leur énergie à résoudre les premiers problèmes rencontrés ont été pour moi un exemple et une motivation. Vint ensuite le professeur Xavier Guyon. Qu'il excuse le piètre élève que j'ai été. A cette époque, mes pensées étaient déjà dirigées vers le Colorado, où je devais effectuer mon service national. Encore une fois ce fût grâce à la confiance et au soutien amical de Corinne Muller que je dois cette aventure.

Sur place, à l'université du Colorado à Boulder, j'ai été confronté à d'autres méthodes de travail et de communication. Plusieurs personnes m'ont permis de continuer mon travail de recherche et de compléter mes connaissances. Citons Andréas Weigend, mon *advisor* aux Etats-Unis, Mike Mozer et toutes la sympathique et dynamique équipe du *Boulder Connexionnist Research Group*. Je revins en France une semaine pour une école d'été organisée par EDF et j'y fis la connaissance de Michael Jordan qui inspira une partie de mes travaux sur les modèles modulaires. Je le remercie grandement pour ses clairs exposés et pour m'avoir fait la sympathie de partager quelques instants musicaux avec moi et mes amis.

Une fois revenu en France, je repris le travail théorique debuté avec Xavier Guyon, en collobaration avec Jian-Feng Yao. Ses grandes qualités de rigueur ont permis d'obtenir rapidement des résultats concrets. Je lui dois rien moins qu'un des chapitres de ma thèse, et au risque de contrevenir aux règles de discrétion que lui et Xavier Guyon s'imposent, j'aimerais les remercier tous deux pour leur précieuse aide. Enfin je tiens à remercier chaleureusement Marie Cottrell, mon directeur de thèse, pour l'attention portée à mon travail et pour sa lecture attentive de ce document. Ses conseils et ses remarques se sont souvent avérés judicieux et ont grandement aidé à améliorer ce mémoire.

Je voudrais remercier aussi Christian Jutten et Jean-Pierre Raoult pour avoir accepté la charge de rapporter cette thèse, pour leurs remarques et leurs encouragements.

Et un clin d'œil à mon amie, Murielle, qui a traversé ces années avec moi, aux soadiens (le groupe le plus dynamique et sympathique de la DER, Corinne, Daniela, Didier, Arman, Cecile, Yves, Christiane,...) et à mes ami(e)s de toujours, Christian et sa famille, Roger, et à tous les autres...

Table des matières

Glossaire relatif aux réseaux de neurones

- **algorithme génétique :** algorithme itératif et stochastique d'optimisation globale, dont le fonctionnement est le suivant : un ensemble (une population) de solutions potentielles (des individus) évolue selon un principe analogue à l'évolution naturelle. Les individus les mieux adaptés (pour lesquels la valeur de la fonction à maximiser est supérieure à celle des autres) se reproduisent plus que les autres, certains des moins performants ne se reproduisant pas du tout, et la reproduction implique un changement dans la représentation informatique de ces solutions (mutation, ou croisement) ;

- **apprentissage supervisé :** procédure d'apprentissage d'un certain nombre d'exemples d'entrées/sorties consistant à régler les paramètres d'un réseau de neurones, de façon à minimiser l'erreur entre les sorties effectives de ce réseau et les sorties désirées ;

- **base d'apprentissage :** base de données contenant des couples d'entrées-sorties servant à déterminer les valeurs des paramètres d'un réseau de neurones lors de la phase d'apprentissage supervisé ;

- **base de généralisation** ou **base de test :** base de données constituée d'observations qui ne sont pas utilisées lors de l'apprentissage du réseau et qui sont totalement indépendantes de la structure et des poids du réseau. Cette base sert à estimer la capacité de généralisation du réseau de neurones ;

- **base de validation :** base de données constituées d'observations qui ne servent pas directement pour l'apprentissage, mais qui influencent la valeur finale des poids ou l'architecture du réseau ;

- **cellule** ou **neurone :** unité de base composant les réseaux de neurones ;

- **connexion** ou **synapse :** liaison pondérée entre deux cellules (ou neurones artificiels) permettant de transférer l'information d'une cellule vers une autre cellule ;

- **couche :** ensemble de neurones non connectés entre eux recevant leurs entrées des neurones de la couche précédente et envoyant leurs sorties aux neurones de la couche suivante ;

- **couche d'entrée :** première couche d'un réseau de neurones. Cette couche de cellules n'a pas de connexions incidentes et reçoit comme signaux les données de la base d'apprentissage. D'autre part, la fonction d'activation des neurones de cette couche est la fonction identité ;
- **couche cachée :** couche située entre la couche d'entrée et la couche de sortie (il peut y avoir plusieurs couches cachées) ;
- **critère d'erreur, fonction d'erreur** ou **fonction de coût :** fonction utilisée pour mesurer l'erreur entre les sorties désirées et les sorties délivrées par le réseau de neurones. La fonction de coût classiquement utilisée est la somme des erreurs quadratiques réalisées sur la base d'apprentissage ;
- **descente du gradient :** algorithme de minimisation de la fonction de coût dans le cadre d'un apprentissage supervisé. Le principe consiste à calculer le gradient de la fonction de coût par rapport aux paramètres du réseau (les poids synaptiques) et à les modifier dans la direction opposée au gradient ;
- **fonction de transfert** ou **fonction d'activation** (d'un neurone) **:** fonction appliquée à la sortie du neurone (dans le modèle de McCullogh et Pitts, c'est une fonction seuil) ;
- **fonction logistique :** fonction sigmoïde à valeurs dans [0,1] ;
- **généralisation :** capacité d'un réseau de neurones entraîné sur une base d'apprentissage à répondre correctement à des données non présentées lors de l'apprentissage ;
- **perceptron multicouches :** un des modèles de réseaux de neurones le plus utilisé, représenté par un graphe acyclique orienté, composé de couches successives (au moins une couche cachée) ;
- **poids :** pondération de la liaison (ou synapse) entre neurones ;
- **rétropropagation** (algorithme de) **:** algorithme fournissant un calcul exact du gradient de la fonction à optimiser pour les réseaux de neurones à propagation directe ;
- **sigmoïde :** fonction continue, strictement croissante, bornée supérieurement et inférieurement (voir équation 4.2) ;
- **sur-ajustement :** ajustement trop important des données conduisant à une mauvaise généralisation (*overfitting* en anglais).
- **sur-apprentissage :** fait de sur-apprendre les données de la base d'apprentissage (bruit inclus) ce qui conduit à un sur-ajustement.

Chapitre 1

Introduction

1.1 Cadre de l'étude

L'un des premiers réflexes du chercheur, lorsqu'il s'agit d'étudier un phénomène physique, économique ou social, est de le quantifier et de le traduire sous la forme d'une suite de nombres capable d'être manipulée et analysée. C'est ainsi que des outils mathématiques sont apparus ayant pour but de mettre en équations des phénomènes afin de les reproduire et de les comprendre.

Un grand nombre de phénomènes ne sont décrits cependant que de manière partielle, sont difficiles à quantifier de manière fine ou fiable, ou encore sont soumis à des perturbations et des impondérables que nous nommerons dans le reste de ce document du *bruit*[1]. Ainsi est née la théorie des probabilités et statistiques, destinée à décrire des phénomènes dits *aléatoires*, par le biais non pas de leurs valeurs précises à un moment donné, mais de leur loi de probabilité et de leurs propriétés statistiques, considérées comme stables dès lors que les conditions initiales associées au phénomène restent les mêmes. Certains diront ici que l'aléatoire n'existe pas, et qu'il n'y a que des données manquantes ; nous n'entrerons pas dans ce débat. Au vu de notre savoir actuel, on peut cependant constater que nous n'avons pas une connaissance suffisante de la grande majorité des phénomènes observables autour de nous pour espérer en comprendre tous les mécanismes. Les probabilités et statistiques fournissent alors un cadre théorique extrêmement utile, permettant l'emploi de puissants outils d'analyse.

On s'intéresse plus particulièrement dans ce mémoire de thèse à des séries d'évènements quantifiables qui apparaissent, ou sont observés, à des intervalles réguliers comme par exemple le nombre de poussières nocives absorbées par un passant place de l'Étoile à 17h tous les jours, les tirages successifs du loto enregistrés toutes les semaines, ou la quantité d'électricité, exprimée en kiloWatt-heures, consommée en Bre-

1. Les bruits blancs, par exemple, sont par définition des évènements purement aléatoires de même loi, de même moyenne nulle et de même variance.

tagne chaque année. Toutes ces séries répondent à la définition de *série temporelle à temps discret*. Ces phénomènes sont néanmoins de natures différentes, les deux premiers étant à valeurs dans $\mathbb{N}$, le dernier à valeurs dans $\mathbb{R}$. Ils sont aussi observés à des intervalles de temps différents, le deuxième ayant la caractéristique que chaque tirage est complètement indépendant des tirages précédents, entraînant une évolution du phénomène purement aléatoire, alors que la quantité d'electricité consommée durant l'année 1996 est fortement corrélée à celles consommées les années précédentes. Pour la plupart des phénomènes chronologiques d'ailleurs, l'hypothèse d'indépendance des observations n'a le plus souvent aucun sens. On est donc amené à considérer des techniques prenant en compte les corrélations du phénomène avec son passé.

Pour analyser ces données, la modélisation consistant à mettre en équation la série et à déterminer ses caractéristiques statistiques constitue l'une des solutions les plus utilisées. Pour des suites de données assez régulières et lorsque la durée d'observation est assez longue, on peut alors dégager certaines propriétés asymptotiques[2] concernant le processus associé ou les méthodes statistiques de modélisation employées. C'est d'ailleurs l'un des buts que l'on se fixe ici. Après avoir modélisé et analysé la série, on peut tenter de prévoir les évolutions futures du phénomène, en évaluant les valeurs futures de la série. A l'opposé du voyant avec sa boule de cristal, qui prétend avoir des révélations spontanées plus ou moins précises sur des faits futurs, on utilise ici exclusivement la connaissance du passé pour prévoir l'avenir.

Dans ce document, on s'intéresse plus particulièrement à la *modélisation paramétrique* et à son utilisation dans *la prévision de séries temporelles*. La modélisation consiste dans ce cas à fixer la structure générale du modèle et à trouver le vecteur de paramètres s'adaptant le mieux aux données. Tout au long de ces dernières décennies, les modèles paramétriques linéaires, connus sous l'appellation d'*Auto-Regressive-Moving-Average* (*ARMA*[3]) introduite par *Box et Jenkins* dans les années 70 (Box et Jenkins, 1976), ont été largement développés, étudiés et expérimentés. Des systèmes experts et des logiciels automatisant toutes les étapes de construction tels que *MANDRAKE* (Azencott et al., 1991) ont été mis au point mettant à profit l'ensemble des connaissances théoriques et pratiques acquises. Néanmoins, si la modélisation linéaire peut rendre compte d'un nombre important de types de données, elle s'avère insuffisante pour maîtriser certaines dynamiques pour lesquelles la relation entre la valeur à un instant donné de la série et les valeurs passées est de nature non-linéaire. Ces phénomènes sont pourtant nombreux et variés et sont présents dans la plupart des domaines d'application, tels que la médecine, la finance, l'économie, etc.

Parmi les nombreux modèles non-linéaires existants, les modèles basés sur les *réseaux de neurones* sont de plus en plus étudiés. Historiquement, c'est par leur capacité à résoudre certains problèmes par apprentissage qu'ils ont connu un succès important dans les années 80. Dans les années qui suivirent, certains les ont présentés comme un outil fabuleux, capable de résoudre les tâches les plus complexes. Si certaines applica-

2. Valides pour de longues séries.
3. Il existe plusieurs variantes telles que les modèles incluant des variables explicatives (modèle *ARMAX*).

tions basées sur les réseaux de neurones sont reconnues comme efficaces et opérationnelles de nos jours, on est cependant loin des résultats annoncés. En particulier, s'il est vrai qu'il est facile, en suivant les méthodologies neuronales standards, d'obtenir rapidement des résultats cohérents, il n'est en général pas aisé de surpasser en performance les méthodes classiques. Sur ce point les réseaux de neurones ont apporté un éclairage nouveau sur l'utilisation de modèles non-linéaires. Il est simple en effet, en utilisant des principes connexionistes, de concevoir des modèles de plus en plus complexes. Encore faut-il maîtriser et employer cette complexité de manière judicieuse. Ainsi, la force de certains types de modèles neuronaux, capables d'approximation universelle, s'avère être quelquefois un lourd handicap. Par exemple, utiliser un modèle de complexité trop grande pour résoudre un problème pour lequel on dispose de trop peu de données, entraîne souvent une mauvaise adéquation entre le modèle et le phénomène réel. Ce problème est à tel point aigu pour les réseaux de neurones, qu'on s'apercoit que le secret d'une bonne modélisation réside plus en général dans la définition d'une structure de modèle adéquate que dans l'emploi d'une méthode d'apprentissage performante. C'est précisément sur ce point que les réseaux de neurones déçoivent : ils sont en effet loin d'être le modèle universel s'adaptant à tous les types de problèmes pour peu que l'on utilise suffisamment de neurones et un bon algorithme d'apprentissage.

Les réseaux de neurones ont cependant apporté une vision nouvelle dans des domaines comme la classification, la modélisation et la prévision de séries temporelles, la régression simple ou la reconnaisance de forme. La manière habituelle d'aborder de tels problèmes réside généralement dans la résolution de systèmes plus ou moins complexes d'équations. L'approche neuronale est plutôt basée sur une manipulation visuelle des variables en modifiant les structures des modèles, rendant les méthodes plus aisées à appréhender, et plus accessibles aux non-professionnels des mathématiques et aux personnes rebutées par les équations.

Outre cet aspect technique, d'autres atouts plus théoriques ont contribué au succès des réseaux de neurones. En particulier un réseau de neurones à propagation directe connu sous le nom de *perceptron multicouches* a rapidement suscité l'intérêt des mathématiciens en s'inscrivant dans des cadres généraux classiques (approximation fonctionnelle, processus de Markov, algorithmes adaptatifs, etc). L'analyse de ce modèle neuronal se révèle cependant difficile, notamment en raison de son caractère non-linéaire et met à jour des problèmes de nature nouvelle. D'autre part, si les simulations, voire les applications opérationnelles, sont la plupart du temps très développées, les analyses théoriques le sont peu, et beaucoup de questions sur leurs mécanismes de fonctionnement et leurs propriétés mathématiques restent encore sans réponse. Un des points sur lesquels les efforts se sont portés est l'étude de la capacité d'approximation de fonction des perceptrons multicouches. Il a été ainsi démontré qu'un perceptron multicouches avec une seule couche cachée pourvue d'un nombre suffisant de neurones, peut approcher n'importe quelle fonction continue sur un compact avec la précision souhaitée. Dans de ce mémoire de thèse, on tente en particulier de répertorier les propriétés statistiques de ce modèle, afin d'évaluer et d'améliorer à travers différentes techniques, ses capacités à effectuer de la prévision de séries temporelles.

1.2 Organisation de la thèse et motivations

Ce mémoire est organisé autour d'une idée directrice, qui est la prévision de séries temporelles par des modèles paramétriques non-linéaires et en particulier par des modèles de réseaux de neurones à propagation directe. Autour de cette idée, s'articulent quatre parties pouvant être lues séparément, suivant les connaissances et les aspirations du lecteur. Néanmoins, le plan est conçu de telle manière que les chapitres s'enchaînent, passant graduellement de la théorie à la pratique, et d'une vision macroscopique du sujet à une vision plus détaillée.

On distingue :

- les chapitres 2 et 3 traitant de la modélisation paramétrique en général et non-linéaire en particulier ;
- les chapitres 4, 5, 6 détaillant le perceptron multicouches, ses propriétés statistiques, ainsi que les techniques d'estimation des paramètres et de choix de l'architecture,
- les chapitres 7 et 8 qui contiennent respectivement un type de modèle permettant la modélisation de séries stationnaires par morceaux et une méthodologie de recherche stochastique d'architecture, établie dans le but d'améliorer les étapes de la modélisation et en particulier celle de l'identification ;
- le chapitre 9 qui résume les diverses expériences et comparaisons des méthodes, appliquées à la série des consommations journalières d'électricité sur la France métropolitaine.

Les notations qui sont utilisées dans ce document peuvent sembler quelquefois confuses, tout d'abord pour les statisticiens abordant pour la première fois le « connexionisme », et inversement aux personnes ayant l'habitude de lire des documents traitant des réseaux de neurones et rencontrant des parties plus « statistiques ». Le vocabulaire employé pour désigner une même notion est en effet quelquefois différent d'un domaine à l'autre. Ainsi, pour un statisticien, le *vecteur des paramètres*[4] d'un modèle neuronal, se traduit par *l'ensemble des poids du réseau* pour le connexionniste. De même, les termes statistiques « *estimation du vecteur des paramètres* » et « *identification du modèle* » représentent pour le connexioniste « *apprentissage des valeurs des poids* » et « *choix de l'architecture*». Tout au long de cette thèse on tente de rester cohérent à la fois avec les notations et le vocabulaire employés. Cet exercice est cependant difficile, et certains lecteurs devront sans doute consentir à quelques efforts d'adaptation.

4. On peut aussi utiliser le terme *paramètre* au singulier pour désigner le *vecteur des paramètres*.

Voici un résumé du contenu de ce mémoire :

La modélisation paramétrique en général et non-linéaire en particulier.

La première partie de cette thèse est constituée tout d'abord d'un chapitre didactique, ayant pour but de décrire et d'analyser certains modèles paramétriques de prévision de séries temporelles parmi les plus utilisés. En particulier, on propose une classification des familles de modèles basée sur certaines propriétés statistiques. On effectue alors une analyse descriptive au sein de chaque famille de modèles afin de dégager les différents types de dynamiques qui les caractérisent. Le lecteur aura ainsi la possibilité de choisir le modèle, parmi ceux proposés, susceptible de rendre compte des données dont il dispose.

Dans le chapitre 3, on s'intéresse plus particulièrement aux propriétés statistiques des modèles autorégressifs fonctionnels, linéaires ou non, et à l'estimation du vecteur des paramètres associés. On rappelle en particulier les conditions assurant la stabilité de ce modèle, et les conditions permettant d'appliquer une loi forte des grands nombres pour les fonctions moments d'ordre supérieur à deux. Ceci offre alors un cadre statistique suffisamment riche pour permettre d'établir un résultat original : la consistance forte et la normalité asymptotique de l'estimateur des moindres carrés du vecteur des paramètres. On établit ensuite une loi du logarithme itéré pour cet estimateur, ce qui conduit à un résultat d'identification presque-sûre du vrai modèle dans le cadre de la sélection de modèles par contraste pénalisé.

Le perceptron multicouches, ses propriétés statistiques, ses techniques d'évaluation des poids et de choix de l'architecture

Dans cette section, après un bref rappel historique sur le perceptron multicouches, on tente de dégager les particularités de ce modèle paramétrique spécifique. En particulier, on reprend les conditions associés aux propriétés statistiques décrites dans le chapitre 3 pour les adapter au modèle neuronal. On tente aussi d'effectuer des comparaisons entre le modèle neuronal et les modèles linéaires classiques.

Les chapitres 5 et 6 sont consacrés respectivement à l'apprentissage du réseau (estimation des valeurs des poids synaptiques) et au choix de l'architecture. On propose en particulier au sein du chapitre 6 une méthode statistique d'élimination des poids synaptiques superflus (technique aussi appelée en anglais *pruning*). Cette technique appelée *SSM* pour « *Statistical Stepwise Method* », est basée sur un test statistique de nullité des paramètres, directement dérivé des résultats théoriques établis dans les chapitres précédents. Cette méthode inclut en particulier un critère rigoureux permettant de stopper l'élimination.

Modèles et applications spécifiques

On présente aussi dans le chapitre 7, une classe de modèle appelé *modèle neuronal modulaire* (MNM), introduit dans la communauté des réseaux de neurones par (Jacobs et al., 1991) sous le nom de « *mixture of experts* ». L'idée de base, inhérente aux modèles modulaires, est de diviser un problème complexe en un certain nombre de sous-problèmes plus simples et plus spécifiques. Dans l'approche que l'on a choisie, on utilise des réseaux du type perceptron multicouches pour émuler à la fois les sous-modèles et le modèle de partitionnement. Ce modèle s'avère être un outil puissant pour modéliser des séries temporelles stationnaires par morceaux.

Dans un tout autre registre, on propose chapitre 8 une technique de recherche stochastique d'architecture initiale. En effet, trouver une structure de départ adaptée facilite l'étape d'estimation des paramètres et l'étape d'identification du modèle. Dans ce but, on a choisi d'effectuer une recherche stochastique au sein d'un espace de modèle déterminé, mais assez large. Cette méthodologie est alors décrite, dans un contexte où les modèles sont des réseaux de neurones à propagation directe sans notion de couches, et où la recherche stochastique est basée sur des algorithmes génétiques.

Application à la série des consommations électriques journalières

Ce mémoire de thèse se conclut par la description de l'application des diverses techniques et méthodes développées dans cette thèse au problème concret de la prévision de la consommation électrique journalière sur la France métropolitaine. On dispose d'une longue chronique de données de consommations ainsi que de données explicatives. Cette caractérisque, ainsi que certaines corrélations non-linéaires entre variables, rend cette série précieuse à étudier et la comparaison des méthodes de prévision d'autant plus intéressante. Ceci fait l'objet du chapitre 9.

À la fin de ce mémoire, en annexe A, sont reproduits deux articles publiés dans des revues internationales : *IEEE Transactions on Neural Networks* Vol. 6 pages 1355-1364 et *International Journal of Neural Systems* Vol. 6 N°4 pages 368-396 et un article plus théorique soumis pour publication. Ces articles ainsi que les 8 communications et les trois rapports internes E.D.F. (voir annexe B) présentées dans des conférences internationales et citées dans la bibliographie sont tous issus des travaux de recherches repris dans ce document.

Chapitre 2

Modèles paramétriques et prévision de séries temporelles

Lorsqu'un statisticien, un économètre ou un praticien observe un phénomène chronologique quantifiable sous la forme d'une suite finie de réels $(X_t)_{t=1,2,\ldots,T}$, il peut tenter de *modéliser* ces données pour les analyser et en comprendre la structure sous-jacente associée. Il peut aussi vouloir prévoir l'évolution future du phénomène, en calculant les valeurs $(X_{T+k})_{k\geq 1}$. Il est alors confronté à un problème de *prévision de séries temporelles*.

A ce stade, il est nécessaire de poser un certain nombre d'hypothèses (HYPOTHÈSES **[H]**) :

- On suppose que les T *observations* $X_1, X_2, \ldots, X_T$ sont extraites d'une suite infinie aléatoire $(X_t)_{t\in\mathbb{Z}}$ aussi appelée *processus à temps discret* [1].
- On associe au processus $(X_t)_{t\in\mathbb{Z}}$, une suite de variables aléatoires indépendantes et identiquement distribuées $(\varepsilon_t)_{t\in\mathbb{Z}}$, appelées *bruit*, de moyenne 0 et de variance σ^2 finie, avec $\forall t, s \in \mathbb{Z}$, $s \leq t$, ε_t indépendant de X_s.
- On suppose que la variable aléatoire 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 (ε_t) peut être vue comme la somme des erreurs de relevés des données à disposition, et/ou des impondérables liés au phénomène. On considère ici que cette perturbation intervient de manière additive dans la formulation de X_t :

$$\forall t \in \mathbb{Z}, \quad X_t = f(X_{t-1}, X_{t-2}, \ldots, X_{t-p}, \varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}) + \varepsilon_t \tag{2.1}$$

1. On s'intéresse ici uniquement aux phénomènes à temps discret. La modélisation à temps continu nécessite des techniques très différentes de celles exposées dans ce document.

Cette équation définit alors un *modèle fonctionnel autorégressif 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 :

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

Dans la représentation 2.1, la fonction f est inconnue. Entreprendre une modélisation paramétrique consiste alors à approximer f par une fonction f_θ paramétrée par un vecteur de paramètres θ. On peut alors réécrire l'équation 2.1 en introduisant l'approximation paramétrique :

$$X_t = f_\theta(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) + \varepsilon_t + \eta_t \tag{2.2}$$

avec

$$\eta_t = f(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) - f_\theta(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) \quad .$$

Ici f_θ peut être de la forme d'un modèle linéaire du type AR, d'un modèle bilinéaire du type polynômial, ou d'un réseau de neurones basé sur des combinaisons de fonctions sigmoïdes (modèles dont nous donnons les définitions dans la suite), ou de toute autre famille de fonctions paramétrées. Le terme η_t correspond alors à l'erreur due à l'approximation de f par f_θ. Nous considèrerons par la suite qu'il existe un vecteur de paramètres θ_0 tel que l'erreur η_t soit nulle ou négligeable devant ε_t. Le phénomène se met ainsi en équation sous la forme :

$$X_t = f_{\theta_0}(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) + \varepsilon_t \quad . \tag{2.3}$$

Cette équation définit le « *vrai modèle* », et on appelle θ_0 la « *vraie valeur* » du paramètre. Effectuer une modélisation consiste alors à supposer que la structure du modèle est correcte et à tenter d'ajuster le vecteur des paramètres θ. Le modèle s'écrit finalement :

$$X_t = f_\theta(X_{t-1}^{(p)}, \varepsilon_{t-1}^{(q)}) + \varepsilon_t \quad , \tag{2.4}$$

avec θ à estimer.

Dans le cas où les retards du bruit n'interviennent pas, on est en présence d'un modèle *autorégressif fonctionnel*[2](ARF), modèle dont les propriétés probabibilistes (irréductibilité, stabilité, ergodicité géométrique, mélangeance) sont étudiées par de nombreux auteurs [voir par exemple (Doukhan et Ghindes, 1992; Robinson, 1977; Jones, 1978; Duflo, 1996)]. Ce modèle s'écrit :

$$X_t = f_\theta(X_{t-1}^{(p)}) + \varepsilon_t \tag{2.5}$$

Notons aussi que beaucoup de modèles nécessitent l'introduction de variables explicatives dites *exogènes*. On parle alors de *modèle fonctionnel autorégressif avec variables*

2. À partir du chapitre 2, on se limite à l'étude de modèles de ce type, en étudiant certaines propriétés de l'estimateur du vecteur des paramètres θ, et en considérant le cas très particulier où f est une fonction de la classe des *perceptrons multicouches* (chapitres 4 et suivants).

exogènes. Ainsi, si $\{Y^1, Y^2, \ldots, Y^k\}$ sont k variables exogènes corrélées au phénomène, le modèle avec variables explicatives s'écrit :

$$X_t = f_\theta(X_{t-1}^{(p)}, Y_t^1, Y_t^2, \ldots, Y_t^k, \varepsilon_{t-1}^{(q)}) + \varepsilon_t \quad . \tag{2.6}$$

Une fois le cadre général défini, il reste à résoudre un certain nombre de problèmes, tels que :

1) Choisir la structure générale du modèle paramétrique (déterminer la famille de fonction f_θ la plus adaptée aux données).

2) Identifier un modèle au sein de cette famille (choisir une structure définie et fixe au sein de la famille de modèles).

3) Estimer efficacement le vecteur des paramètres θ du modèle identifié (découvrir la valeur de θ_0).

4) Évaluer la fiabilité du modèle à l'aide de tests statistiques sur les paramètres et sur les prévisions.

Ces quatre tâches sont accomplies généralement de manière séquentielle, en s'appuyant sur des méthodes statistiques ou des heuristiques plus ou moins fiables, suivant la famille de modèles choisie et la quantité de données à disposition. On peut alors avancer que si ces quatre étapes sont correctement spécifiées et mises en oeuvre, la recherche aboutit, sauf cas pathologique, à un modèle possédant les mêmes propriétés statistiques que le processus $(X_t)_{t\in\mathbb{Z}}$ étudié, et reproduisant son comportement à court ou moyen terme[3].

En ce qui concerne le choix de la famille de modèles, on peut déjà tenter de discerner, suivant la nature des données, dans quelle classe de modèles on peut opérer ce choix.

Dans la première partie de ce chapitre, on tente donc de classer les différents types de modèles existants. Dans la deuxième partie, on donne une définition de quelques *familles* de modèles utilisés dans la pratique, et de leurs propriétés statistiques. On en fera aussi des analyses descriptives. Cette partie a pour but d'orienter la recherche du modèle susceptible de reproduire au mieux les trajectoires des processus à modéliser. On termine enfin, en énumérant un certain nombre de problèmes et de questions qui peuvent intervenir tout au long de la modélisation.

3. Si on observe une série de T *réalisations* $(X_1, X_2, \ldots, X_T)$ et que l'on tente de prévoir les valeurs futures de la série X_{T+h}, $h \in \mathbf{N}^*$ à l'aide de la modélisation (équation 2.4, avec f et θ fixés), la difficulté d'obtenir une bonne approximation du comportement du processus s'accroît généralement avec la valeur de h appelée *horizon de prévision*. On verra d'ailleurs que la prévision à un horizon supérieur à un ne s'obtient pas de manière aussi directe dans le cas non-linéaire que dans le cas linéaire.

2.1 Classification des modèles paramétriques

On peut trouver les définitions et descriptions de différentes classes de modèles paramétriques dans les livres très complets de (Meyn et Tweedie, 1993) et (Guégan, 1994) par exemple. L'ensemble de ce chapitre s'en inspire d'ailleurs dans une large mesure. Plusieurs grandes classes de modèles paramétriques se dégagent, possédant des propriétés statistiques différentes. Elles peuvent être définies de manière duale. On distingue :

1- Les modèles
  - stochastiques
    - stationnaires / non-stationnaires,
    - gaussiens / non-gaussiens,
  - déterministes.

2- Les modèles dont la structure sous-jacente est
  - linéaire par rapport aux paramètres,
  - non linéaire par rapport aux paramètres.

On pourra ainsi classer un modèle suivant son appartenance à l'une des sous-parties de chacune des classes 1 et 2. Les propriétés statistiques de normalité et de stationnarité forment deux grands ensembles au sein des modèles stochastiques. D'autres propriétés telles que l'ergodicité, la stabilité ou la mélangeance n'apparaissent pas dans cette classification. Elles sont néanmoins essentielles, comme nous le verrons pour les modèles spécifiques ARF dans le chapitre 3, pour une étude statistique appronfondie.

2.1.1 Les modèles déterministes/stochastiques

On entend par *déterministe* un modèle dénué d'aléas, pour lequel la valeur du processus X au temps t s'obtient uniquement en fonction des valeurs passées. Inversement, un modèle *stochastique* inclut du bruit et donc des aléas dont il est impossible de connaître les valeurs précises par avance.

Les modèles déterministes

Un modèle déterministe est par exemple le modèle relatif à l'équation 2.5, avec p fini, et $\varepsilon_t = 0$ quel que soit t. Notons que si la fonction f_θ est linéaire en θ, l'étude des trajectoires des modèles de ce type est restreinte à trois type de comportements : le processus associé tend géométriquement vers l'infini, décroît géométriquement vers zéro, ou est purement cyclique (somme d'un nombre fini de composantes périodiques). Ces modèles n'ont donc pas des types de trajectoires aussi variées que celles des modèles linéaires stochastiques et ne sont pas d'un grand intérêt.

Dans le cas où le modèle est non-linéaire, les comportements des processus associés peuvent être beaucoup plus complexes, voire proches de comportements stochastiques. Par exemple, les suites de nombres dits aléatoires générés sur ordinateur proviennent généralement de processus déterministes non-linéaires au comportement très erratique, que l'on garde artificiellement, via des fonctions *modulos*, dans un intervalle choisi.

Un sous-ensemble intéressant de modèles déterministes est constitué des modèles dits *chaotiques*, dont le comportement dépend fortement des conditions initiales et donc des valeurs passées du processus. Une perturbation infime de ces conditions amène à des trajectoires de réalisations totalement différentes. D'où la difficulté de prévoir les processus chaotiques, avec des modélisations paramétriques puisque la plus petite erreur d'estimation des paramètres entraîne des prévisions complètement erronées, en particulier pour un horizon de prévision supérieur à un. Plusieurs travaux récents sur la prévision de tels processus sont apparus dans les années 80. Citons ceux de l'approche par espace d'états (Farber, 1987; Farmer et Sidorowich, 1988; Casdagli, 1989), ceux utilisant des modèles linéaires par morceaux (Farber, 1982), les modèles neuronaux (perceptrons multicouches) (Lapedes et Farber, 1987), les modèles à base de fonctions radiales (Casdagli, 1989).

Les séries chaotiques déterministes sont définies sur un intervalle fini, et ne peuvent pas être nécessairement étendues sur un espace infini. On peut distinguer plusieurs sortes de mouvements chaotiques :

– les mouvements chaotiques transients qui, par croisement d'oscillations périodiques, se caractérisent par des trajectoires erratiques, très sensibles aux conditions initiales,

– les mouvements chaotiques intermittents, caractérisés par des périodes d'oscillations régulières entrecoupées d'explosions de chaos.

Citons par exemple le célèbre processus quadratique (ou logistique) (voir figure 2.1) dont le comportement s'apparente aux processus relatifs à la première définition :

$$\text{pour } \lambda > 3.57, \quad t \in \mathbf{N}, \quad X_0 \in]0,1[\quad X_t = \lambda X_{t-1}(1 - X_{t-1}) \quad . \tag{2.7}$$

La distribution marginale bivariée de ce processus est représentée figure 2.2. Ce processus devient chaotique par mélange de deux oscillations dès que $\lambda > 3.57$. Une étude approfondie de ce processus pourra être trouvée dans (Guégan, 1994).

Si les séries chaotiques suscitent un intérêt croissant dans la communauté scientifique, leur application aux phénomènes réels reste marginale. Dans toute la suite de ce document, on ne s'intéresse qu'aux modèles stochastiques.

Les modèles stochastiques

On entend par modèles stochastiques des modèles comprenant des variables aléatoires. Le modèle défini équation 2.4 est par exemple un modèle stochastique, incluant

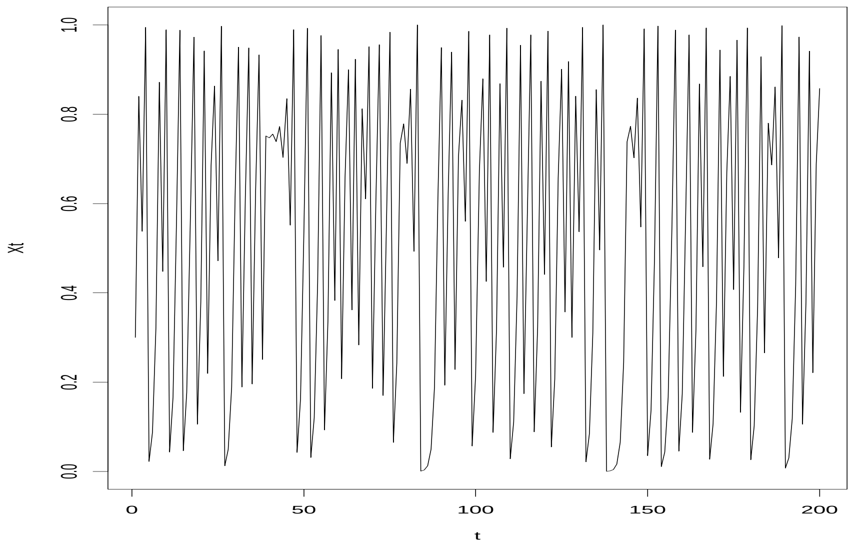


FIG. 2.1 – *200 points de la série quadratique dite « logistique », pour* $\lambda = 4$.

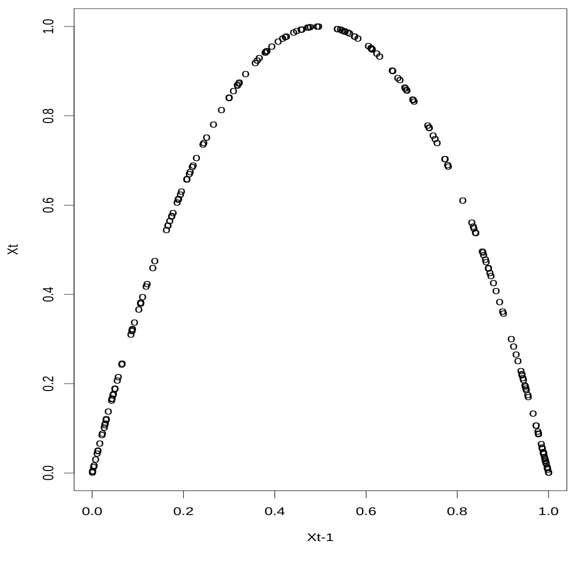


FIG. 2.2 – *Distribution marginale bivariée de la série logistique pour* $\lambda = 4$. *On remarque sa très grande simplicité et régularité, alors que la série engendrée possède un comportement erratique.*

un bruit de type additif qui vient s'ajouter au terme autorégressif déterministe. On peut alors définir une densité de probabilité associée à la variable aléatoire X, et utiliser la batterie usuelle d'outils statistiques.

Deux sous-classes peuvent être définies.

Les modèles stationnaires/ non-stationnaires

La notion de *stationnarité* correspond à celle de stabilité en loi au cours du temps. L'idée sous-jacente est que deux suites extraites d'une série stationnaire, provenant de deux intervalles de temps quelconques mais de même taille, possède des propriétés statistiques identiques (même distribution, et donc même moyenne, même variance, etc).

Définition 2.1 :

> *Soit $(X_t)_{t\in\mathbb{Z}}$ un processus possédant des moments d'ordre 2. Ce processus est stationnaire au sens strict si, pour tout $(r,s)\in\mathbb{Z}^2$ et pour tout $h>0$, $(X_\tau)_{r\leq\tau\leq r+h}$ et $(X_\tau)_{s\leq\tau\leq s+h}$ ont même loi.*
>
> *Il est dit* stationnaire au second ordre *si sa moyenne m est constante et si, pour tout $(r,s)\in\mathbb{Z}^2$, $r<s$, la covariance $cov(X_{t+r},X_{t+s}) = E(X_{t+r}X_{t+s}) - E(X_{t+r})E(X_{t+s})$ ne dépend que de $k=s-r$ quelle que soit la valeur de t.*
>
> *On note alors $\gamma(k)$ cette covariance, et la fonction $k\mapsto\gamma(k)/\gamma(0)$ est appelé* fonction d'autocorrélation.

Cette propriété devient essentielle dès lors que l'on veut utiliser des outils statistiques. En particulier, tous les processus divergents ou incluant des *tendances* ne satisfont pas aux conditions de cette définition. Une importante sous-classe de processus non-stationnaires est constituée des séries *stationnaires par morceaux* possédant des comportements relatifs à des changement de régimes. Pour ce type de série, la stationnarité est vérifiée sur certains intervalles associés à un régime particulier, mais pas sur la globalité de la série. Un exemple de modèle de ce type est présenté chapitre 7. On y détaille en particulier le calcul de la vraisemblance (voir 7.2.1).

Les modèles gaussiens/ non-gaussiens

Reprenons le modèle défini équation 2.4, et supposons que le bruit associé $(\varepsilon_t)_{t\in\mathbb{Z}}$ suive une loi gaussienne.

$$X_t = f_\theta(X_{t-1},X_{t-2},\ldots,X_{t-p})+\varepsilon_t, \qquad \text{avec} \qquad \varepsilon_t\sim\mathcal{N}(0,\sigma^2)$$

Dans ce cas, la variable X_t, sachant $X_{t-1},X_{t-2},\ldots,X_{t-p}$ est aussi une variable aléatoire, de loi gaussienne d'espérance $f_\theta(X_{t-1},X_{t-2},\ldots,X_{t-p})$ et de variance σ^2. L'estimation du vecteur de paramètre θ s'opère alors de manière classique par la méthode du maximum de vraisemblance. L'hypothèse gaussienne permet aussi d'utiliser des tests et des théorèmes bien connus.

Il existe néanmoins bon nombre de phénomènes dont le bruit associé n'est pas gaussien. Dans ce cas, les recherches restent encore embryonnaires, et les applications peu développées. Citons par exemple des phénomènes dont le bruit associé possède une loi du type *mélange de lois gaussiennes*, du type *loi de Poisson* ou du type χ^2. Dans ce cas, la théorie du maximum de vraisemblance s'applique, mais la fonction de vraisemblance est différente de celle retenue dans l'hypothèse gaussienne et les formules de dérivation par rapport aux paramètres sont quelquefois difficiles à obtenir, en particulier lorsque la loi n'appartient pas à la famille des lois exponentielles.

Voir (Lii et Rosenblatt, 1982) pour une étude de la structure de probabilité de tels processus, et (Breidt et al., 1990) pour une étude de leurs vraisemblances approchées.

2.1.2 Les modèles linéaires/non-linéaires

On distingue les modèles linéaires *par rapport aux paramètres* des modèles linéaires *par rapport aux variables*. Ainsi le processus $(Y_t)_{t\in\mathbb{Z}}$ défini par

$$Y_t = \alpha \ln(Y_{t-1}) + \varepsilon_t, \quad \alpha \in \mathbb{R} \quad ,$$

est un modèle non-linéaire par rapport à la variable Y mais linéaire par rapport au paramètre α. La méthode d'estimation linéaire standard ainsi que tous les tests appliqués aux paramètres dans un cadre linéaire restent dans ce cas valides. Cette classe de modèles offre néanmoins des possibilités limitées en terme d'identification et n'est utilisé dans la pratique que lorsque ce type de non-linéarité est flagrante.

Par la suite, les termes *modèle linéaire* et *non-linéaire* sont toujours associés de manière implicite à « *par rapport aux paramètres* ».

Les modèles linéaires

Le modèle est dit *linéaire*, dès lors que la fonction paramétrée f_θ est linéaire en θ. Si on reprend l'équation 2.4 avec une telle fonction, X_t s'exprime alors sous la forme d'une combinaison linéaire des retards du processus et du bruit :

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

avec $a_i \in \mathbb{R}, \quad \forall i \in \{1, 2, \ldots, p\}, b_j \in \mathbb{R}, \quad \forall j \in \{1, 2, \ldots, q\}$.

Ces modèles, appelés modèles *ARMA* pour « *AutoRegressive Moving Average* » peuvent modéliser une large classe de processus stationnaires, et sont amplement utilisés dans la pratique. Leurs propriétés statistiques ont été largement étudiées et le choix du modèle le plus performant est maintenant bien maîtrisé, tant au niveau de l'identification que de l'estimation des paramètres. Voir la section 2.2.1 pour plus de détails.

On peut néanmoins souligner les limites des modèles linéaires qui ne sont plus utilisables dès lors que les relations entre les variables à différents pas de temps sont de nature non-linéaire.

Les modèles non-linéaires

On entend par modèle non-linéaire par rapport aux paramètres, l'ensemble des modèles dont la description ne correspond pas à celle du modèle linéaire. Cette classe contient donc des modèles de structure variée, possédant des propriétés statistiques et des qualités d'approximation diverses. D'où la nécessité de définir des sous-classes de modèles que l'on nomme *famille*, aux propriétés statistiques communes et aux structures explicitement définies afin de faciliter l'étape de l'identification. Ces structures se composent généralement de combinaisons de fonctions particulières, non-linéaires par rapport aux paramètres, appliquées aux retards du processus et du bruit associé.

Les types de comportement pris en compte par une modélisation non-linéaire peuvent être de nature fondamentalement différente. On peut observer ainsi des non-linéarités du type discontinuité dues à des changements brusques d'évolution (processus stationnaires par morceaux), des non-linéarités quadratiques, exponentielles ou autres. Ce que l'on cherche néanmoins le plus souvent est une famille de modèles, possédant des qualités d'approximation universelle, permettant de modéliser une large classe de processus (par exemple les processus associés à l'équation 2.1 avec f continue sur un compact).

On dispose actuellement de différents critères permettant de s'orienter vers des modèles linéaires ou non-linéaires. L'un de ces critères est l'analyse des représentations graphiques des réalisations :

- L'étude des histogrammes permet par exemple de déceler des densités de probabilités bimodales ou l'existence d'attracteurs typiques dans le cas non-linéaire.
- L'étude des distributions bivariées du processus $(X_t)_{t\in\mathbb{Z}}$ qui sont obtenues en représentant pour tout t, X_t en fonction de $X_{t-\tau}$, $\tau > 0$, permettent de mettre en évidence des formes spécifiques caractérisant des processus non-linéaires ou non-gaussiens. On sait en effet que dans le cas gaussien linéaire, on obtient une forme ellipsoïdale régulière. Toute forme s'en écartant est donc non-gaussienne et/ou non-linéaire.

Il s'avère cependant qu'en cas de doute, le mieux est encore d'effectuer plusieurs modélisations du type linéaire et non-linéaire, et de choisir celle qui semble la mieux adaptée au problème. Ceci sous-entend néanmoins que l'on dispose d'outils statistiques fiables pour évaluer les performances des modèles.

2.2 Famille de modèles, définitions et descriptions

Dans toute la suite, on se place dans le cadre des hypothèses **[H]**, et on considère des modèles correspondant à l'équation 2.4 avec une fonction f de forme définie et un vecteur de paramètres θ associé à cette structure. On parle alors de *famille* de modèles. On en donne ici un bref aperçu, en se limitant aux définitions et à la description de certaines de leurs principales propriétés statistiques. A noter que tous ces modèles peuvent inclure un traitement utilisant des variables explicatives exogènes tel que ceux décrits équation 2.6. On en fait référence uniquement pour les modèles relativement récents basés sur les réseaux de neurones.

Dans le but d'aller au-delà des équations, on illustre aussi l'ensemble des modèles définis par des graphiques montrant des comportements typiques de processus que l'on peut trouver au sein de chaque famille.

2.2.1 Les modèles ARMA et dérivés

Les modèles ARMA sont des modèles linéaires largement utilisés dans la pratique. Cette classe de modèles permet d'atteindre, à l'aide d'un nombre de paramètres relativement limité, une gamme de modélisations très variée. Ils furent étudiés et diffusés dans les années 70 par (Box et Jenkins, 1976) qui décrivent des techniques efficaces d'estimation du vecteur des paramètres et d'identification de ce modèle. Pour des études plus récentes et complètes sur les propriétés statistiques de tels modèles, on pourra lire par exemple le livre de (Brockwell et Davis, 1991).

On donne tout d'abord la définition du modèle autorégressif pur :

Définition 2.2 :

Soit $p \in \mathbb{N}^*$. On appelle modèle autorégressif ($\mathrm{AR}(p)$) un modèle de la forme :

$$X_t = a_0 + \sum_{i=1}^{p} a_i X_{t-i} + \varepsilon_t$$

avec $a_i \in \mathbb{R}, \quad \forall i \in \{0, 1, \ldots, p\}$.

L'avantage d'introduire des retards du bruit est, dans certains cas, de diminuer sensiblement le nombre de paramètres (les coefficients) à employer.

Définition 2.3 :

On appelle modèle « AutoRegressive Moving Average » (ARMA(p,q)) un modèle de la forme :

$$\text{pour} \quad p \geq 1, \quad q \geq 1, \quad X_t = a_0 + \sum_{i=1}^{p} a_i X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t$$

avec $a_i \in \mathbb{R}, \quad \forall i \in \{0,1,\ldots,p\}, b_j \in \mathbb{R}, \quad \forall j \in \{1,2,\ldots,q\}$.

La stabilité de tels modèles est étudiée par (Akaike, 1974). On pourra en retrouver la démonstration dans le livre de (Duflo, 1996). Pour ce qui est des propriétés de mélange et d'ergodicité, on pourra en trouver une étude détaillée dans (Mokkadem, 1987b). L'estimation s'opère en général à l'aide d'une méthode classique d'estimation des moindres carrés, ou en résolvant le système d'équations du type Yule-Walker (voir des livres d'initiation comme (Gourieroux et Monfort, 1990) pour plus de détails). On se limite ici à donner les conditions assurant la propriété essentielle de stationnarité.

Proposition 2.4 :

Soient $\psi(z)$ et $\theta(z)$ les polynômes relatifs au modèle ARMA de la définition 2.3 définis par

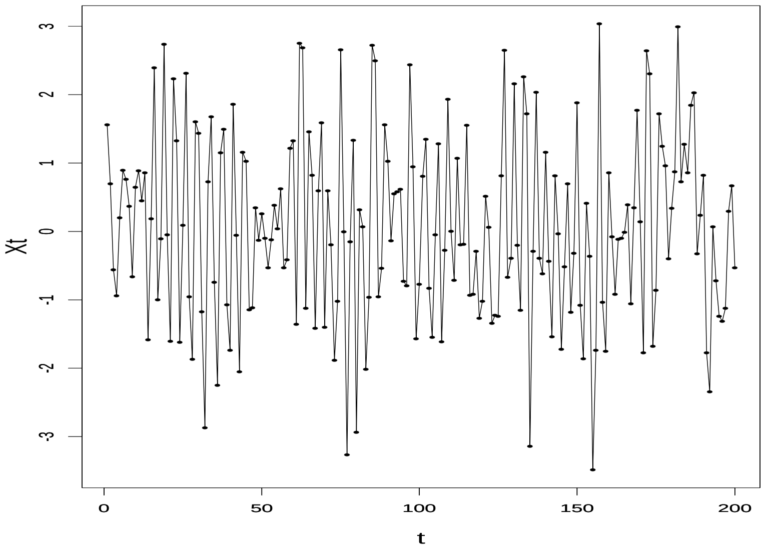
$$\begin{aligned}\psi(z) &= 1 - a_1 z - a_2 z^2 - \ldots - a_p z^p \\ \theta(z) &= 1 - b_1 z - b_2 z^2 - \ldots - b_q z^q\end{aligned}$$

alors, le modèle est stationnaire ssi :

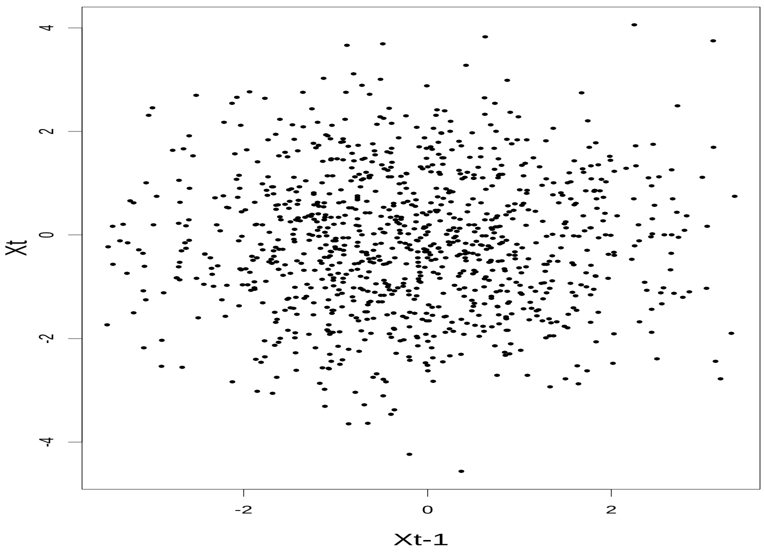
$$\left\{ \begin{array}{l} \text{– Les polynômes } \psi(z) \text{ et } \theta(z) \text{ n'ont pas de racines communes.} \\ \text{– Le polynôme } \psi(z) \text{ a toutes ses racines contenues dans le cercle unité.} \end{array} \right. \tag{2.8}$$

On peut voir figure 2.3 un comportement typique de série engendrée par un autorégressif d'ordre 2. On remarque la distribution bivariée des données en forme d'ellipse, forme classique dans le cas de processus linéaires.

$$(a) \qquad \begin{cases} X_t &= 0.1\ X_{t-1} - 0.4\ X_{t-2} - 0.1 + \varepsilon_t \\ \varepsilon_t &\sim \mathcal{N}(0,1) \end{cases}$$



(b)



(c)

FIG. 2.3 – *(a) Formule spécifique d'un modèle AR (définition 2.2). (b) 200 points de la série relative au modèle (a). (c) Représentation bivariée de la série relative au modèle (a).*

2.2.2 Les modèles TAR

Les modèles appelés *Threshold AutoRegressive* (ou modèles TAR) sont des modèles linéaires par morceaux, conçus dans le but de modéliser des processus stationnaires liés à des phénomènes soumis à différents régimes. Les modèles TAR restent populaires en économie et économétrie. La technique utilisée est d'introduire un brusque changement de modèles en s'appuyant sur les variables. Chaque modèle se spécialise dans un hyperplan. Des contraintes de continuités sont introduites à la main. Les partitions sont souvent relatives à une variable exogène telle que la volatilité pour des séries ayant trait à des indices boursiers (Engle, 1982; Bollerslev, 1986; Bollerslev et al., 1990). Un modèle plus flexible, de la même veine (nommé MARS, « multivariate adaptative regression splines ») (Friedman, 1991) a été aussi appliqué avec succès dans la prévision de séries financières (Lewis et al., 1994). Cette approche est utilisée dans de nombreux domaines aussi variés que l'hydrologie, la démographie, les systèmes radio, etc. On peut se reporter à (Tong, 1990) pour une présentation détaillée du modèle et de ses propriétés statistiques.

Définition 2.5 :

Soient $(r_1, \ldots, r_\ell)$, ℓ *réels et* $(R_1, \ldots, R_{\ell+1})$ *une partition de* $\mathbb{R}$ *tels que*

$$\begin{cases} R_1 &= \;]-\infty, r_1] \\ R_i &= \;]r_{i-1}, r_i], \; \forall i = 2, 3, \ldots, \ell \\ R_{\ell+1} &= \;]r_\ell, +\infty[. \end{cases}$$

Soit $p \in \mathbb{N}$, $p \geq 1$ *et* $a_i^{(j)} \in \mathbb{R} \quad \forall i \in \{0, 1, \ldots, p\}, \forall j \in \{1, 2, \ldots, \ell+1\}$. *On appelle modèle* TAR *un modèle de la forme :*

$$X_t = a_0^{(j)} + \sum_{i=1}^{p} a_i^{(j)} X_{t-i} + \varepsilon_t^{(j)} \quad si \; X_{t-1} \in R_j, \; \forall j = 1, 2, \ldots, \ell+1 \quad .$$

On impose aussi que $\varepsilon_t^{(j)}$ *soit indépendant de* $\varepsilon_t^{(j')}$ *pour* $j \neq j'$.

L'ergodicité et la stationnarité de tels processus ont été étudiées par (Chan et Tong, 1986). On pourra trouver une étude sur la mélangeance et l'egodicité géométrique dans (Diebolt et Guégan, 1991). On peut trouver une étude d'un cas non-gaussien, avec un bruit ayant une densité de probabilité de Cauchy dans (Andel et Barton, 1986).

Pour l'estimation des paramètres, les tentatives pour prévoir les seuils des modèles ne sont pas convainquants. Dans la pratique d'ailleurs, la valeur du seuil est estimée souvent suivant les connaissances que l'on possède du phénomène. Citons tout de même les efforts de (Chatfield, 1989) qui tente de résoudre ce problème pour une classe limitée de modèle à seuils.

En ce qui concerne l'identification de ce type de modèle, (Tong et Lim, 1980) proposent une procédure basée sur un critère d'Akaike adapté. Citons enfin l'approche bayésienne developpé par (Kheradmania et Tong, 1990) qui tente, dans le même temps, de résoudre le problème de l'estimation des seuils.

On peut voir figure 2.4 un comportement typique de série engendrée par un modèle TAR d'ordre 2. Conformément à la forme du modèle, la distribution bivariée est composée de deux nuages de points en forme d'ellipses de différents centres et de différentes largeurs.

2.2.3 Les modèles ARCH

Historiquement, les modèles ARCH ont été introduits par (Engle, 1982). Certains bruits associés à des phénomènes économiques et financiers ont en effet la particularité d'avoir des variances dont la valeur dépend du temps. Il a donc semblé intéressant d'inventer des modèles incluant cette spécificité, modèles que l'on qualifie d'*hétéroscédastiques*. Les modèles ARCH sont sans doute les plus connus de cette famille de modèles.

Définition 2.6 :

Pour un $p \geq 1$, on appelle modèle ARCH *un modèle de la forme :*

$$X_t = \varepsilon_t \times h^{1/2} \tag{2.9}$$

où

$$h_t = a_0 + a_1 X_{t-1}^2 + a_2 X_{t-2}^2 + \ldots + a_p X_{t-p}^2$$

avec $a_i \in \mathbb{R}, \quad \forall i \in \{0, 1, \ldots, p\}$

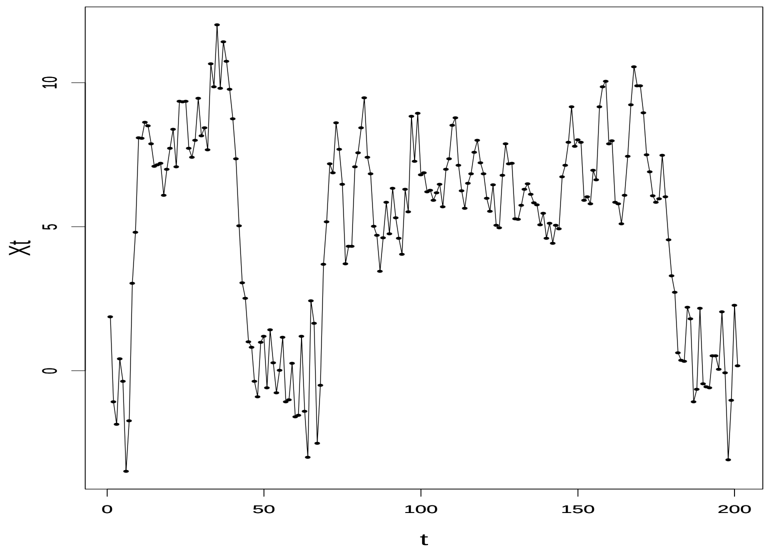
Leur étude probabiliste et statistique a été menée par de nombreux auteurs, voir en particulier (Engle, 1982) et (Guégan et Diebolt, 1994). Ils sont non-stationnaires puisque la variance du bruit associé au processus évolue au cours du temps.

Un des problèmes importants, qui est apparu avec la formalisation de ces modèles, est la construction d'un test permettant de distinguer un processus *hétéroscédastique*[4] d'un processus *homoscédastique*[5]. De nombreux tests ont été proposés. On peut en trouver quelques uns dans (Breush et Pagan, 1978) ou (Tsay, 1987).

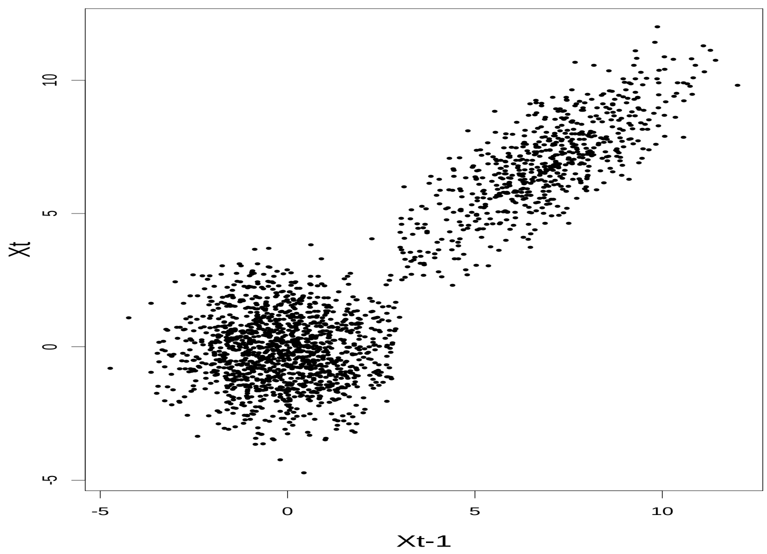
On peut voir figure 2.5 un comportement typique de série engendrée par un modèle ARCH d'ordre 2. On remarque les nettes fluctuations de variance au cours de son évolution, ce qui se traduit par une distribution bivariée quelquefois très dispersée.

4. La variance du bruit associé au processus dépend du temps.
5. La variance du bruit associé au processus est constante.

$$(a) \qquad \begin{cases} X_t &= 0.2\ X_{t-1} - 0.4\ X_{t-2} - 0.1 + \varepsilon_t^{(1)} \quad \text{si } X_{t-1} < 3 \\ X_t &= 0.8\ X_{t-1} - 0.3\ X_{t-2} + 2 + \varepsilon_t^{(2)} \quad \text{sinon} \\ \varepsilon_t^{(1)} &\sim \mathcal{N}(0,1)\ ;\ \varepsilon_t^{(2)} \sim \mathcal{N}(0,1)\ ; \quad \varepsilon_t^{(1)} \perp \varepsilon_t^{(2)} \end{cases}$$



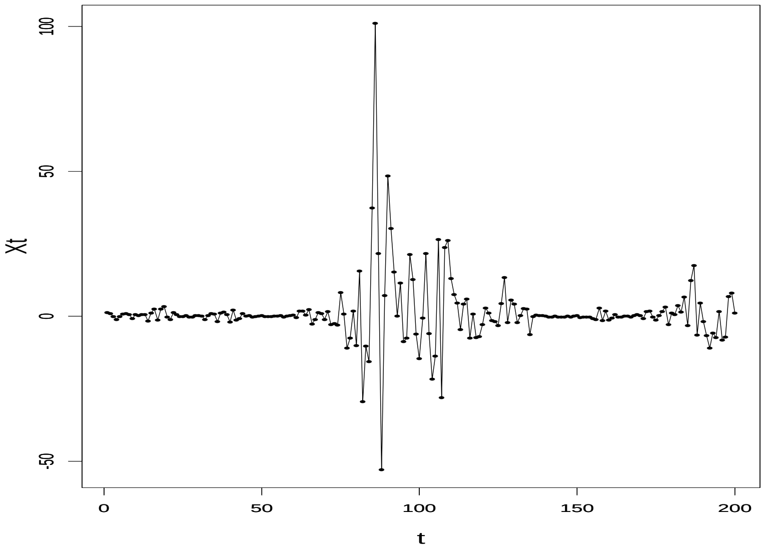
(b)



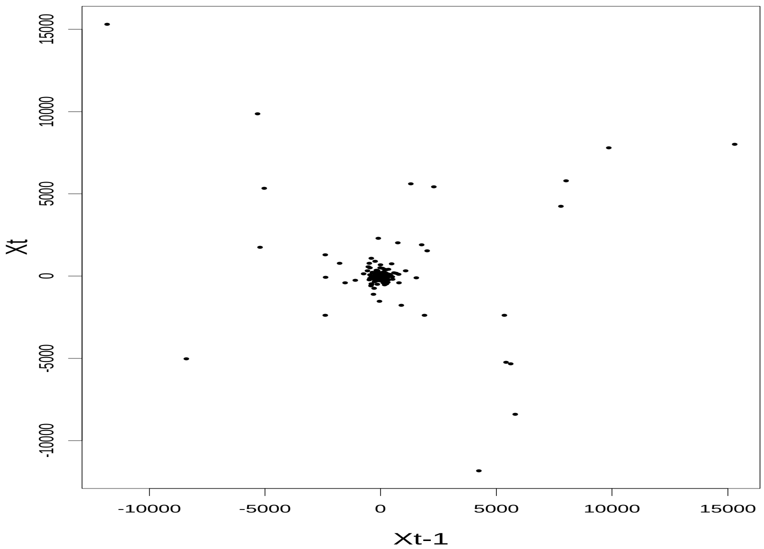
(c)

FIG. 2.4 – *(a) Formule spécifique d'un modèle TAR (définition 2.5). (b) 200 points de la série relative au modèle (a). (c) Représentation bivariée de la série relative au modèle (a).*

$$(a) \qquad \begin{cases} X_t &= \sqrt{0.9\ X_{t-1}^2 + 0.7\ X_{t-2}^2 + 0.05} \times \varepsilon_t \\ \varepsilon_t &\sim \mathcal{N}(0,1) \end{cases}$$



(b)



(c)

FIG. 2.5 – *(a) Formule spécifique d'un modèle ARCH (définition 2.6). (b) 200 points de la série relative au modèle (a). (c) Représentation bivariée de la série relative au modèle (a).*

2.2.4 Les modèles Bilinéaires

Les modèles bilinéaires sont une extension polynômiale naturelle des modèles ARMA. Ses modèles se caractérisent par un comportement erratique avec un enchaînement de plages de fortes perturbations et de plages significativement plus calmes. Ce comportement a amené les physiciens ainsi que des météorologues à s'intéresser à ce type de processus. À l'origine, ils ont été étudiés d'un point de vue déterministe puis repris par (Granger et Andersen, 1978) pour modéliser des séries économiques.

Définition 2.7 :

On appelle modèle bilinéaire *un modèle de la forme :*

$$X_t = a_0 + \sum_{i=1}^{p} a_i X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \sum_{i=1}^{p} \sum_{j=1}^{q} c_{ij} X_{t-i}\, \varepsilon_{t-j} + \varepsilon_t$$

avec $a_i \in \mathbb{R}, \quad \forall i \in \{1, 2, \ldots, p\}$

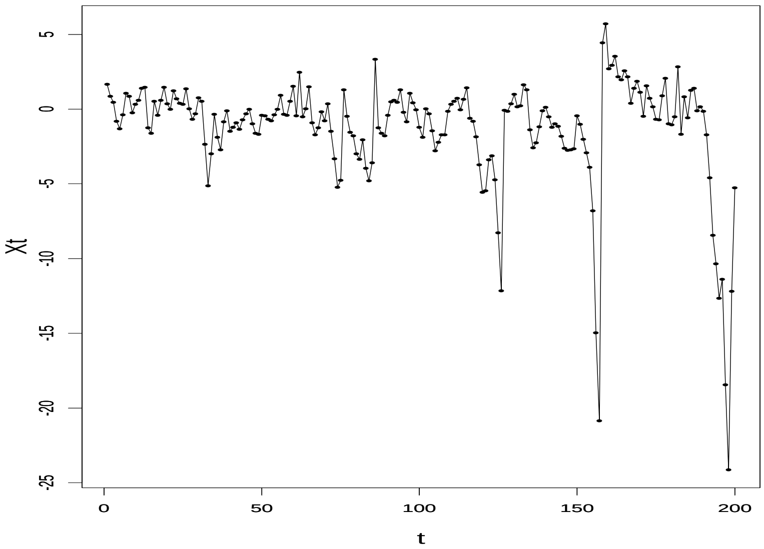
L'existence et l'unicité d'une solution stationnaire est donné par (Liu, 1988). On peut trouver dans (Pham, 1986) une étude sur la mélangeance et l'ergodicité géométrique de tels modèles.

On peut voir figure 2.6 l'évolution typique d'une série engendrée par un modèle bilinéaire d'ordre 2. On remarque les brusques changement de comportements et la forme très particulière de sa distribution bivariée qui peut prendre la forme d'un papillon.

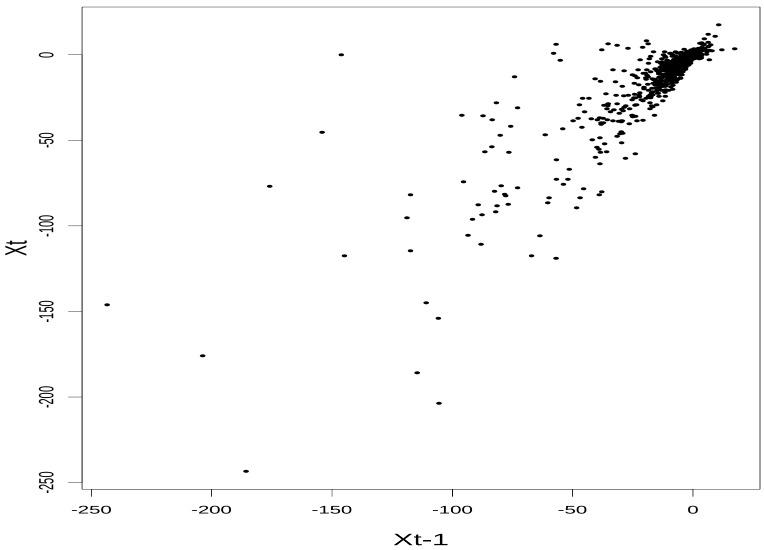
2.2.5 Les modèles neuronaux

Sous l'appelation « réseaux de neurones » se cachent un grand nombre de modèles, de structures variées, utilisés dans différents domaines. En ce qui concerne la prévision de séries temporelles, l'un des modèles les plus employés est basé sur le modèle du perceptron simple défini par (Rosenblatt, 1962). Ce modèle, ainsi que le modèle dérivé le plus connu, le *perceptron multicouches* sont décrits, ainsi que leurs propriétés, dans le chapitre 4. Ces modèles constituent une extension naturelle dans le domaine non-linéaire des modèles autorégressifs linéaires. On distingue ici les modèles à propagation directe qui constituent des modèles entrée-sortie sans boucles et les modèles récurrents qui possèdent des boucles de la sortie vers l'entrée ou au sein même de sa structure. Ces derniers ne seront pas étudiés ici. La mise en équation des modèles neuronaux peut sembler complexe. Les réseaux de neurones ont cependant l'avantage de pouvoir 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. Néanmoins, afin de rester dans le ton de cette section, nous en restons à l'écriture

$$(a) \qquad \begin{cases} X_t &= 0.2\ X_{t-1} + 0.8\ X_{t-2} \times \varepsilon_{t-1} + \varepsilon_t \\ \varepsilon_t &\sim \mathcal{N}(0,1) \end{cases}$$



(b)



(c)

FIG. 2.6 – *(a) Formule spécifique d'un modèle bilinéaire (définition 2.7). (b) 200 points de la série relative au modèle (a). (c) Représentation bivariée de la série relative au modèle (a).*

mathématique habituelle[6].

A l'instar du modèle autorégressif linéaire AR(p) (voir la définition 2.2), on considère que dans un modèle à propagation directe, X_t est fonction uniquement des retards $(X_{t-1}, X_{t-2}, \ldots, X_{t-p})$. Le modèle est alors constitué d'un réseau de filtres non-linéaires ordonnés suivant une certaine logique et appliqués à une combinaison linéaire des retards. Ces filtres appelés « fonctions d'activation » ou « fonctions de transfert » peuvent être de plusieurs natures.

Les perceptrons multicouches

Historiquement (voir chapitre 4) l'un des premiers modèles à propagation directe est le « perceptron multicouches », dont le principe a été étudié par (Rosenblatt, 1962) qui a mis en évidence ses possibilités d'apprentissage. Les propriétés statistiques de ce modèle, et en particulier sa propriété d'approximation universelle, sont décrites en détail section 4.2. On donne ici la définition formelle d'un *perceptron multicouches possédant une seule couche cachée*[7].

Définition 2.8 :

Soit $(p, n) \in \mathbb{N}^2$ *non-nuls. On appelle modèle* neuronal *à une couche cachée, un modèle de la forme :*

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

où

- n *est le nombre de neurones de la couche cachée.*
- $\theta = \{(\alpha_j)_{0 \leq i \leq n}, (\beta_{ij})_{0 \leq i \leq p, 1 \leq j \leq n}\} \in \mathbb{R}^{n \times (p+2)+1}$ *est le vecteur des paramètres, l'ensemble des* poids, *ou encore* connexions *en langage connexioniste.*
- ψ *est une fonction de* $\mathbb{R}$ *dans* $\mathbb{R}$, *généralement non linéaire, ni même polynômiale.*

La fonction d'activation la plus souvent utilisé est la fonction logistique :

$$x \mapsto \frac{1}{1 + e^{-x}}$$

6. Nous donnons la traduction graphique des modèles neuronaux que nous étudions plus en détail dans la suite : les perceptrons multicouches et les réseaux à propagation directe sans notion de couches, respectivement dans le chapitre 4 et dans le chapitre 8.

7. Comme nous le verrons chapitre 4, l'ensemble des transformations comprises entre l'entrée et la sortie peuvent être représentées sous la forme de *couches* dites *cachées*

ou la fonction tangente hyperbolique

$$x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad .$$

Nous y reviendrons plus longuement chapitre 4.

On peut voir figure 2.7 l'évolution d'une série engendrée par un modèle du type perceptron multicouches muni de fonction d'activation tangente hyperbolique et comportant deux entrées. En leur qualité d'approximateur universel[8], les perceptrons multicouches peuvent virtuellement reproduire n'importe quel type de comportement. On a choisi ici de reproduire un comportement du type TAR (figure 2.4). On remarque néanmoins que la transition entre les deux nuages de points (les deux régimes de la série) peut être beaucoup moins brutale que celle relative à un modèle TAR.

Les modèles « fonctions radiales de base »

D'abord utilisés pour résoudre des problèmes de classification, les modèles « *radial basis function* » (*RBF*) ont fait leur apparition à la fin des années 80 (Moody et Darken, 1988; Moody et Darken, 1989) et ont été appliqués peu de temps après au problème de prévision de séries temporelles (Casdagli, 1989). L'idée ici est d'utiliser des fonctions filtres gaussiennes. Une telle modélisation s'avère intéressante dans le cas d'individus non-linéairement séparables. Elle a aussi été employée avec succès pour prédire la série non-linéaire chaotique décrit équation 2.7 ((Casdagli, 1991)). Un bon résumé des capacités et des propriétés de ce type de modèle est donné par (Powell, 1987). On se limite ici à en donner la définition. Dans la suite, nous n'étudions pas ce type de modèle neuronal.

Définition 2.9 :

*On appelle modèle « radial basis function » (*RBF*) un modèle neuronal à une couche cachée constitué d'une combinaison linéaire de fonctions gaussiennes :*

$$X_t = \sum_{i=1}^{n} w_i \exp(- \parallel X_t^{(p)} - C_i \parallel^2 / (2\sigma_i^2)) + \varepsilon_t, \tag{2.10}$$

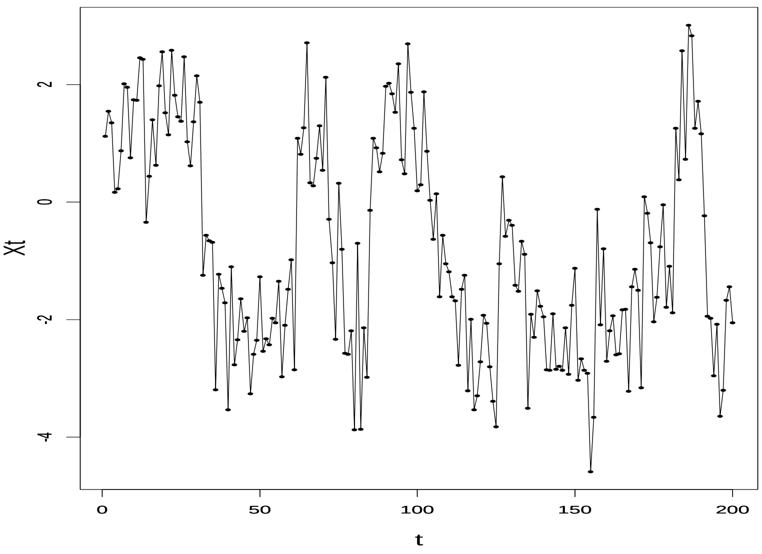
où

- n *est le nombre de neurones cachés,* p *est le nombre d'entrées du réseau (nombre de retards utilisés) ;*
- $(w_i)_{1 \leq i \leq n}$ *sont les « poids » ou « connexions » du réseau,* $w_i \in \mathbb{R}$, $\forall i \in \{1, 2, \ldots, n\}$.
- $(C_i)_{1 \leq i \leq n}$ *sont les centres des gaussiennes, et* $(\sigma_i)_{1 \leq i \leq n}$ *leurs variances,* $(C_i) \in \mathbb{R}^p$, $\sigma_i \in \mathbb{R}$, $\forall i \in \{1, 2, \ldots, n\}$

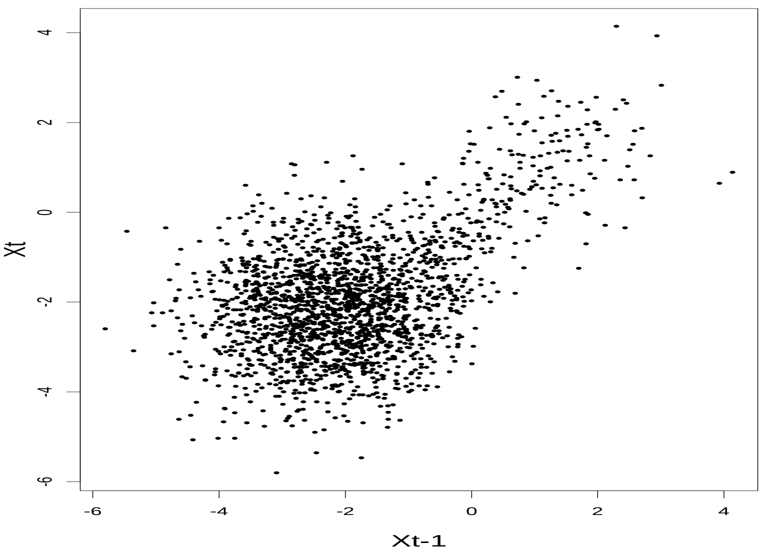
Le vecteur des paramètres du modèle s'écrit alors $\theta = \{w_i; C_i, \sigma_i\}_{1 \leq i \leq n}$

8. Confère chapitre 4, section 4.2 pour plus de détails.

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



(b)



(c)

FIG. 2.7 – *(a) Formule spécifique d'un modèle neuronal du type perceptron multicouches (définition 2.8). (b) 200 points de la série relative au modèle (a). (c) Représentation bivariée de la série relative au modèle (a).*

On peut voir figure 2.8 l'évolution typique d'une série engendrée par un modèle du type *radial basis function* comportant deux entrées. On peut remarquer la forme du nuage de points relative à la distribution bivariée, qui s'éloigne de la classique forme ellipsoïdale.

2.3 Problèmes ouverts

On termine cette section en énumérant un certain nombre de problèmes susceptibles d'être rencontrés lors de la modélisation des modèles décrits précédemment :

– Définition d'un critère statistique de linéarité : il n'existe pas à l'heure actuelle de critère fiable pour évaluer la linéarité d'un processus.

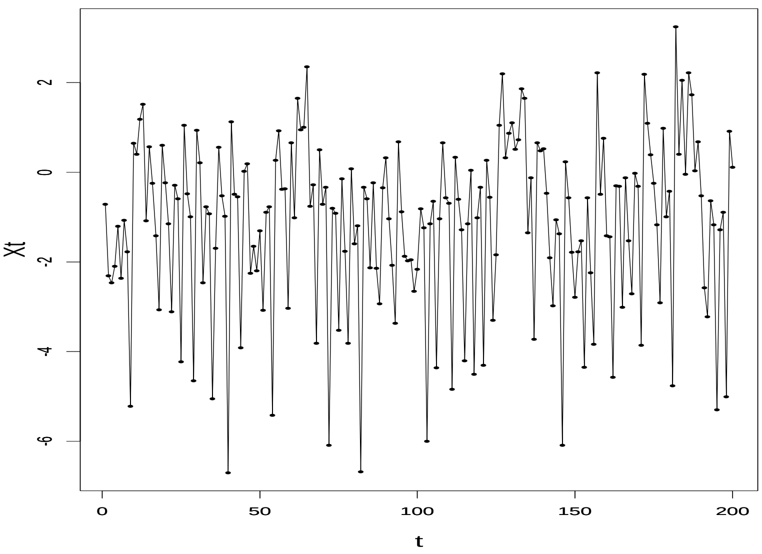
– Comment mener une étude conjointe de la non-stationnarité et de la non-linéarité : la non-stationnarité d'un modèle non-linéaire peut avoir de multiples formes, par exemple la stationnarité par morceaux, la non-stationnarité sur certains intervalles, etc. Dans un cadre non-linéaire, il serait raisonnable d'élargir ou d'affiner le critère de stationnarité.

– Les modèles stochastiques constituent-ils la meilleure alternative pour modéliser les processus chaotiques ? Doit-on chercher à reproduire des processus déterministes, même si les étapes d'identification et d'estimation des paramètres s'avèrent délicates en utilisant une méthodologie stochastique ?

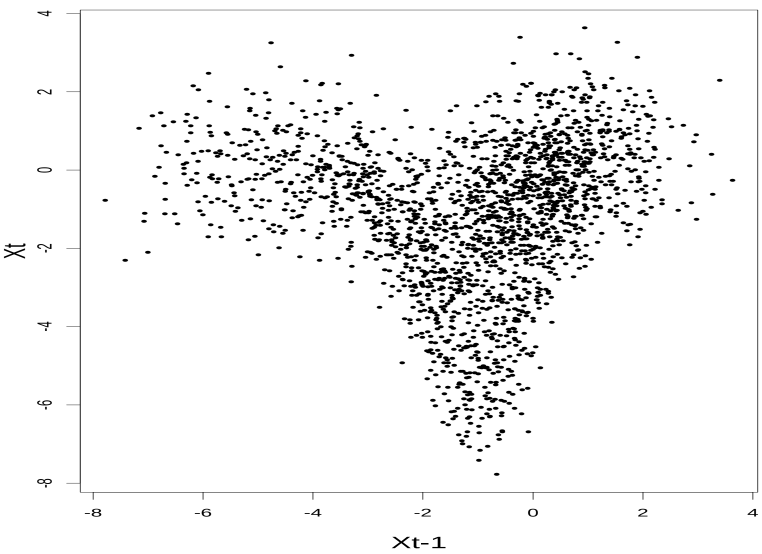
– Quelles propriétés statistiques peuvent être mises en évidence pour permettre une prévision à moyen et long terme, dans le cadre des processus non-linéaires. Dans un cadre linéaire, la meilleure prévision à plusieurs pas est obtenue en utilisant la prévision comme une réalisation. Ceci n'est plus vrai dans un cadre non-linéaire (voir chapitre 4, section 4.3.3). Peut-on trouver des résultats d'optimalité de la méthodologie de prévision à plusieurs pas ?

– Pour les modèles bilinéaires, le problème de l'identification reste ouvert. La théorie ne fournit pas actuellement de méthode fiable pour identifier les modèles bilinéaires.

(a) $$\begin{cases} X_t &= \exp\left[\frac{(X_{t-1}-1)^2+(X_{t-2}-1)^2}{2}\right] - 6\exp\left[\frac{(X_{t-1}+1)^2+(X_{t-2}+1)^2}{2}\right] + \varepsilon_t \\ \varepsilon_t &\sim \mathcal{N}(0,1) \end{cases}$$



(b)



(c)

FIG. 2.8 – *(a) Formule spécifique d'un modèle neuronal du type* RBF *(définition 2.9). (b) 200 points de la série relative au modèle (a). (c) Représentation bivariée de la série relative au modèle (a).*

Chapitre 3

Propriétés statistiques du modèle autorégressif fonctionnel (ARF)

3.1 Introduction

Un modèle autorégressif linéaire correspond assez naturellement à l'idée de régression linéaire à chaque instant sur l'espace des observations passées. Ce type de modèle peut néanmoins s'avérer être insuffisant pour décrire certains processus pour lesquels la relation entre la variable que l'on tente de modéliser à un instant donné et les variables passées n'est pas linéaire. Il peut alors être intéressant d'utiliser une autorégression fonctionnelle (généralement non-linéaire) adaptée au phénomène étudié. Ces idées conduisent à la généralisation du modèle autorégressif classique correspondant au modèle associé à l'équation 2.5 dans le chapitre 2. Des travaux ont été notamment menés par (Tong, 1990; Guégan, 1994; Jones, 1978)),

Dans ce chapitre, afin d'énoncer des propriétés les plus générales possibles, l'ensemble des résultats sont établis pour des processus réels de dimension quelconque. Néanmoins, dans les chapitres 4 et suivants, la dimension des processus étudiés est toujours égale à un.

Soient deux entiers $p, d \geq 1$. Un processus autoregressif fonctionnel sur $\mathbb{R}^d$ est une suite $(X_t)_{t>-p}$ de vecteurs aléatoires vérifiant :

$$X_t = f_\theta(X_{t-1}, \ldots, X_{t-p}) + \varepsilon_t, \qquad t > 0 \tag{3.1}$$

où (ε_t) est un bruit i.i.d. et la fonction f connue. Ici θ est un paramètre appartenant à Θ, sous-ensemble de $\mathbb{R}^s$ (s entier > 0). Par la suite, le sigle $\mathrm{ARF}_d(p)$ désignera un tel modèle. Nous noterons aussi par $X^{(p)} = (X_t^{(p)})_{t>0}$ le *processus vectorisé* associé, défini par $X_t^{(p)} := (X_t, \ldots, X_{t-p+1})$ pour tout $t > 0$. Lorsque $p > 1$, (X_t) n'est pas une chaîne de Markov ; par contre le processus vectorisé $(X_t^{(p)})$ est une chaîne de Markov à valeurs dans $(\mathbb{R}^d)^p$.

Fixons quelques notations. Pour un vecteur $\mathbf{x} := (x_1, \ldots, x_p)$ de $(\mathbb{R}^d)^p$, où (x_i) sont des vecteurs de $\mathbb{R}^d$, on choisit la norme $|\mathbf{x}| := ||x_1|| + \cdots + ||x_p||$ où $||\cdot||$ est la norme euclidienne de $\mathbb{R}^d$. Si A est une matrice, sa norme est définie par : $||A|| = \sup\{||Ax|| \ : \ ||x|| = 1\}$. Le produit scalaire sur $\mathbb{R}^d$ est noté $\langle \cdot, \cdot \rangle$. θ_0 désigne la vraie valeur du paramètre à estimer.

Le *processus de contraste des moindres carrés* est défini par :

$$U_n(\theta) = \frac{1}{n} \sum_{t=1}^{n} ||X_t - f(X_{t-1}, X_{t-2}, ..., X_{t-p} \,; \theta)||^2. \tag{3.2}$$

C'est un processus adapté à la filtration naturelle du processus $\mathcal{F}_n = \sigma(X_t \,; -p < t \leq n)$. *L'estimateur des moindres carrés* est défini pour tout $n \geq 1$, par :

$$\widehat{\theta}_n := \operatorname{Arg} \min_{\theta \in \Theta} U_n(\theta). \tag{3.3}$$

Notons aussi la somme des carrés $S_n(\theta) = nU_n(\theta)$, son gradient DS_n et sa matrice hessienne $D^2 S_n$.

Nous étudions dans ce chapitre, les propriétés asymptotiques de $(\widehat{\theta}_n)$. Lorsque la fonction de régression f est linéaire, on retrouve le modèle classique $\mathrm{AR}_d(p)$ pour lequel les propriétés asymptotiques de l'estimateur des moindres carrés sont bien connues, voir par exemple (Lai et Wei, 1983), (Hannan et Kavalieris, 1986), et pour le cas vectoriel et une loi du logarithme itéré (Duflo et al., 1990).

Lorsque f est non linéaire et pour le cas scalaire ($d = 1$), (Klimbo et Nelson, 1978) considèrent des processus plus généraux que les $\mathrm{ARF}_d(p)$ et le cas où Θ est un ouvert (non nécessairement borné). Ils montrent que sous les conditions :

$$\left\{ \begin{array}{l} \bullet \ \displaystyle\limsup_{n \to \infty, \delta \to 0} \frac{1}{n\delta} \sup_{||\theta - \theta_0|| \leq \delta} ||D^2 S_n(\theta) - D^2 S_n(\theta_0)|| < \infty, \qquad p.s. \\[2ex] \bullet \ \displaystyle\frac{1}{2n} D^2 S_n(\theta_0) \xrightarrow{p.s.} V, \quad \text{avec } V \text{ une matrice } s \times s \text{ définie positive} \\[2ex] \bullet \ \displaystyle\frac{1}{n} DS_n(\theta_0) \xrightarrow{p.s.} 0, \end{array} \right. \tag{3.4}$$

il existe une suite $(\widehat{\theta}_n^*)$, solution de $DU_n(\theta) = 0$, qui converge p.s. vers θ_0. Leur preuve de la consistance, dans un cadre où Θ n'est pas nécessairement borné, utilise un développement de Taylor à l'ordre 2 faisant intervenir le gradient et la matrice hessienne de $S_n(\theta)$. Lorsque Θ est un *compact* - c'est le cadre que nous adoptons -, il est connu que la consistance de $\widehat{\theta}_n$ ne nécessite qu'un bon contrôle du module de continuité de S_n.

En supposant Θ compact et toujours pour le cas scalaire, (Lai, 1994) considère un modèle général de régression stochastique :

$$X_t = f_t(\theta) + \varepsilon_t,$$

où, étant donné une filtration $(\mathcal{G}_t)$, f_t est $\mathcal{G}_{t-1}$ mesurable et (ε_t) est une suite d'accroissements de martingale telle que p.s. $\sup_n \ \mathbb{E}(\varepsilon_t^2 \mid \mathcal{G}_{t-1}) < \infty$.

Afin de formuler ses résultats, posons $J(m) = \{\mathbf{j} = \{j_1, \ldots, j_m\} \ : \ 1 \le j_1 < \cdots < j_m \le s\}$ pour $1 \le m \le s$, et $J = \cup_{m=1}^{s} J(m)$, s étant la dimension de Θ. Pour $\mathbf{j} = \{j_1, \ldots, j_m\}$ et une fonction $\phi(\theta)$, on note $D_{\mathbf{j}}\phi := \partial^m \phi / \partial\theta_{j_1} \cdots \partial\theta_{j_m}$. Si $B(\lambda)$ est une boule centrée en $\lambda \in \mathbb{R}^s$, $B(\lambda\,; \mathbf{j}) := \{\xi \in B(\lambda) \ : \ \xi_j = \lambda_j \text{ pour } j \notin \mathbf{j}\}$. Soit enfin $\Delta_n(\xi\,;\theta_0) = \sum_{t=1}^{n} |f_t(\xi) - f_t(\theta_0)|^2$.

Lai montre que $\widehat{\theta}_n$ est fortement consistant sous les conditions :

(i) pour tout t et tout $\mathbf{j} \in J$, f_t a des dérivées partielles $D_{\mathbf{j}} f_t$ continues sur Θ ;

(ii) pour tout $\lambda \neq \theta_0$, il existe $\rho_\lambda \in]1, 2[$ et une boule ouverte $B(\lambda)$ centrée en λ tel que presque-sûrement :

(a). $\inf_{\xi \in B(\lambda)} \Delta_n(\xi\,;\theta_0) \longrightarrow \infty$;

(b). $\Delta_n(\lambda\,;\theta_0) + \max_{\mathbf{j} \in J} \left[\sum_{t=1}^{n} \int_{B(\lambda\,;\mathbf{j})} |D_{\mathbf{j}} f_t|^2 d\xi_{j_1} \cdots d\xi_{j_m} \right] = O\left(\left\{ \inf_{\xi \in B(\lambda)} \Delta_n(\xi\,;\theta_0) \right\}^{\rho_\lambda} \right).$

Examinant le cas d'une régression linéaire sur une suite prévisible, l'auteur montre que la condition *(ii)-(a)* est « optimale ». Cependant, dans le contexte des $\mathrm{ARF}_d(p)$ on aimerait disposer de conditions suffisantes plus simples et explicites. La condition *(ii)-(b)* impose une régularité $\mathcal{C}^s$ aux fonctions (f_t) si s est la dimension du paramètre, dimension qui peut augmenter rapidement pour les modèles comme des perceptrons multicouches (*cf.* §4). Cette régularité élevée est essentiellement due au fait que l'auteur s'appuie sur une loi des grands nombres pour des martingales à valeurs dans un espace de Hilbert.

Avec certaines conditions supplémentaires, (Lai, 1994) établit également un théorème central limite (TLC) pour $\widehat{\theta}_n$. Ces conditions semblent encore une fois très faibles, mais nécessitent néanmoins d'être explicitées pour les modèles spécifiques tels que les $\mathrm{ARF}_d(p)$. Elles utilisent également la régularité $\mathcal{C}^s$ définie ci-dessus.

Dans ce chapitre, nous proposons, pour les modèles $\mathrm{ARF}_d(p)$, un ensemble simple de conditions assurant successivement la consistance forte, la normalité asymptotique et une loi du logarithme itéré pour l'estimateur $\widehat{\theta}_n$. Cette dernière loi nous assure une identification presque-sûre du vrai modèle à l'aide d'une pénalisation convenable du contraste. En ce qui concerne la fonction de régression f, on demande au maximum une régularité $\mathcal{C}^2$. Nous appuyant sur les résultats de la théorie de la stabilité (Duflo, 1990; Meyn et Tweedie, 1993; Duflo, 1996), ces conditions assurent la stabilité de la chaîne vectorisée $(X_t^{(p)})$, et une loi forte des grands nombres (LFGN) pour les fonctionnelles de la chaîne majorées à l'infini par une fonction moment.

Ce chapitre est organisé comme suit. La section 3.2 décrit le type de LFGN utilisées et rappelle des conditions assurant ces LFGN pour un processus $\mathrm{ARF}_d(p)$. La section 3.4 établit les premières propriétés du processus de contraste (U_n). Nous prouvons la consistance forte de $(\widehat{\theta}_n)$ dans la section 3.5, et sa normalité asymptotique dans la section 3.6. Ces résultats permettent alors la construction d'un test de sous-hypothèse régulière.

Dans la section 3.7, nous précisons la vitesse p.s. de cet estimateur en établissant une loi du logarithme itéré. Nous l'appliquons ensuite au problème de sélection de modèle $\mathrm{ARF}_d(p)$ par la méthode de contraste pénalisé, suivant la démarche de (Senoussi, 1990) et (Guyon, 1995). Les conditions assurant une identification presque-sûre du modèle sont établies lorsque les modèles possibles, y compris le vrai modèle, sont en nombre fini, possèdant en commun un modèle majorant.

3.2 Loi forte des grands nombres pour les fonctions non bornées d'un processus $\mathrm{ARF}_d(p)$

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

$$X_t^{(p)} = \begin{pmatrix} X_t \\ X_{t-1} \\ \vdots \\ X_{t-p+1} \end{pmatrix} = \begin{pmatrix} f(X_{t-1}, \ldots, X_{t-p}\,; \theta) \\ X_{t-1} \\ \vdots \\ X_{t-p+1} \end{pmatrix} + \begin{pmatrix} \varepsilon_t \\ 0 \\ \vdots \\ 0 \end{pmatrix} =: F(X_{t-1}^{(p)}\,; \theta) + \eta_t, \qquad (3.5)$$

avec les définitions implicites correspondantes pour F et η. $(X_t^{(p)})_{t>0}$ est alors un modèle itératif markovien.

On note par $\mathbb{P}_{\theta_0}$ la loi sous le vrai modèle, et sauf indication explicite, toute convergence $\xrightarrow{p.s.}$ (respectivement $\xrightarrow{\mathbf{CL}}$) signifie la convergence p.s. ((respectivemen « en loi ») sous $\mathbb{P}_{\theta_0}$, et ceci quelle que soit la loi initiale de la chaîne $X^{(p)}$.

La LFGN pour les fonctions moments d'un ordre suffisant de cette chaîne constitue l'une des clés principales de notre étude. Plus précisément, nous nous placerons dans le cadre suivant.

HYPOTHÈSE **[S]** DE STABILITÉ. *On suppose que la chaîne$X^{(p)}$ possède sous θ_0 une unique loi invariante μ_{θ_0} satisfaisant, pour un $a \geq 1$:*

(i) pour tout t et toute loi initiale, $\mathbb{E}_{\theta_0}|X_t^{(p)}|^a < \infty$

(ii) $\mu_{\theta_0}(|\cdot|^a) := \int_{(\mathbb{R}^d)^p} |\mathbf{x}|^a \mu_{\theta_0}(d\mathbf{x}) < \infty.$

(iii) Pour toute fonction ϕ de $(\mathbb{R}^d)^p$ dans $\mathbb{R}$, μ_{θ_0}-p.s. continue, satisfaisant

$$|\phi(\cdot)| \leq \text{cte } (1 + |\cdot|^a),$$

on a, pour toute loi initiale , une LFGN, i.e.

$$\frac{1}{n}\sum_{t=1}^{n} \phi(X_t^{(p)}) \xrightarrow{p.s.} \int_{(\mathbb{R}^d)^p} \phi(\mathbf{x})\mu_{\theta_0}(d\mathbf{x}). \qquad \square$$

Les récents développements sur la stabilité des chaînes de Markov (Meyn et Tweedie, 1993; Duflo, 1996) ont motivé cette formulation. Notons qu'en particulier, la condition **[S]**-*(iii)* implique la stabilité de la chaîne $X^{(p)}$. Pour évaluer la stabilité d'une chaîne de Markov on utilise en général le critère de Forster-Lyapounov. On reporte le lecteur aux livres de (Duflo, 1990; Meyn et Tweedie, 1993) et (Doukhan, 1994a) pour plus de détails. Dans le cas d'un $\text{ARF}_d(p)$, nous rappelons ci-dessous des conditions suffisantes assurant ce type de LFGN.

Théorème 3.1 :

Supposons que le modèle $ARF_d(p)$ (3.1) vérifie l'un ou *l'autre ensemble de conditions suivantes :*

[f.1]

- *le bruit (ε_t) a un moment d'ordre $a \geq 1$;*
- *il existe p nombres positifs $\lambda_1, \ldots, \lambda_p$ tels que $\lambda_1 + \cdots + \lambda_p < 1$, satisfaisant pour tout $\mathbf{x}, \mathbf{y} \in (\mathbb{R}^d)^p$,*

$$||f(\mathbf{x};\theta_0) - f(\mathbf{y};\theta_0)|| \leq \lambda_1||x_1 - y_1|| + \cdots + \lambda_p||x_p - y_p||.$$

[f.2]

- *le bruit (ε_t) a une densité strictement positive par rapport à la mesure de Lebesgue , et possède un moment d'ordre $a > 1$;*
- *il existe p nombres positifs $\lambda_1, \ldots, \lambda_p$ tels que $\lambda_1 + \cdots + \lambda_p < 1$, et une constante $\kappa \geq 0$ satisfaisant pour tout $\mathbf{x} \in (\mathbb{R}^d)^p$,*

$$||f(\mathbf{x};\theta_0)|| \leq \lambda_1||x_1|| + \cdots + \lambda_p||x_p|| + \kappa.$$

Alors, le modèle $ARF_d(p)$ (3.1) sous θ_0 satisfait l'hypothèse de stabilité **[S]**.

Remarques.

Ce résultat fournit des critères simples pour une LFGN du type **[S]**-*(iii)*. Dans les deux cas **[f.1]**-**[f.2]**, il s'agit d'une extension directe du modèle linéaire $\text{AR}_d(p)$. Par rapport à **[f.2]**, **[f.1]** est plus restrictif sur la fonction de régression f, et moins restrictif sur le bruit. Notons cependant qu'il existe des fonctions f qui ne satisfaisant ni **[f.1]** ni **[f.2]**, mais l'hypothèse **[S]**. Le critère **[f.2]** est bien connu et est reporté par plusieurs auteurs (voir (Doukhan et Ghindes, 1992; Duflo, 1990; Doukhan et Tsybakov, 1993)). Notre formulation suit celle de (Mokkadem, 1987a).

Le système de conditions **[f.1]** provient de (Duflo, 1996). Le principal intérêt de ces conditions est qu'elles n'impose pas au bruit une densité. Noter aussi qu'un modèle qui rempli les conditions **[f.1]** peut ne pas être ergodique (par exemple le modèle AR(1): $X_t = \frac{1}{2}X_{t-1} + \varepsilon_t$, avec une suite de (ε_t) de Rademacher). Voir aussi (Attali et Pagès, 1995b) pour plus de précisions.

3.3 Ergodicité et β-mélangeance

On s'intéresse maintenant aux propriétés d'ergodicité et de mélangeance du processus ARF$_d(p)$. L'ergodicité du processus implique que la moyenne temporelle du processus converge vers la moyenne statistique. La mélangeance permet d'évaluer les relations de dépendance d'une variable temporelle au cours du temps. Nous nous contentons ici de poser les définitions et de rappeler les théorèmes sans en donner de démonstrations. Ces propriétés ne nous serviront pas directement dans les démonstrations contenues dans ce mémoire. Elles sont néanmoins souvent utiles dès lors que l'on veut utiliser des résultats de convergence et permettent d'accéder à des outils statistiques puissants.

3.3.1 Définitions

On rappelle ici les définitions formelles des propriétés de β-mélangance géométrique, d'α-mélangance géométrique, et d'ergodicité géométrique.

Définition 3.2 :

((Doukhan, 1994b) §1) Soit $(\Omega, \mathcal{C}, P)$ l'espace de probabilité d'un processus $(X_t)_{t\in\mathbb{N}}$, et $\mathcal{A}$, $\mathcal{B}$ deux sous-tribus de $\mathcal{C}$.
Soit le coefficient de mélange suivant :

$$\beta(\mathcal{A}, \mathcal{B}) = E\left[Sup(\mid P(V/\mathcal{A}) - P(V) \mid ; V \in \mathcal{B})\right] \tag{3.6}$$

On dira que $(X_t)_{t\geq 0}$ est un processus géométriquement β-mélangeant si il existe deux constantes $C > 0$ et $0 < \eta < 1$ telles que, $\forall j \in \mathbb{N}$,

$$\beta(\sigma(X_0, ..., X_j), \sigma(X_{j+n}, ...)) \leq C\eta^n \tag{3.7}$$

Remarque 3.3 :

Soit $(X_t)_{t\in\mathbb{N}}$ un processus β-mélangeant à valeurs dans $\mathbb{R}^d$. $\forall p \in \mathbb{N}^*$, $\forall g$ fonction continue, $g : \mathbb{R}^d \mapsto \mathbb{R}^d$, le processus $(g(X_t))_{t\in\mathbb{N}}$ est encore un processus β-mélangeant.
Soient $p \in \mathbb{N}^*$. Alors le processus $(X_t^{(p)})_{t\in\mathbb{N}}$, est aussi un processus β-mélangeant.

Remarque 3.4 :

((Doukhan, 1994b) §1, proposition 1) Soit $(\Omega, \mathcal{C}, P)$ l'espace de probabilité d'un processus $(X_t)_{t\in\mathbb{N}}$, et $\mathcal{A}$, $\mathcal{B}$ deux sous-tribus de $\mathcal{C}$. Soit le coefficient de mélange définis par :

$$\alpha(\mathcal{A}, \mathcal{B}) = Sup(\mid P(U)P(V) - P(U \cap V) \mid ; U \in \mathcal{A}, V \in \mathcal{B}) \qquad (3.8)$$

alors

$$2\alpha(\mathcal{A}, \mathcal{B}) \leq \beta(\mathcal{A}, \mathcal{B}) \qquad (3.9)$$

Donc, si $(X_t)_{t\in\mathbb{N}}$ est un processus géométriquement β-mélangeant, il est aussi géométriquement α-mélangeant.

La notion d'ergodicité implique l'existence d'une loi invariante pour le processus considéré. Cette notion est utile pour utiliser des théorèmes concernant la loi des grands nombres. L'ergodicité géométrique est une notion propre aux processus markoviens, elle englobe en particulier la propriété d'Harris-récurrence.

Définition 3.5 :

Soit $(\Omega, \mathcal{C}, P)$ 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 $\mathcal{C}$ et un réel $\rho < 1$ telle que pour tout $x \in \Omega$:

$$\parallel P(X_t \in \cdot \mid X_0 = x) - \pi(\cdot) \parallel = O(\rho^t) \quad \textit{pour } \pi\textit{-presque tout } x \quad .$$

Remarque 3.6 :

Soit $(X_t)_{t\in\mathbb{N}}$ un processus ergodique à valeurs dans $\mathbb{R}^d$,alors, $\forall g$ fonction intégrable, $g : \mathbb{R}^d \mapsto \mathbb{R}^d$, le processus $(g(X_t))_{t\in\mathbb{N}}$ est encore un processus ergodique. Soit $p > 0$, alors le processus $(X_t^{(p)})_{t\in\mathbb{N}}$ est aussi un processus ergodique.

3.3.2 Cas du processus ARF$_d(p)$

Le théorème suivant fournit des hypothèses suffisantes pour que le modèle ARF$_d(p)$soit géométriquement ergodique et géométriquement β-mélangeant (Doukhan, 1994b). Ce théorème nous servira à établir des conditions pour que les modèles paramétriques neuronaux que l'on décrit chapitre 4 possèdent ce type de propriétés.

Théorème 3.7 :

> *Soit le modèle $ARF_d(p)$ définit équation 3.1. Sous le système de conditions* **[f.2]** *la chaîne vectorisée $(X_t^{(p)})_{t\geq 1}$ associée au modèle $ARF_d(p)$ est géométriquement ergodique. Si de plus cette chaîne est stationnaire, alors le modèle est géométriquement β-mélangeant.*

À partir des résultats de R.L Tweedie, (Mokkadem, 1987a) prouve que sous le système de conditions **[f.2]**, la chaîne $(X_t^p)_{t\in\mathbb{Z}}$ est géométriquement ergodique et telle que μ_{θ_0} est équivalente à la mesure de Lebesgue avec $\mu_{\theta_0}(|\cdot|^a) < \infty$.

La β-mélangeance est alors obtenu par (Doukhan, 1994b), (§2.4.2, proposition 5), lorsque le processus est stationnaire.

Remarque 3.8 :

> On sait que sous le système d'hypothèse **[f.2]** la chaîne vectorisée $(X_t^{(p)})_{t>0}$ associé au modèle ARF$_d(p)$ est stable. Pour obtenir la β-mélangeance géométrique, il suffit alors de supposer que la loi initiale des variables initiales $X_0^{(p)}$ de cette chaîne de Markov est sa loi invariante par la probabilité de transition, ce qui entraîne la stationnarité du modèle.

Pour plus de détails, on pourra aussi consulter l'ouvrage de (Duflo, 1996).

3.4 Modèles, estimateur des moindres carrés et fonction de contraste associée

Nous formulons ci-dessous le cadre exact dans lequel nous nous plaçons. On dira qu'une fonction $g : [0,\infty[\to [0,\infty[$ est un *module de continuité* si *(i).* g est croissante ; *(ii).* $\lim_{x\to 0} g(x) = g(0) = 0$.

CADRE **[M]** DES MODÈLES ÉTUDIÉS :

(i) (Bruit et paramétrage) *On considère une famille de modèles $ARF_d(p)$ définis par (3.1) où :*

(a) *$(\varepsilon_t)_{t>0}$ est un bruit i.i.d. à valeurs dans $\mathbb{R}^d$, centré, de matrice de covariance Γ, indépendant de l'état initial $X_0^{(p)}$ de la chaîne $X^{(p)}$.*

(b) *La famille de modèles considérés est identifiée par la famille de fonctions de régression $\{f(\cdot;\theta)\}$, toutes de $(\mathbb{R}^d)^p$ dans $\mathbb{R}^d$, où le paramètre θ appartient à un*

compact Θ *de* $\mathbb{R}^s$, *s entier* > 0, *tel que* $\theta_0 \in \overset{\circ}{\Theta}$.

(ii) (Stabilité) *Pour le vrai modèle, la fonction de régression* $f(\cdot;\theta_0)$ *ainsi que le bruit* (ε_t) *satisfont à l'hypothèse* **[S]** *de stabilité avec un* $a \geq 2$.

(iii) (Continuité et croissance à l'infini)

(a) pour tout θ, $\mathbf{x} \mapsto f(\mathbf{x};\theta)$ *est* μ_{θ_0}*-p.s. continue ;*

(b) il existe un module de continuité G *tel que :*

$$\forall x \in (\mathbb{R}^d)^p, \forall(\alpha,\beta) \in \Theta^2, \qquad ||f(\mathbf{x};\alpha) - f(\mathbf{x};\beta)|| \leq G(||\alpha-\beta||)(1+|\mathbf{x}|^{a/2}). \qquad \square$$

La condition **[M]**-*(iii)-(b)* impose une continuité en θ à x fixe, et une croissance en $\mathbf{x}$ majorée par $|\mathbf{x}|^{a/2}$. Notons que, puisque Θ est borné, $G(||\alpha-\beta||) \leq G(2\mathrm{diam}(\Theta)) < \infty$ pour tout α, β.

La méthode d'estimation consiste à estimer θ_0 en minimisant une fonctionnelle, *processus de contraste*, $[U_n(\theta)]$ convenable. Pour un exposé classique du cas ergodique, voir (Dacunha-Castelle et Duflo, 1993), et le cas non ergodique, (Guyon, 1995; Bayomog et al., 1996). Notre exposé suit leur présentation.

Notre premier résultat identifie la fontion de contraste pour le paramètre θ_0 associé au contraste des moindres carrés (U_n) (3.2).

Proposition 3.9 :

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

$$\lim_{n\to\infty} [U_n(\theta) - U_n(\theta_0)] = \int_{(\mathbb{R}^d)^p} ||f(\mathbf{x};\theta) - f(\mathbf{x};\theta_0)||^2 \mu_{\theta_0}(d\mathbf{x}) =: K(\theta,\theta_0) \quad . \quad (3.10)$$

De plus, $K(\theta,\theta_0)$ *est une fonction continue en* θ.

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

$$U_n(\theta) - U_n(\theta_0) = \frac{B_n}{n} + \frac{C_n}{n},$$

avec :

$$B_n = \sum_{0\leq t<n} ||\Delta f_t||^2, \quad C_n = 2\sum_{0\leq t<n} \langle \varepsilon_{t+1}, \Delta f_t \rangle.$$

D'après le **[M]**-*(iii)-(b)*,

$$||f(\mathbf{x};\theta_0) - f(\mathbf{x};\theta)||^2 \leq \text{cte } (1+|\mathbf{x}|^a), \qquad \mathbf{x} \in (\mathbb{R}^d)^p. \qquad (3.11)$$

Puisque le modèle sous θ_0 vérifie l'hypothèse de stabilité **[S]**, la LFGN **[S]**-*(iii)* assure que :

$$\frac{B_n}{n} \overset{p.s.}{\longrightarrow} \int_{(\mathbb{R}^d)^p} ||f(\mathbf{x}\,;\theta) - f(\mathbf{x}\,;\theta_0)||^2 \mu_{\theta_0}(d\mathbf{x}).$$

$M_n := C_n/2$ est une martingale de carré intégrable (**[S]**-*(i)*). Son crochet $\langle M\rangle_n$ qui vaut :

$$\langle M\rangle_n = \sum_{0\le t<n} {}^{\mathrm{t}}\Delta f_t \Gamma \Delta f_t \quad ,$$

tend vers $M_\infty \le \infty$. D'après la loi des grands nombres pour les martingales de carrés intégrables (*cf.* par exemple, (Duflo, 1990), théorème 1.III.15, p. 22), 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]**-*(iii)-(b)* et l'inégalité (3.11) assurent que $\theta \mapsto K(\theta,\theta_0)$ est continue. ∎

θ_0 est clairement un minimum absolu de la fonction K. C'est le seul si le modèle vérifie la condition suivante d'identifiabilité.

CONDITION D'IDENTIFIABILITÉ **[D]** : *Le modèle* **[M]** *est dit* identifiable *si :*

pour tout $\theta \in \Theta$, $f(\cdot\,;\theta) = f(\cdot\,;\theta_0)$ $\mu_{\theta_0} - p.s.$ *implique que* $\theta = \theta_0$. □

3.5 Consistance forte

Nous allons maintenant établir la consistance de l'estimateur des moindres carrés défini par (3.2)-(3.3).

Théorème 3.10 :

On se place dans le cadre **[M]** *de la section 3.4. On suppose en plus que la condition d'identifiabilité* **[D]** *est satisfaite. Alors, l'estimateur des moindres carrés* $(\widehat{\theta}_n)$ *est fortement consistant.*

Preuve. Si on note W_n le module de continuité uniforme de U_n, i.e.

$$W_n(\eta) := \sup_{\substack{\alpha,\beta\in\Theta \\ ||\alpha-\beta||\le\eta}} |U_n(\alpha) - U_n(\beta)|, \qquad \eta > 0.$$

une condition suffisante assurant la consistance forte de $(\widehat{\theta}_n)$ est (voir (Guyon, 1995), §3.4) :

$$P_{\theta_0}\left[\limsup_{n\longrightarrow\infty}\left\{W_n\left(\frac{1}{k}\right) \ge \varepsilon_k\right\}\right] = 0 \quad . \tag{3.12}$$

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

$$\begin{aligned}
& n|U_n(\alpha) - U_n(\beta)| \\
= & \left| \sum_{0 \le t < n} \left\langle \delta(X_t^{(p)}\,; \theta_0, \alpha) + \delta(X_t^{(p)}\,; \theta_0, \beta) + \varepsilon_{t+1}\ ,\ \delta(X_t^{(p)}\,; \alpha, \beta) \right\rangle \right| \\
\le & \ G(||\alpha - \beta||)(1 + |X_t^{(p)}|^{a/2}) \sum_{0 \le t < n} \left[\text{cte } (1 + |X_t^{(p)}|^{a/2}) + ||\varepsilon_{t+1}|| \right] \\
\le & \ G(||\alpha - \beta||) \sum_{0 \le t < n} \left[\frac{1}{2} ||\varepsilon_{t+1}||^2 + \text{cte } (1 + |X_t^{(p)}|^{a}) \right]. \qquad (3.13)
\end{aligned}$$

Notons par S_n la somme dans la dernière inégalité. Par la LFGN appliquée à la suite i.i.d. et intégrable $(||\varepsilon_{t+1}||^2)$ d'une part, à la fonction $(1 + |\mathbf{x}|^a)$ d'autre part, S_n/n tend p.s., vers une limite constante $\ell > 0$.

D'après (3.13), $W_n(\eta) \le G(\eta) S_n/n$. Pour k entier positif, définissons $\varepsilon_k = 2\ell G(1/k)$. C'est une suite décroissant vers 0. Alors, pour k fixe (on note *i.s.* pour *infiniment souvent*),

$$\begin{aligned}
\limsup_n \left\{ W_n(\frac{1}{k}) \ge \varepsilon_k \right\} &= \left\{ W_n(\frac{1}{k}) \ge \varepsilon_k \ \ i.s. \ \right\} \\
&\subset \left\{ G(\frac{1}{k}) \frac{S_n}{n} \ge \varepsilon_k \ \ i.s. \ \right\} = \left\{ \frac{S_n}{n} \ge 2\ell \ \ i.s. \ \right\}.
\end{aligned}$$

Sur $A := \left\{ \frac{S_n}{n} \ge 2\ell \ \ i.s. \ \right\}$, S_n/n ne peut converger vers ℓ ; A est donc un évènement négligeable. La condition (3.12) est satisfaite, et la consistance forte établie. ∎

3.6 Normalité asymptotique

Le T.L.C. pour $(\widehat{\theta}_n)$ nécessite des conditions supplémentaires et usuelles sur la dérivabilité d'ordre 2 du processus de contraste (U_n). Si $\phi(\theta)$ est une fonction scalaire, ses dérivées partielles sont notées par $D_i \phi = \frac{\partial \phi}{\partial \theta_i}$, $D_{ij}^2 \phi = \frac{\partial^2 \phi}{\partial \theta_i \partial \theta_j}$, son gradient par $D\phi$ et sa matrice hessienne par $D^2 \phi$. On pose les hypothèses suivantes.

HYPOTHÈSE **[N]** *On suppose que les conditions du cadre* **[M]** *et d'identifiabilité* **[D]** *(section 3.4) sont satisfaites. On suppose de plus qu'il existe un voisinage V de θ_0, sur lequel pour tout $\mathbf{x} \in (\mathbb{R}^d)^p$, les d fonctions coordonnées $f_1, \ldots, f_d$ de $\theta \mapsto f(\mathbf{x}\,; \theta)$ sont deux fois continûment dérivables telles que, pour tout $k = 1, \ldots, d$ et $i, j = 1, \ldots, s$, on ait :*

(i) pour tout $\theta \in V$, $x \mapsto D_i f_k(\mathbf{x}\,; \theta)$ et $x \mapsto D_{ij}^2 f_k(\mathbf{x}\,; \theta)$ sont μ_{θ_0}-p.s. continues.

(ii) pour tout $\mathbf{x} \in (\mathbb{R}^d)^p$,

$$|D_i f_k(\mathbf{x}\,; \theta_0)| \le \text{cte } (1 + |\mathbf{x}|^{a/2}), \qquad \left| D_{ij}^2 f_k(\mathbf{x}\,; \theta_0) \right| \le \text{cte } (1 + |\mathbf{x}|^{a/2}).$$

(iii) il existe un module de continuité σ_{ijk} telle que

$$\left|D^2_{ij}f_k(\mathbf{x}\,;\theta) - D^2_{ij}f_k(\mathbf{x}\,;\theta_0)\right| \le \sigma_{ijk}(||\theta-\theta_0||)(1+|\mathbf{x}|^{a/2}), \qquad \theta \in V. \qquad \square \tag{3.14}$$

Notons que la condition **[N]**-*(iii)* est analogue à **[M]**-*(iii)-(b)* fournissant un contrôle (en $\mathbf{x}$) de la croissance de ces fonctions à l'infini. De même, la compacité de Θ et **[N]**-*(ii)-(iii)* impliquent qu'il existe une constante $\gamma > 0$ telle que :

$$\forall i,j,k,\ \forall\theta \in V,\ \forall x \in (\mathbb{R}^d)^p, \qquad \left|D^2_{ij}f_k(\mathbf{x}\,;\theta)\right| \le \gamma(1+|\mathbf{x}|^{a/2}). \tag{3.15}$$

On en déduit immédiatement un contrôle d'accroissements des dérivées premières :

$$\forall i,k,\ \forall\theta \in V,\ \forall x \in (\mathbb{R}^d)^p, \qquad |D_i f_k(\mathbf{x}\,;\theta) - D_i f_k(\mathbf{x}\,;\theta_0)| \le \gamma||\theta-\theta_0||(1+|\mathbf{x}|^{a/2}). \tag{3.16}$$

Et enfin il existe une autre constante γ' telle que :

$$\forall i,k,\ \forall\theta \in V,\ \forall x \in (\mathbb{R}^d)^p, \qquad |D_i f_k(\mathbf{x}\,;\theta)| \le \gamma'(1+|\mathbf{x}|^{a/2}). \tag{3.17}$$

Posons aussi les matrices :

$$\begin{aligned} Df(\mathbf{x}\,;\theta) &:= [D_j f_k(\mathbf{x}\,;\theta)]_{1\le k\le d,\ 1\le j\le s}, \qquad \text{matrice } d\times s,\\ M(\mathbf{x}\,;\theta) &:= {}^{\mathrm{t}}Df(\mathbf{x}\,;\theta)Df(\mathbf{x}\,;\theta), \qquad \text{matrice } s\times s, && (3.18)\\ D^2_{ij}f(\mathbf{x}\,;\theta) &:= \left[D^2_{ij}f_k(\mathbf{x}\,;\theta)\right]_{1\le k\le d}, \qquad \text{vecteur } d\times 1\,, \quad 1\le i,j\le s. && (3.19)\end{aligned}$$

Les contrôles (3.16)-(3.17) entraînent :

$$\begin{aligned} ||M(\mathbf{x}\,;\theta) - M(\mathbf{x}\,;\theta_0)|| &\le \text{cte } ||\theta-\theta_0||(1+|\mathbf{x}|^a), \quad \mathbf{x} \in (\mathbb{R}^d)^p, \theta \in V && (3.20)\\ ||M(\mathbf{x}\,;\theta)|| &\le \text{cte } (1+|\mathbf{x}|^a), \quad \mathbf{x} \in (\mathbb{R}^d)^p, \theta \in V && (3.21)\\ & && (3.22)\end{aligned}$$

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

$$DU_n(\theta) = -\frac{2}{n}\sum_{0\le t<n} {}^{\mathrm{t}}\varepsilon_{t+1}Df(X_t^{(p)}\,;\theta), \tag{3.23}$$

$$\frac{1}{2}D^2U_n(\theta) = \frac{1}{n}\sum_{0\le t<n} M(X_t^{(p)}\,;\theta) - \frac{1}{n}\left[\sum_{0\le t<n} {}^{\mathrm{t}}\varepsilon_{t+1}D^2_{ij}f(X_t^{(p)}\,;\theta)\right]_{1\le i,j\le s} \tag{3.24}$$

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

Proposition 3.11 :

On se place dans le cadre **[N]** *de la section 3.6. On a pour toute loi initiale de la chaîne* $X^{(p)}$:

(i) $D^2U_n(\theta_0) \xrightarrow{p.s.} I_0$ *avec*

$$I_0 := 2\int_{(\mathbb{R}^d)^p} M(\mathbf{x}\,;\theta_0)\mu_{\theta_0}(d\mathbf{x}). \tag{3.25}$$

(ii) $\sqrt{n}DU_n(\theta_0) \xrightarrow{\mathbf{CL}} \mathcal{N}(0, J_0)$ *avec :*

$$J_0 := 4\int_{(\mathbb{R}^d)^p} {}^tDf(\mathbf{x}, \theta_0)\Gamma Df(\mathbf{x}, \theta_0)\mu_{\theta_0}(d\mathbf{x}). \tag{3.26}$$

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

Preuve. **Partie *(i)*** : dans l'expression (3.24) de D^2U_n écrite pour θ_0, le premier terme converge p.s. vers la matrice I_0. En effet, La LFGN **[S]**-*(iii)* s'applique d'après le contrôle (3.21) de la fonction matricielle $M(\mathbf{x}, \theta_0)$.

Quant au second terme, son élément (i, j), noté $M_n := \sum_{0\leq t<n} {}^t\varepsilon_{t+1} D^2_{ij} f(X_t^{(p)}\,;\theta_0)$, est une martingale de carré intégrable. Son crochet vaut :

$$\langle M\rangle_n = \sum_{0\leq t<n} \mathrm{tr}\left[\Gamma \cdot (D^2_{ij} f\,{}^tD^2_{ij} f)(X_t^{(p)}\,;\theta_0)\right]$$

Compte tenu de (3.15), un argument analogue à celui utilisé à la fin de la preuve de la proposition 3.9 permet de conclure que M_n/n tend p.s. vers 0. D'où la conclusion *(i)*.

Partie *(ii)* : notons cette fois-ci :

$$M_n := -\frac{n}{2}DU_n(\theta_0) = \sum_{0\leq t<n} \varepsilon_{t+1} Df(X_t^{(p)}\,;\theta_0). \tag{3.27}$$

C'est une martingale vectorielle, de carré intégrable d'après (3.17). Son crochet vaut :

$$\langle M\rangle_n = \sum_{0\leq t<n} {}^tDf(X_t^{(p)}\,;\theta_0)\Gamma Df(X_t^{(p)}\,;\theta_0). \tag{3.28}$$

Toujours d'après (3.17), chaque terme de la fonction matricielle

$$\mathbf{x} \mapsto J(\mathbf{x}\,;\theta_0) := {}^tDf(X_t^{(p)}\,;\theta_0)\Gamma Df(X_t^{(p)}\,;\theta_0)$$

est majorée (en module) par cte $(1+|\mathbf{x}|^a)$. Ainsi d'après la LFGN **[S]**-*(iii)*,

$$\frac{1}{n}\langle M\rangle_n \xrightarrow{p.s.} \int_{(\mathbb{R}^d)^p} J(\mathbf{x}\,;\theta_0)\mu_{\theta_0}(d\mathbf{x}) = \frac{J_0}{4}. \tag{3.29}$$

Le TLC *(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 $\varepsilon > 0$, en notant $\Delta_t := M_t - M_{t-1} = \varepsilon_{t+1}Df(X_t^{(p)}\,;\theta_0)$,

$$L_n := \frac{1}{n}\sum_{0\le t<n} \mathbb{E}\left[||\Delta_t||^2\,\mathbb{I}_{\{||\Delta_t||\ge\varepsilon\sqrt{n}\}}\,|\,\mathcal{F}_{t-1}\right] \xrightarrow{\mathbf{P}_{\theta_0}} 0. \tag{3.30}$$

Soit $A>0$ et :

$$F_n(A) := \frac{1}{n}\sum_{0\le t<n} \mathbb{E}\left[||\Delta_t||^2\,\mathbb{I}_{||\Delta_t||\ge\varepsilon A}\,|\,\mathcal{F}_{t-1}\right] = \frac{1}{n}\sum_{0\le t<n} h(X_t^{(p)},A),$$

avec :

$$h(\mathbf{x},A) = \mathbb{E}\left[{}^{\mathrm{t}}Df(\mathbf{x}\,;\theta_0)\varepsilon_1{}^{\mathrm{t}}\varepsilon_1 Df(\mathbf{x}\,;\theta_0)\,\mathbb{I}_{\{||Df(\mathbf{x}\,;\theta_0)\varepsilon_1||>A\}}\right].$$

Il est clair que d'après (3.17),

$$h(\mathbf{x},A) \le \text{cte } (1+|\mathbf{x}|^a). \tag{3.31}$$

D'où, toujours en vertu de **[S]**-*(iii)*,

$$F_n(A) \xrightarrow{p.s.} \phi(A) := \int_{(\mathbb{R}^d)^p} h(\mathbf{x},A)\mu_{\theta_0}(d\mathbf{x}).$$

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

- l'équation (3.31) établit la domination puisque $|\cdot|^a$ est μ_{θ_0}-intégrable ;
- Pour $\mathbf{x}$ fixé, posons $\xi := {}^{\mathrm{t}}Df(\mathbf{x}\,;\theta_0)\varepsilon_1$. $||\xi||^2$ est une variable intégrable, puisque $\mathbb{E}||\varepsilon_1||^2$ est finie. Alors :

  $$0 \le h(\mathbf{x},A) = \mathbb{E}\left[||\xi||^2\,\mathbb{I}_{\{||\xi||^2>A^2\}}\right],$$

  tend vers 0 quand $A\to\infty$.

À A fixé, on a $\varepsilon\sqrt{n} > A$ pour n assez grand, et $L_n = F_n(\varepsilon\sqrt{n}) \le F_n(A)$. Donc p.s., $\limsup_n L_n \le \phi(A)$. En faisant tendre A vers ∞, on obtient p.s., $\lim L_n = 0$. La condition de Lindeberg (3.30) est ainsi satisfaite. Et $M_n/\sqrt{n} \xrightarrow{\mathbf{CL}} \mathcal{N}(0, J_0/4)$. ∎

Théorème 3.12 :

On suppose satisfaites les hypothèses **[N]**. *Alors, pour toute loi initiale de la chaîne vectorisée* $X^{(p)}$,

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

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

$$0 = DU_n(\widehat{\theta}_n) = DU_n(\theta_0) + \Delta_n(\widehat{\theta}_n)(\widehat{\theta}_n - \theta_0), \tag{3.32}$$

avec :

$$\Delta_n(\widehat{\theta}_n) := \int_0^1 D^2_\theta U_n \left[\widehat{\theta}_n + u(\widehat{\theta}_n - \theta_0)\right] du.$$

Supposons vérifiée la condition suivante (*cf.* Lemme (3.13)) :

$$\Delta_n(\widehat{\theta}_n) - D^2_\theta U_n(\theta_0) \xrightarrow{\mathbb{P}_{\theta_0}} 0, \tag{3.33}$$

et compte tenu de la proposition 3.11, on obtient le résultat annoncé. ∎

Lemme 3.13 :

Dans le cadre du théorème 3.12, on a :

$$\Delta_n(\widehat{\theta}_n) - D^2_\theta U_n(\theta_0) \xrightarrow{p.s.} 0, \qquad \Delta_n(\widehat{\theta}_n) \xrightarrow{p.s.} I_0.. \tag{3.34}$$

Preuve. Pour $\theta \in V$, notant (κ_j) une suite de constantes positives, on a d'après (3.15), (3.21) et (3.24) :

$$\begin{aligned}
&\frac{n}{2} \left\| D^2_\theta U_n(\theta) - D^2_\theta U_n(\theta_0) \right\| \\
&= \left\| \sum_{0 \leq t < n} \left[M(X_t^{(p)};\theta) - M(X_t^{(p)};\theta_0)\right] - \sum_{0 \leq t < n} {}^{\mathrm{t}}\varepsilon_{t+1} \left[D^2_{ij} f(X_t^{(p)};\theta) - D^2_{ij} f(X_t^{(p)};\theta_0)\right]_{1 \leq i,j \leq s} \right\| \\
&\leq \kappa_1 ||\theta - \theta_0|| \sum_{0 \leq t < n} (1 + |X_t^{(p)}|^a) + \kappa_2 \sum_{1 \leq i,j \leq s, 1 \leq k \leq d} \sigma_{ijk}(||\theta - \theta_0||) \sum_{0 \leq t < n} ||\varepsilon_{t+1}||(1 + |X_t^{(p)}|^a) \\
&\leq \kappa_3 \sigma(||\theta - \theta_0||) \sum_{0 \leq t < n} ||\varepsilon_{t+1}||^2 + \kappa_4 \left[||\theta - \theta_0|| + \sigma(||\theta - \theta_0||)\right] \sum_{0 \leq t < n} (1 + |X_t^{(p)}|^a),
\end{aligned}$$

où on a noté $\sigma(z) := \sum_{i,j,k} \sigma_{ijk}(z)$. D'autre part, d'après (3.32)

$$\|\Delta_n(\widehat{\theta}_n) - D^2_\theta U_n(\theta_0)\|$$

$$
\begin{aligned}
&= \left\| \int_0^1 \left\{ D^2_\theta U_n \left[\widehat{\theta}_n + u(\widehat{\theta}_n - \theta_0) \right] - D^2_\theta U_n(\theta_0) \right\} du \right\| \\
&\leq 2\kappa_3 \sigma(||\widehat{\theta}_n - \theta_0||) \frac{1}{n} \sum_{0 \leq t < n} ||\varepsilon_{t+1}||^2 + 2\kappa_4 \left[||\theta - \theta_0|| + \sigma(||\theta - \theta_0||) \right] \frac{1}{n} \sum_{0 \leq t < n} (1 + |X_t^{(p)}|^a).
\end{aligned}
$$

Comme les deux séries converges p.s. et $\widehat{\theta}_n \overset{p.s.}{\longrightarrow} \theta_0$, on en déduit la convergence p.s. vers 0 de $\Delta_n(\widehat{\theta}_n) - D^2_\theta U_n(\theta_0)$. Le deuxième résultat est une conséquence de la proposition 3.11.

∎

Test asymptotique de différence de contraste :

Soit q un entier inférieur à s. L'hypothèse « $\theta \in \Theta \subset \mathbb{R}^s$ » est notée (H_s). Une sous-hypothèse (H_q) de (H_s) exprime le fait que θ appartient à un sous-ensemble de Θ de dimension paramétrique plus petite que s. Plus précisément, soit Λ un ouvert de $\mathbb{R}^q$ et r une fonction de classe $\mathcal{C}^2$ de Λ dans $\mathbb{R}^s$. On considère une *sous-hypothèse régulière* (H_q) de la forme : « $\theta = r(\alpha),\ \alpha \in C$ », pour un compact C de Λ, et telle que $\theta_0 = r(\alpha_0)$ pour un α_0 intérieur à C, et $R_0 = D_\alpha r(\alpha_0)$ soit de rang plein q.

On notera $\bar{U}_n(\alpha) = U_n(r(\alpha))$ le contraste sous (H_q), $\hat{\alpha}_n$ un estimateur du minimum de contraste associé ; $\bar{I}_0$ la matrice analogue à I_0 (*cf.* (3.25)), mais relative au contraste $\bar{U}_n$.

Pour tester (H_q) contre (H_s), on utilisera la statistique de différence de contrastes :

$$T_n = 2n \left[\bar{U}_n(\hat{\alpha}_n) - U_n(\hat{\theta}_n) \right].$$

En applicant des résultats classiques (*cf.* théorème 3.4.6 de (Guyon, 1995) et (Bayomog et al., 1996)), nous obtenons la loi asymptotique de la statistique T_n du test de (H_q) contre (H_s).

Proposition 3.14 :

Supposons que le modèle (3.1) et les contrastes (U_n), $(\bar{U}_n)$ satisfont les hypothèses du théorème 3.12 et que les matrices I_0, $\overline{I}_0$ sont inversibles. Alors, sous la sous-hypothèse (H_q),

$$T_n \overset{\mathbf{CL}}{\longrightarrow} \sum_{i=1}^{s-q} \lambda_i \chi^2_{i,1}, \tag{3.35}$$

où les $(\chi^2_{i,1})$ sont $s-q$ variables i.i.d. de χ^2_1, et (λ_i) les $s-q$ valeurs propres strictement positives de la matrice :

$$A_0 = J_0(I_0^{-1} - R_0 \bar{I}_0^{-1t} R_0). \tag{3.36}$$

Remarques. Dans le cas scalaire, A_0 est proportionnelle à une matrice idempotente de rang $s-q$ avec $\lambda_1 = \cdots = \lambda_{s-q} = 2\sigma^2$. On retrouve alors un test du χ^2 à $s-q$ degrés de liberté.

3.7 Vitesse et identification presque-sûre par contraste pénalisé

Le cadre de la section précédente nous a permis d'obtenir un TLC pour l'estimateur des moindres carrés $\widehat{\theta}_n$. (Mangeas et Yao, 1996) montrent que ce même cadre, très légèrement renforcé, garantit l'existence d'une loi du logarithme itéré pour le processus gradient du contraste. Cette loi du logarithme itéré (L.L.I.) est immédiatement transférable à l'estimateur $\widehat{\theta}_n$, ce qui permet de préciser sa vitesse de convergence. Nous terminons enfin par une application de ce résultat à l'identification p.s. de modèles par contraste pénalisé.

3.7.1 Vitesse et loi du logarithme itéré

Dans cette partie, la norme vectorielle d'un vecteur réel est sa norme euclidienne, et celle d'une matrice réelle A la racine carrée du rayon spectral de ${}^tAA$. Pour une matrice réelle et symétrique A, $\lambda_{max}A$ (resp. $\lambda_{min}A$) désignera la plus grande (resp. plus petite) valeur propre de A.

Théorème 3.15 :

On se place dans le cadre du théorème 3.12. On suppose de plus que :

(i) le bruit (ε_t) a un moment d'ordre > 2 ;

(ii) la matrice de covariance Γ du bruit et la matrice I_0 sont inversibles.

Alors, presque-sûrement,

$$\limsup_n \sqrt{\frac{n}{2\log\log n}}||DU_n(\theta_0)|| \quad \leq \quad \sqrt{\lambda_{max}J_0}, \tag{3.37}$$

$$\limsup_n \sqrt{\frac{n}{2\log\log n}}||\widehat{\theta}_n - \theta_0|| \quad \leq \quad \frac{\sqrt{\lambda_{max}J_0}}{\lambda_{min}I_0}. \tag{3.38}$$

Preuve. Réexaminons la martingale M_n (3.27) associée au gradient du contraste. Soit u un vecteur non nul de $\mathbb{R}^s$, et $\widetilde{M}_n := \langle M_n, u\rangle = \sum_{0\leq t<n} \varepsilon_{t+1} Df(X_t^{(p)}\,;\theta_0)u$. On notera par la suite $f_t' = Df(X_t^{(p)}\,;\theta_0)$. $\widetilde{M}_n$ est une martingale scalaire de puissance $2+2\alpha$ intégrable

pour tout $\alpha \in (0, a/2-1]$. Notons :

$$\begin{aligned} T_t &:= \mathbb{E}\left[|\widetilde{M}_{t+1} - \widetilde{M}_t|^{2+2\alpha} \mid \mathcal{F}_t\right]^{1/(2+2\alpha)} = ({}^{\mathrm{t}}u{}^{\mathrm{t}}f_t'\Gamma f_t'u)^{1/2}, \\ \tau_n &= \sum_{0\le t<n} T_t^2 = {}^{\mathrm{t}}u\langle M\rangle_n u, \end{aligned}$$

où le crochet $\langle M\rangle_n$ est donné par (3.28). En vertu de (3.29), τ_n/n tend p.s. vers $\frac{1}{4}{}^{\mathrm{t}}uJ_0u$ - noté β_u -, qui est strictement positif d'après les hypothèses. D'où $\tau_n \xrightarrow{p.s.} \infty$. La loi du logarithme itéré pour une martingale de puisance $2+2\alpha$ intégrable (*cf.* par exemple (Duflo et al., 1990), corollaire 6) assure que :

$$p.s. \qquad \limsup_n \frac{|\widetilde{M}_n|}{\sqrt{2\tau_{n-1}\log\log\tau_{n-1}}} \le 1, \tag{3.39}$$

si *la série* $\sum (T_n^2/\tau_n)^{1+\alpha}$ *est p.s. convergente.*

Posons $s_n := T_1^{2+2\alpha} + \cdots + T_n^{2+2\alpha}$. Pour un $\alpha < a/2-1$, on a la LFGN pour $(T_n^{2+2\alpha})$ d'après **[S]**-*(iii)*, i.e. s_n/n converge p.s. vers une constante $\gamma \ge 0$. Par ailleurs, $(T_n^2/\tau_n)^{1+\alpha} \sim$ cte $T_n^{2+2\alpha}/n^{1+\alpha}$ et

$$\sum_{k=1}^{n} \frac{T_k^{2+2\alpha}}{k^{1+\alpha}} = \frac{s_n}{n^{1+\alpha}} + \sum_{k=1}^{n-1}\left[\frac{1}{k^{1+\alpha}} - \frac{1}{(k+1)^{1+\alpha}}\right] - s_1.$$

Puisque $s_n/n^{1+\alpha}$ tend p.s. vers 0, et

$$\frac{1}{k^{1+\alpha}} - \frac{1}{(k+1)^{1+\alpha}} \sim \frac{1+\alpha}{k^{2+\alpha}},$$

la série $\sum T_n^{2+2\alpha}/n^{1+\alpha}$ converge p.s et il en est de même pour $\sum T_n^{2+2\alpha}/\tau_n^{1+\alpha}$.

En remplaçant $\widetilde{M}_n$ dans (3.39), compte tenu de (3.23) et de $2\tau_{n-1}\log\log\tau_{n-1} \sim 2\beta_u n\log\log n$, on obtient :

$$p.s. \qquad \limsup_n \sqrt{\frac{n}{2\log\log n}}|\langle DU_n(\theta_0), u\rangle| \le \sqrt{{}^{\mathrm{t}}uJ_0u}.$$

D'où la L.L.I. (3.37).

Pour la seconde L.L.I. (3.38), reprenons le développement de Taylor (3.32). Le lemme 3.13 assurent que $\Delta_n(\widehat{\theta}_n) \xrightarrow{p.s.} I_0$, d'où (3.38). ∎

3.7.2 Identification presque-sûre

Suivant la présentation de (Guyon, 1995), on suppose que l'espace des paramètres $\Theta \subset \mathbb{R}^M$ où $\mathbb{R}^M$ correspond au modèle majorant ; soit $\mathcal{F}$ une famille finie de sous-espace de $\mathbb{R}^M$, $\delta \in \mathcal{F}$, l'élement générique de $\mathcal{F}$, $|\delta|$ sa dimension et $\Theta_\delta := \Theta \cap \delta$ le sous-espace (sous-modèle) paramétrique associé. On suppose que la vraie valeur $\theta_0 \in \Theta_{\delta_0}$, $\delta_0 \in \mathcal{F}$ étant le sous-espace minimal associé à θ_0 (si $\delta \subset \delta_0$ et $\delta \neq \delta_0$, $\theta_0 \notin \Theta_\delta$).

Soit $[c(n)]$ une suite positive. Au vu de la réalisation $(X_t)_{-p<t\leq n}$, on utilise comme fonction de décision le *contraste pénalisé à la vitesse* $c(n)$ par la dimension du modèle :

$$W_{n,\delta}(\theta) := U_n(\theta) + \frac{c(n)}{n}|\delta|, \qquad \text{pour } \delta \in \mathcal{F}, \theta \in \Theta_\delta. \tag{3.40}$$

Notons :

$$\overline{W}_{n,\delta} = \overline{U}_{n,\delta} + \frac{c(n)}{n}|\delta|, \qquad \text{avec} \quad \overline{U}_{n,\delta} = U_n(\widehat{\theta}_{n,\delta}), \quad \text{et} \quad \widehat{\theta}_{n,\delta} = \operatorname{Arg}\min_{\theta\in\Theta_\delta} U_n(\theta).$$

On choisira $\widehat{\delta}_n$ réalisant le minimum de $\overline{W}_{n,\delta}$ sur $\mathcal{F}$:

$$\widehat{\delta}_n = \operatorname{Arg}\min_{\delta\in\mathcal{F}} \overline{W}_{n,\delta},$$

qui répond au *principe de parcimonie d'Akaïké* avec la vitesse $c(n)$.

Appliquant les résultats de (Senoussi, 1990; Guyon, 1995), nous avons le résultat suivant d'identification presque-sûre du vrai modèle δ_0.

Proposition 3.16 :

On se place dans le cadre du théorème 3.15. Si la vitesse de pénalisation $c(n)$ *est telle que :*

$$\lim_n \frac{c(n)}{n} = 0, \qquad \liminf_n \frac{c(n)}{2\log\log n} > \frac{\lambda_{max} J_0}{2\lambda_{min} I_0}, \tag{3.41}$$

alors, le couple $(\widehat{\delta}_n, \widehat{\theta}_{n,\widehat{\delta}_n})$ *converge* $\mathbb{P}_{\theta_0}$*-p.s. vers la vraie valeur* (δ_0, θ_0).

Preuve. Il suffit d'appliquer le théorème (3.4.8) de (Guyon, 1995) dont les conditions d'application se vérifient immédiatement ici grâce au théorème 3.15. ∎

Chapitre 4

Perceptron multicouches et prévision de séries temporelles

4.1 Notations-définitions

C'est dans les années 40, que von Neumann, Turing, Mc Cullogh, Minsky, Pitts et d'autres ont tenté de mettre à profit les connaissances nouvelles apportées par la biologie et les sciences cognitives sur le cerveau pour concevoir des systèmes censés reproduire certaines de ses fonctionnalités :

- l'apprentissage de tâches complexes,
- la capacité de raisonnement et de déduction,
- la possibilité d'évaluation, d'estimation et de résolution de problèmes.

Deux écoles ont émergé, l'une s'orientant vers une vision « connexioniste » (Mc Cullogh, Pitts, Minsky,...), l'autre adoptant une démarche « symbolique » (Von Neumann, Turing,...). L'approche symbolique aussi appelée *IA forte*, plus orientée vers l'aspect cognitif, repose sur une modélisation symbolique de l'univers 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 éventuels liens et relations. On lui doit en particulier rien moins que l'ordinateur, et les bases de l'Intelligence Artificielle. L'approche connexioniste, appelée aussi *IA faible*, s'inspire exclusivement de la description biologique, en tentant de construire des systèmes proches du cerveau dans leur organisation, afin de 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.

Dès 1943, W. McCulloch et W. Pitts ont formalisé le modèle du neurone formel basé sur les observations neurophysiologiques des neurones du système nerveux. Ce neurone formel reste encore aujourd'hui un élément de base de la plupart des modèles connexionistes. De nombreuses variantes ont été proposées, plus ou moins biologiquement plausibles, mais reprenant toujours des concepts présentés à cette époque. On sait néanmoins aujourd'hui que ce modèle n'est qu'une approximation du neurone biologique, et qu'en aucune façon il ne peut servir pour une compréhension profonde du système nerveux.

4.1.1 Le neurone formel

Le neurone formel introduit par McCullogh et Pitts en 1943 (Cullogh et Pitts, 1943) est un automate reproduisant la composée de plusieurs fonctions très simples (fig. 4.1). Chacune des p liaisons synaptiques entrantes est affectée d'un poids θ_i, $i \in \{1, 2, \ldots, p\}$, stimulée par une entrée réelle x_i, $i \in \{1, 2, \ldots, p\}$. Par convention, on ajoute aussi une entrée constante égale à 1, pondérée par un poids θ_0. 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 :

(i) son potentiel, c'est-à-dire la somme pondérée des entrées $\sum_{i=1}^{p} \theta_i x_i$,

(ii) 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 ψ la plus simple est la fonction signe $S : \mathbb{R} \mapsto \mathbb{R}$ (ou fonction seuil) définie par

$$\left\{ \begin{array}{lcll} S(x) & = & 1 & \text{si } x \geq 0 \quad ; \\ S(x) & = & -1 & \text{sinon,} \end{array} \right. \tag{4.1}$$

(voir aussi fig. 4.2), clairement non-linéaire. On peut aussi la définir à valeurs dans $\{0, 1\}$. Dans de nombreuses applications, cette fonction est remplacée par d'autres plus régulières (continues, bornées et dérivables sur $\mathbb{R}$) telles que, par exemple, la fonction de Gauss (fig.4.3).

Néanmoins, la famille de fonctions la plus utilisée est la famille des fonctions sigmoïdes :

$$x \mapsto \sigma_{c,k,r}(x) = c\frac{e^{kx} - 1}{e^{kx} + 1} + r \; ; \; c, k, r \in \mathbb{R} \; ; \; c, k > 0 \; . \tag{4.2}$$

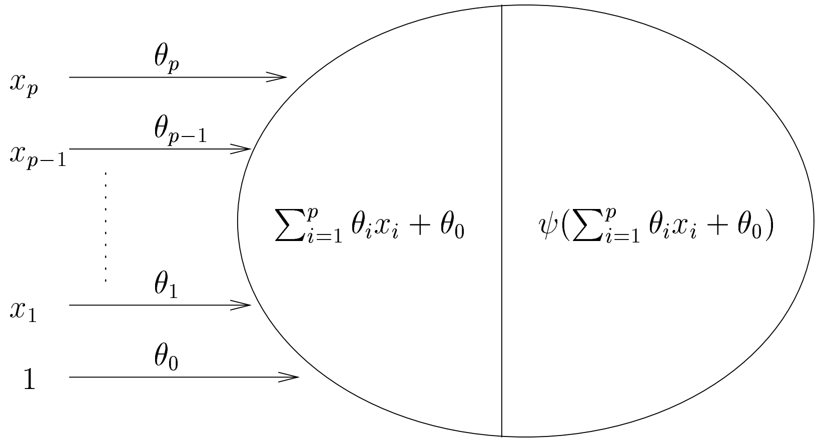


FIG. 4.1 – *Le neurone : unité de base du traitement neuronal.*

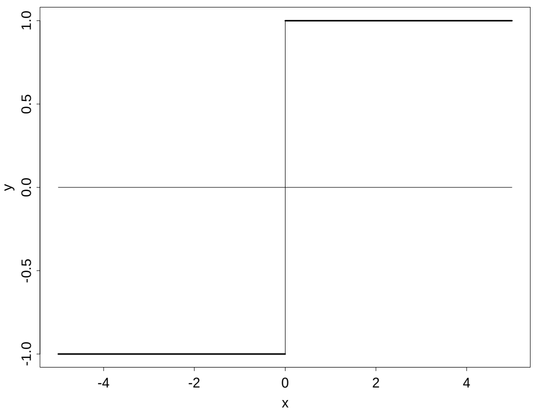


FIG. 4.2 – *Fonction signe aussi appelée fonction seuil ou fonction de Heavyside.*

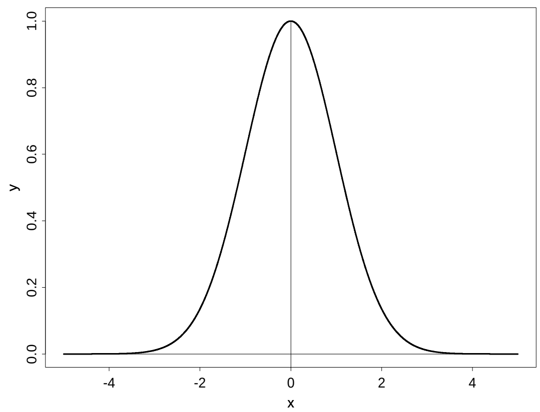


FIG. 4.3 – *Fonction de Gauss.*

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,

$$\sigma_{1,k,0}(x) = \frac{e^{kx} - 1}{e^{kx} + 1} = \tanh\left(\frac{kx}{2}\right) \quad . \tag{4.3}$$

Il est alors facile de voir que sur $\mathbb{R}^*$, la fonction $\sigma_{1,k,0}(x)$ tend simplement vers la fonction signe S (équation 4.1), quand $k \longrightarrow +\infty$.

La fonction $x \mapsto \tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ (fig. 4.4), une des fonctions sigmoïdes les plus utilisées, est alors obtenue en prenant $\{c = 1\ ;\ k = 2\ ;\ r = 0\}$, et la fonction logistique $x \mapsto 1/(1 + e^{-x})$ (fig. 4.5) est obtenue en prenant $\{c = 1/2\ ;\ k = 1\ ;\ r = 1/2\}$.

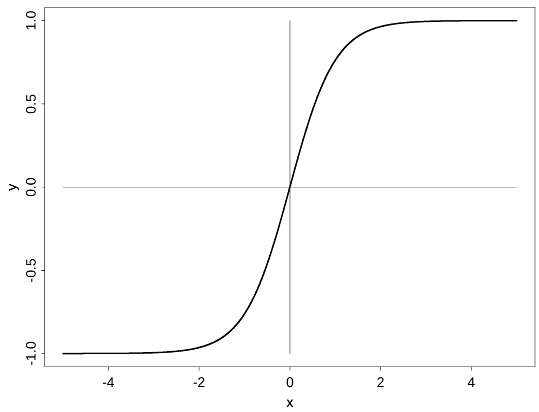


FIG. 4.4 – *Fonction tangente hyperbolique.*

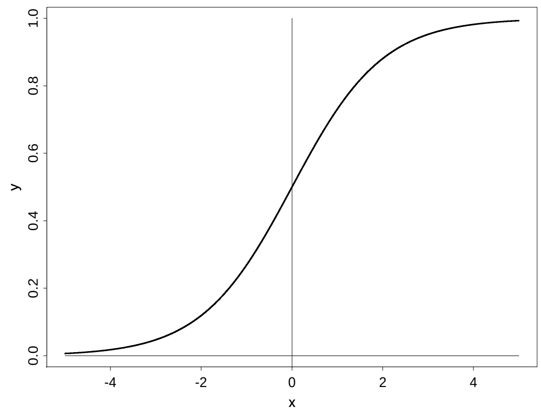


FIG. 4.5 – *Fonction logistique.*

A son apparition, le neurone formel de Mc Cullogh et Pitts suscita un vif intérêt parmi les pionniers du connexionisme. Cependant, malgré quelques tentatives d'implémentation d'opérateurs logiques, la première application notable n'apparut qu'au début des années 60.

4.1.2 Le perceptron simple

Le nom de perceptron a été introduit par (Rosenblatt, 1962) pour désigner un ensemble de neurones formels connectés. Il fût le premier à y associer un algorithme d'*apprentissage*. Dans la structure du *perceptron simple* (voir fig. 4.6), les unités de la couche d'entrée sont directement reliées à l'unité de la couche de sortie. La fonction d'activation associée aux neurones de la couche d'entrée est l'identité, alors qu'elle est la fonction signe pour les unités 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, voir 4.1.1) 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 finies $\mathcal{A}$ et $\mathcal{B}$ de $\mathbb{R}^p$. Après l'apprentissage, on désire en effet que le réseau réponde '1' sur présentation de tout élément de

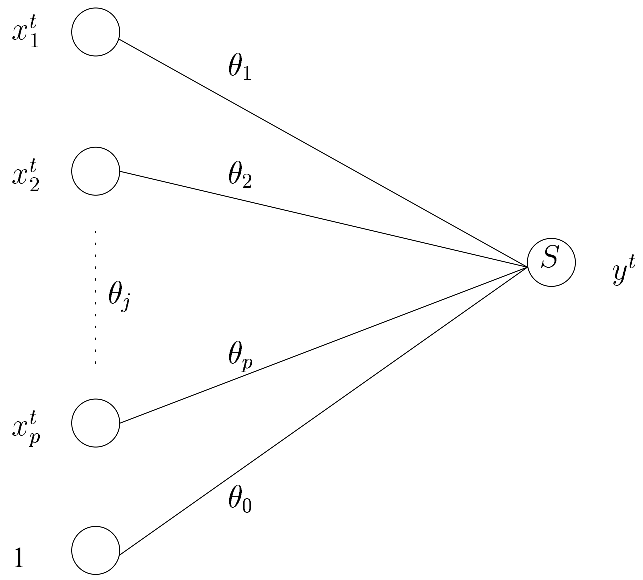


FIG. 4.6 – *Le perceptron simple. S est la fonction signe représentée fig. 4.2. On présente le $t^{ème}$exemple de la base d'apprentissage.*

$\mathcal{A}$ et '-1' sur présentation de tout élément de $\mathcal{B}$. Autrement dit, on cherche à calculer le vecteur des paramètres (les poids des connexions) θ tel que $\forall x = (x_1, x_2, \ldots, x_p) \in \mathcal{A}$, $\sum_{i=1}^{p} \theta_i \, x_i + \theta_0 > 0$ et $\forall x' = (x'_1, x'_2, \ldots, x'_p) \in \mathcal{B}$, $\sum_{i=1}^{p} \theta_i \, x_i + \theta_0 < 0$.

Pour cela, on présente successivement des éléments d'une base d'exemples de taille T. Cette base est composée d'un ensemble de vecteurs $(\mathbf{x}^t)_{t=1,2,\ldots,T} \in \mathcal{A} \cup \mathcal{B}$, $\mathbf{x}^t = (x_1^t, \ldots, x_p^t)$, et d'un ensemble de valeurs binaires associées $(d^t)_{t=1,2,\ldots,T} \in \{-1, 1\}$ correspondant à '1' si $\mathbf{x}^t \in \mathcal{A}$ et '-1' si $\mathbf{x}^t \in \mathcal{B}$ (on appelle l'ensemble des valeurs $(d^t)_{t=1,2,\ldots,T}$ les *valeurs désirées*). Lorsqu'on présente $\mathbf{x}^t$, on ajuste alors le vecteur des paramètres $\theta^t = \{\theta_0^t, \theta_1^t, \ldots, \theta_p^t\}$ 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. Puisque les sorties désirées sont connues *a priori*, on parle d'*apprentissage supervisé*.

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

$$\theta^0 = 0 \in \mathbb{R}^{p+1}, \ \forall t, \ \theta^{t+1} = \theta^t + \frac{1}{2}\varepsilon \left(d^t - y^t\right) \mathbf{x}^t \quad .$$

où ε est un pas unidimensionnel de déplacement dans l'espace des paramètres, petit, positif, et dont la valeur est à régler.

Le vecteur des paramètres n'est modifié qu'en cas d'erreur de classement. Il s'agit donc d'un algorithme du type « correction d'erreurs ».

Rosenblatt dans les années 60 a démontré le théorème de convergence suivant qui suscita beaucoup d'espoir en son temps.

Théorème 4.1 :

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

$$\forall \mathbf{x} \quad = \quad (x_1, x_2, \ldots, x_p) \in \mathcal{A},\ \sum_{i=1}^{p} \theta_i^\infty\ x_i + \theta_0^\infty > 0, \quad et \tag{4.4}$$

$$\forall \mathbf{x} \quad = \quad (x_1, x_2, \ldots, x_p) \in \mathcal{B},\ \sum_{i=1}^{p} \theta_i^\infty\ x_i + \theta_0^\infty < 0 \quad . \tag{4.5}$$

D'un point de vue géométrique, à l'instant t, le vecteur des paramètres θ^t définit un hyperplan $H = \{\mathbf{x} \mid \sum_{i=1}^{p} \theta_i^t\ x_i^t + \theta_0^t = 0\}$ qui divise l'espace des entrées $\mathbb{R}^p$ en deux demi-espaces. Notons que lorsque θ_0^t est nul, l'hyperplan trouvé passe par l'origine.

Un peu plus tard, les deux mathématiciens Minsky et Papert ((Minsky et Papert, 1969)) ont montré que les limites théoriques du *perceptron simple* incapable de séparer deux ensembles non-linéairement séparables constituent un grave handicap. La démonstration est illustrée par le célèbre exemple du « ou-exclusif » (*XOR*) (voir figure 4.7), $\{f\ :\ \{-1,1\}^2 \mapsto \{-1,1\}$ avec $f(-1,-1) = -1\ ;\ f(-1,1) = 1\ ;\ f(1,-1) = 1\ ;\ f(1,1) = -1\}$, incapable d'être modélisé par le *perceptron simple*. La solution naturelle pour dépasser les limites imposées au *perceptron simple* est très vite apparue. Il s'agit de structurer le réseau en couches en opérant une composition de plusieurs perceptrons simples. Ce réseau est alors appelé *perceptron multicouches*. L'algorithme d'apprentissage de Rosenblatt ne fonctionnait cependant plus, rendant le modèle inutilisable. L'effet fût presque immédiat, chercheurs et investisseurs se désintéressèrent des réseaux de neurones pour se tourner vers l'approche symbolique, qui semblait beaucoup plus prometteuse.

Plus récemment, dans les années 1980, deux équipes indépendantes en France (Lecun et al) et aux États-Unis (Rumelhart et al) ont mis au point l'algorithme de rétro-propagation du gradient (voir chapitre 5), permettant, par un simple calcul de dérivées de fonctions composées, un apprentissage du perceptron multicouches par minimisation d'une fonction d'erreur. Dans le même temps l'*Intelligence Artificielle*, basée sur les concepts de l'approche symbolique, affichait ses limites en matière d'*apprentissage*. Les réseaux de neurones sont revenus sur le devant de la scène avec de nouveaux algorithmes comme les réseaux de Kohonen, capables d'auto-organisation (Kohonen, 1982), et les réseaux de Hopfield (Hopfield, 1982).

4.1.3 Le perceptron multicouches

Comme le montre la figure 4.8, un *perceptron muni d'une couche cachée* de deux neurones est capable de résoudre le problème du *XOR*, problème non soluble par un

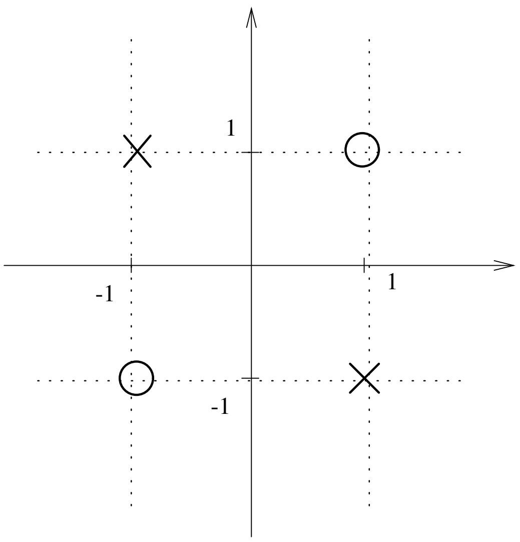


FIG. 4.7 – *La fonction* XOR. *Les ronds correspondent à la valeur '+1' et les croix à la valeur '-1'.*

perceptron simple.

Cette possibilité de séparer des ensembles non-linéairement séparables, plus les qualités, décrites plus haut, attribuées au *perceptron simple*, justifient l'intérêt que suscita le *perceptron multicouches*. Dans ce document, on détaille section 4.2 les propriétés de ces modèles, certains algorithmes d'apprentissages sont décrits chapitre 5, et on aborde les problèmes de détermination d'architectures au chapitre 6.

Le perceptron multicouches décrit figure 4.9, adapté au problème de la régression, comporte p unités en entrée recevant respectivement p variables $\{X_1, X_2, \ldots, X_p\}$, et une seule unité de sortie qui produit la variable Y. Si le réseau dispose de n neurones sur sa couche cachée, on note alors ce réseau $\mathrm{PM}(p, n, 1)$. Un neurone seuil est aussi défini, qui correspond à une entrée constante égale à 1.

Par simplicité, et parce que ce réseau est le plus employé dans la suite de ce document, on considère un réseau avec une sortie scalaire[1] qui est alors complètement déterminée par l'équation :

$$Y = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ij} X_i + \beta_{0j}) + \alpha_0 \tag{4.6}$$

où

– n est le nombre de neurones de la couche cachée.

1. Le modèle du perceptron multicouches comportant une sortie vectorielles ($\forall i = 1, 2, \ldots, p$; $Y \in \mathbb{R}^d$, $d > 1$), est une simple généralisation du modèle présenté ici. Toutes les propriétés et caractéristiques mentionnées dans la suite de ce document, ainsi que tous les résultats théoriques peuvent être généralisés à un perceptron pris dans sa définition la plus large.

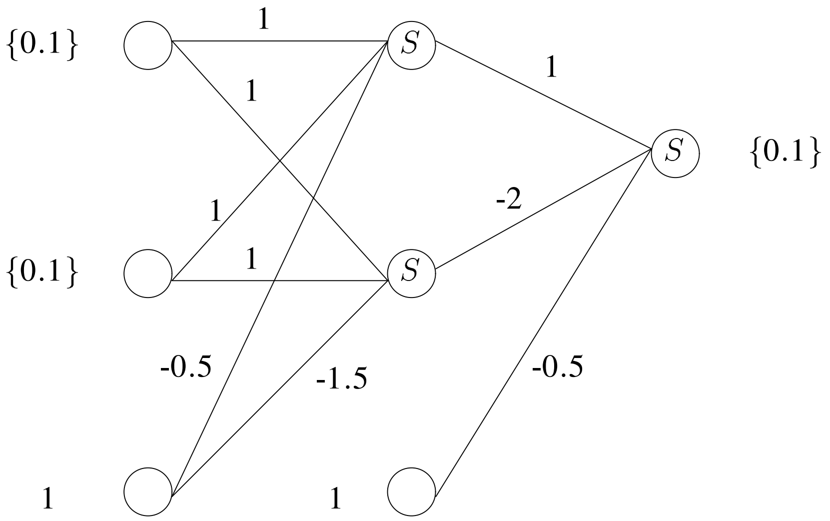


FIG. 4.8 – *Exemple de réseau de neurones du type perceptron multicouches qui résoud le problème du XOR. Tous les neurones sont munis d'une fonction d'activation* seuil S *définie équation 4.1 (voir aussi fig. 4.2).*

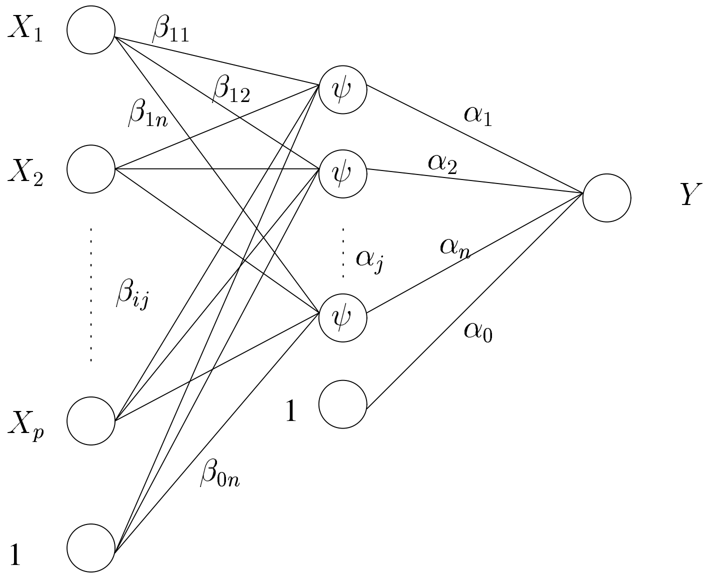


FIG. 4.9 – *Exemple de réseau de neurones du type perceptron multicouches. On utilise les mêmes notations que pour l'équation 4.6.*

- $\theta = \{(\alpha_j)_{0\leq i\leq n}, (\beta_{ij})_{0\leq i\leq p, 1\leq j\leq n}\} \in \mathbb{R}^{n\times(p+2)+1}$ est le vecteur des paramètres, l'ensemble des *poids*, ou encore *connexions* en langage connexioniste. Les constantes $\{\alpha_0, \beta_{01}, \ldots, \beta_{0n}\}$ caractérisent les connexions reliées aux neurones de seuil de la fig. 4.9.

- ψ est une fonction de $\mathbb{R}$ dans $\mathbb{R}$, généralement non linéaire, ni même polynômiale.

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

En ce qui concerne la fonction d'activation associée aux neurones de la couche cachée, on peut utiliser toutes celles citées section 4.1.1, on utilise néanmoins dans le cadre de cette thèse une fonction d'activation de la famille des fonctions sigmoïdes.

L'équation 4.6 ne définit alors rien de plus qu'un modèle de régréssion non-linéaire paramétré par le vecteur θ. On peut remarquer néanmoins, que contrairement à la plupart des modélisations, celle-ci s'exprime aisément sous une forme graphique (telle que la fig. 4.9) permettant une manipulation visuelle et simple des variables à disposition. Cette particularité, qui est loin d'être accessoire, permet aux utilisateurs des réseaux de neurones de redécouvrir visuellement la régression ou la classification, en général plutôt abordées par le biais d'équations.

4.2 Propriétés des perceptrons multicouches

Les perceptrons multicouches ont rapidement suscité l'intérêt de mathématiciens en s'inscrivant dans des cadres généraux classiques (approximation fonctionnelle, processus de Markov, algorithmes adaptatifs,. . .). Ils recèlent néanmoins des difficultés de nature nouvelle en raison de leur caractère foncièrement non linéaire. D'autre part si les simulations, voire les applications opérationnelles, sont souvent très avancées, les questions mathématiques y restent encore beaucoup plus nombreuses que les réponses. Un des points sur lesquels les efforts se sont portés est l'étude de la capacité d'approximation de fonctions des perceptrons multicouches. Il a été ainsi démontré qu'un perceptron multicouches avec une seule couche cachée pourvue d'un nombre suffisant de neurones, peut approximer n'importe quelle fonction continue sur un compact de $\mathbb{R}^p$ avec la précision souhaitée. Néanmoins, cette propriété ne permet pas de choisir, pour un type de fonction donné, le nombre de neurones optimal dans la couche cachée. Autrement dit ce résultat ne mène pas vers une technique de construction d'architecture. Au plus, grâce aux travaux récents d'Attali et de Pagès, on dispose d'une borne supérieure du nombre de neurones à employer sur la couche cachée, pour une certaine classe de perceptrons multicouches.

Dans cette section, on précise cette propriété, ainsi que d'autres, moins déterminantes, mais ayant tout de même contribué au succès du perceptron multicouches.

4.2.1 Le Perceptron Multicouches : un approximateur universel

Bien que récent, le problème de l'approximation de fonctions par des perceptrons multicouches a déjà suscité une abondante littérature. Par exemple, (Cybenko, 1989) (Funahashi, 1989) (Barron, 1993) (Hornik et al., 1989)) ont étudié la propriété d'approximation de fonctions continues sur un compact par des perceptrons multicouches à une seule couche cachée, munis de fonction d'activation sigmoïde. On reprend ici, les résultats de (Hornik et al., 1989) étendus par la suite par (Leshno et al., 1993) aux perceptrons multicouches munis de fonctions d'activation quelconques non-linéaires, non-polynômiales.

Théorème 4.2 :

(Hornik et al., 1989) Reprenons le cadre et les notations du modèle de perceptron multicouches défini équation 4.6, avec $\psi(.)$ une fonction d'activation strictement croissante et bornée. Soit K un compact de $\mathbb{R}^p$. 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 paramètre $\theta = \{(\alpha_i)_{0\leq i\leq n}\ ,\ (\beta_{ij})_{0\leq i\leq p\ ,\ 1\leq j\leq n}\} \in \mathbb{R}^{n\times(p+2)+1}$ tels que, $\forall\ \{X_1, X_2, \ldots, X_p\} \in K$:

$$| \, f(X_1, X_2, \ldots, X_p) - \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ij} X_i + \beta_{0j}) + \alpha_0 \, | < \varepsilon$$

La démonstration s'appuie sur le théorème de Stone-Weierstrass. Rappelons qu'elle ne fournit aucune information concernant le dimensionnement optimal ou minimal d'un perceptron approximant une fonction donnée.

Différents travaux ont suivi, portant sur des fonctions d'activations aux propriétés diverses (White, 1992), (Ito, 1991; Ito, 1992), ou sur la convergence en norme $L_p(\mu)$, μ mesure finie sur K (Leshno et al., 1993), (Ito, 1992). Signalons aussi les travaux de J.G. Attali et G. Pagès (Attali et Pagès, 1995a), qui fournissent une preuve élégante et simple de cette propriété d'approximation universelle, basée sur des développements de Taylor-Young, et sur le déterminant de Vandermonde. La démonstration fonctionne pour une approximation uniforme et conduit naturellement à des bornes, malheureusement assez grandes, sur le nombre de neurones à utiliser dans la couche cachée du perceptron multicouches.

4.2.1.1 Vitesse de convergence de l'approximation

On dispose de très peu de résultats sur la vitesse d'approximation d'une fonction de régularité donnée, en fonction du nombre de paramètres (connexions) ou du nombre n de neurones sur la couche cachée. Nous reprenons néanmoins les résultats (Attali et Pagès, 1995a) et ceux de (Barron, 1993). Citons aussi les travaux de Roynette (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 4.3 :

(Attali et Pagès, 1995a) Soit K un sous-ensemble compact de $\mathbb{R}^p$. On pose $M_K = \sup_{x \in K} \| x \|$ et $\delta_K = \sup_{(x,y) \in K^2} \| x - y \|$.

Soit $\psi \in C^\infty(\mathbb{R}, \mathbb{R})$ une fonction non-polynômiale telle que $\forall k \in \mathbb{N},\ \psi^{(k)} \neq 0$. Soit f une fonction quelconque dont toutes les dérivées jusqu'à l'ordre p appartiennent à $C(K, \mathbb{R})$ et telle que $\forall i,\ 1 \leq i \leq k, \quad \partial f^{(k)} / \partial x_i$ soit ρ-lipschitz.
Soit $(\varepsilon_n)_{n>0}$, une suite de valeurs strictement positives, avec $\lim\limits_{n \longrightarrow \infty} \varepsilon_n = 0$, alors il existe une suite $(g_n)_{n \geq 0}$ de $PM(p, n, 1)$ munis de fonctions d'activation ψ associées aux n neurones de la couche cachée telle que :

$$\| f - g_n \| \leq \rho A_p M_K^{p+1} \frac{(1 + \varepsilon_n)}{n^{p+1}} \quad . \tag{4.7}$$

où A_p est une constante qui dépend uniquement de p.

On remarque néanmoins 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 dès lors que la dimension p des entrées est grande. On vérifie aisément que ce résultat s'applique lorsque ψ, la fonction d'activation des unités de la couche cachée, est la fonction logistique de la classe des fonctions sigmoïdes.

On donne dans la suite des résultats de Barron, plus intéressants en terme de vitesse d'approximation, mais s'appliquant à une classe de fonctions très particulière. On suppose ici que l'on tente d'approximer la fonction f à l'aide d'une base finie de réalisations comprenant T individus $(x_i, f(x_i)_{i=1,2,\ldots,T})$.

Définition 4.4 :

Soit $f : \mathbb{R}^p \mapsto \mathbb{R}$ *une fonction, et* $\int_{\mathbb{R}^p} e^{i\omega^t x}\tilde{f}(\omega)d\omega = f(x)$ *sa représentation de Fourier. Posons* $\mid \omega \mid_1 = \sum_{j=1}^{p} \mid \omega_j \mid$ *le norme* ℓ_1 *de* ω *sur* $\mathbb{R}^p$*. On définit* C_f *par :*

$$C_f = \int_{\mathbb{R}^p} \mid \omega \mid \mid \tilde{f}(\omega) \mid d\omega \tag{4.8}$$

Théorème 4.5 :

(Barron, 1993) Soit K *un sous-ensemble compact de* $\mathbb{R}^p$ *et* $f : K \mapsto \mathbb{R}$ *une fonction continue sur* K*. Soit* $\hat{f}_n$ *son estimateur issu de la classe des perceptrons multicouches (équation 4.6) comportant* n *neurones sur sa couche cachée. Soit* C_f *le critère de complexité de* f *défini équation 4.8 et* T *la taille de l'échantillon sur lequel on effectue l'estimation de* f*. Alors*

$$E\left(\| f - \hat{f}_{n,T} \|^2\right) \leq O\left(\frac{C_f^2}{n}\right) + O\left(\frac{np}{T}\log T\right) \quad . \tag{4.9}$$

La démonstration, que nous ne développerons pas ici, s'appuie sur des techniques de Fourier. Le critère de complexité C_f s'avère néanmoins délicat à manipuler, et il est difficile d'exprimer les propriétés classiques tels que continuité ou dérivabilité en utilisant ce critère. Signalons cependant que pour une certaine classe de fonctions définie plus en détail dans (Barron, 1993), la valeur de C_f croît linéairement avec la dimension p. Ceci entraîne une vitesse de convergence qui croît aussi de manière linéaire en fonction de p (à n et T fixés), alors que celle-ci croît exponentiellement pour des approximations du type polynômiale ou à base de fonctions « ondelettes ».

On notera aussi que la borne définie équation 4.9 s'apparente à un critère du type *Akaike* (voir chapitre 6, section 6.2.2 pour la définition), en tenant compte des deux aspects contradictoires[2] d'une modélisation non-linéaire :

- *Minimiser l'erreur d'approximation*, qui requiert 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 important.

2. On pourra se reporter au chapitre 5 concernant l'estimation des paramètres pour plus de détails.

4.2.2 Autres propriétés du perceptron multicouches

Quoique moins intéressantes pour la modélisation paramétrique, signalons d'autres particularités inhérentes aux perceptrons multicouches. Les deux premières ont, avec la propriété d'approximation universelle, contribué au succès des perceptrons multicouches ; les deux suivantes sont plus de l'ordre de la remarque.

– **Robustesse à la détérioration**. Cela provient de la répartition et de la redondance de l'information au sein du réseau. Si la couche cachée du réseau est munie de suffisamment de neurones, supprimer une connexion[3] peut avoir un effet minime dans le calcul de la sortie. Les calculs étant menés par plusieurs neurones, il y a différents chemins de l'entrée vers la sortie dans le graphe orienté que constitue l'architecture d'un perceptron multicouches. Pour cette raison, on parle de robustesse à la détérioration du modèle. Peu de travaux étudient cette robustesse, et peu d'applications en tirent profit.

– **Résistance aux variables d'entrées aberrantes**. Dans le cas linéaire, une entrée aberrante signifie automatiquement une prévision aberrante, la valeur de la sortie étant linéairement proportionnelle à celle de l'entrée. Du fait des filtres saturants que constituent les fonctions d'activation sigmoïdes, le réseau limite ce genre d'inconvénient. Si on reprend les notations de l'équation 4.6, la sortie Y est bornée par $\sum_{j=0}^{n} \mid \alpha_j \mid$.

– **Non-unicité du modèle par rapport aux paramètres**. On peut trouver deux jeux de poids différents (deux ensembles de paramètres différents) qui génèrent les mêmes sorties. Ceci peut être obtenu par exemple en permutant l'ordre des neurones de la couche cachée.

– **Échelle des données en entrée et sortie**. L'échelle des données n'a théoriquement pas d'influence sur l'efficacité de la modélisation. Ainsi, soit un modèle correspondant à celui de l'équation 4.6. Supposons que l'on opère une homotéthie et une translation sur les entrées $(X_i)_{i=1,2,\ldots,p}$ et la sortie Y :

  – $Y' = Y/a + b$, $a \in \mathbb{R}^*$, $b \in \mathbb{R}$,
  – $X'_i = X_i/c_i + d_i$, $c_i \in \mathbb{R}^*$, $d_i \in \mathbb{R}$, pour $i = 1, 2, \ldots, p$,

  Alors, en prenant le vecteur des paramètres $\theta' = (a \times \alpha_j, \alpha_0 - b, c_j \times \beta_{ij}, \beta_{0j} - d_j)_{1 \leq i \leq p, 1 \leq j \leq n}$, on obtient un modèle identique à celui décrit équation 4.6.

  On verra néanmoins que lors de l'estimation des paramètres, il est préférable de normaliser les entrées et les sorties, afin d'accélérer la recherche du meilleur estimateur.

Citons enfin un résultat propre aux perceptrons multicouches munis de fonction d'activation sigmoïdes. Ces réseaux possèdent en effet des propriétés de dérivabilité et de contraction qui sont utiles section 5.1.1.

3. Cela revient à annuler le coefficient relatif à cette connexion dans l'équation 4.6.

Proposition 4.6 :

> *Soit une fonction f_θ définie de $\mathbb{R}^p$ dans $\mathbb{R}$ appartenant à la classe des perceptrons multicouches définie équation 4.6 . Alors f_θ est lipschitzienne et la dérivée de f_θ d'ordre m ($m \geq 1$ quelconque), existe et est lipschitzienne.*

Preuve. D'après la définition de f_θ (équation 4.6) f_θ est une combinaison linéaire de fonctions de la famille des sigmoïdes. Le lemme 4.7 suffit alors pour entraîner le proposition. ∎

Lemme 4.7 :

> *Soit une fonction σ définie de $\mathbb{R}$ dans $\mathbb{R}$ appartenant à la famille des fonctions sigmoïdes définie équation 4.2. Alors σ est lipschitzienne et la dérivée de σ d'ordre m, $m \geq 1$ quelconque, existe et est lipschitzienne.*

Preuve. De par la définition d'une fonction sigmoïde (4.2), on a

$$\sigma'_{c,k,r}(x) = \frac{\partial \sigma_{c,k,r}(x)}{\partial x} = 2ck\frac{e^{kx}}{(e^{kx}+1)^2} \tag{4.10}$$

$$= \frac{k}{2c}\left(c^2 - (\sigma_{c,k,r}(x) - r)^2\right) \quad . \tag{4.11}$$

On a aussi (avec l'équation 4.10), $\sigma''(x) = 2ck^2e^{kx}(1-e^{kx})/(e^{kx}+1)^3$ avec $\sigma''(x) > 0$, si $x < 0$, et $\sigma''(x) \leq 0$, si $x \geq 0$.

On en déduit que $\forall x$, $\sigma'(x) \leq \sigma'(0) = (ck)/2$, ce qui implique que $\sigma_{c,k,r}$ est $(ck)/2$-lipschitzienne.

Puisque $\sigma'_{c,k,r}(x)$ peut aussi s'écrire sous la forme d'un polynôme en $\sigma_{c,k,r}(x)$ (éq 4.11), avec $\sigma_{c,k,r}(x)$ bornée pour tout $x \in \mathbb{R}$, la dérivée de $\sigma_{c,k,r}(x)$ d'ordre m, pour un m quelconque, existe et est bornée. Puisque, pour tout $m > 1$, la dérivée de $\sigma_{c,k,r}(x)$ d'ordre $m+1$ est bornée pour tout $x \in \mathbb{R}$, on en déduit que la dérivée de $\sigma_{c,k,r}(x)$ d'ordre m est lipschitzienne. ∎

Utiliser un réseau de neurones pour résoudre un problème particulier, signifie chercher une solution sous la forme d'une fonction de la classe associée. Cette solution est alors approchée, généralement de manière itérative, en découvrant l'architecture la plus adéquate (identification du modèle), et en calculant itérativement les poids des connexions qui constituent les paramètres du modèle neuronal (estimation des paramètres).

4.3 Le modèle paramétrique NAR$_n(p)$ basé sur le perceptron multicouches

Un modèle autorégressif linéaire correspond assez naturellement à l'idée de régression linéaire à chaque instant sur l'espace des observations passées. Ce type de modèle peut néanmoins s'avérer être insuffisant pour décrire certains processus pour lesquels la relation entre la variable que l'on tente de modéliser à un instant donné et les variables passées n'est pas linéaire. Il peut alors être intéressant d'utiliser une autorégression basée sur le perceptron multicouches, et adaptée au phénomène étudié. Puisqu'un perceptron multicouches munie d'une seule couche cachée possède la propriété d'approximation universelle énoncée dans le théorème 4.2, on étudie dans la suite un modèle basé sur ce type de perceptron multicouches. Ces idées conduisent à l'extension non-linéaire suivante du modèle autorégressif classique.

Définition 4.8 :

Reprenons le cadre et les notations du modèle de perceptron multicouches défini équation 4.6. Soient deux entiers $p, n \geq 1$. Un processus autorégressif fonctionnel sur $\mathbb{R}^p$, dont la fonction associée est de la classe des perceptrons multicouches à une couche cachée décrit 4.6 munis de fonctions de transfert σ sigmoïdes, est une suite $(X_t)_{t\in\mathbb{Z}}$ de variables aléatoires à valeurs dans $\mathbb{R}$ vérifiant :

$$X_t = \sum_{j=1}^{n} \alpha_j \sigma(\sum_{i=1}^{p} \beta_{ij} X_{t-i} + \beta_{0j}) + \alpha_0 + \varepsilon_t, \tag{4.12}$$

où

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

Nous appelons par la suite ce modèle NAR$_n(p)$ (« Neural Autoregression ») et nous notons par $X^{(p)} = (X_t^{(p)})_{t\in\mathbb{Z}}$ le processus vectorisé *associé, défini par $X_t^{(p)} := (X_t, \ldots, X_{t-p+1})$.*

Lorsque $p > 1$, (X_t) est clairement non markovien ; par contre le processus vectorisé $(X_t^{(p)})$ est une chaîne de Markov à valeurs dans $\mathbb{R}^p$.

On se retrouve donc dans le cadre défini du modèle ARF$_1(p)$ (autorégressif fonctionnel) décrit chapitre 3.

De même que pour les modèles linéaires, on peut définir des modèles[4] incluant d'autres variables aléatoires, dites variables explicatives ou exogènes, et/ou certains re-

4. On ne représente en schéma que les modèle NAR$_n(p)$ (fig. 4.10) et NARX$_n(p, m)$ (fig. 4.11).

tards du bruit (ε_t) :

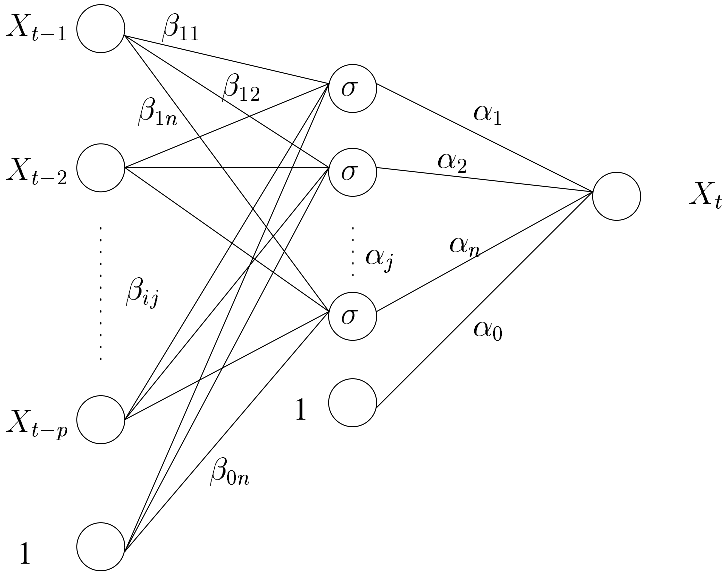


FIG. 4.10 – *Le modèle $NAR_n(p)$. On utilise les mêmes notations que pour l'équation 4.12.*

– Le modèle autorégressif neuronal avec variables exogènes ($\mathrm{NARX}_n(p,m)$) :

$$X_t = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ij} X_{t-i} + \sum_{l=1}^{m} \beta'_{lj} Y_t^l + \beta_{0j}) + \alpha_0 + \varepsilon_t, \tag{4.13}$$

où les $(Y_t^l)_{1<l<m,t\in\mathbb{Z}}$ sont des variables explicatives.

– Le modèle autorégressif neuronal avec moyennes mobiles ($\mathrm{NARMA}_n(p,q)$) :

$$X_t = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ij} X_{t-i} + \sum_{k=1}^{q} \beta''_{kj} \varepsilon_{t-k} + \beta_{0j}) + \alpha_0 + \varepsilon_t, \tag{4.14}$$

où les $(\varepsilon_{t-k})_{k=1,2,\ldots,q}$ sont les q résidus passés.

– Le modèle autorégressif neuronal avec moyennes mobiles et variables exogènes ($\mathrm{NARMAX}_n(p,q,m)$) :

$$X_t = \sum_{j=1}^{n} \alpha_j \psi(\sum_{i=1}^{p} \beta_{ij} X_{t-i} + \sum_{l=1}^{m} \beta'_{lj} Y_t^l + \sum_{k=1}^{q} \beta''_{kj} \varepsilon_{t-k} + \beta_{0j}) + \alpha_0 + \varepsilon_t, \tag{4.15}$$

Il est à noter que le modèle $\mathrm{NAR}_n(p)$ et ses variantes ne peuvent pas prendre en considération des processus incluant des tendances. En effet, puisque les fonctions d'activation sont des fonctions sigmoïdes bornées, la modélisation de processus pouvant sortir de n'importe quel compact n'a pas de sens. Aucun résultat, à notre connaissance, n'a été obtenu sur l'incidence de la saisonnalité sur la modélisation neuronale d'une série temporelle.

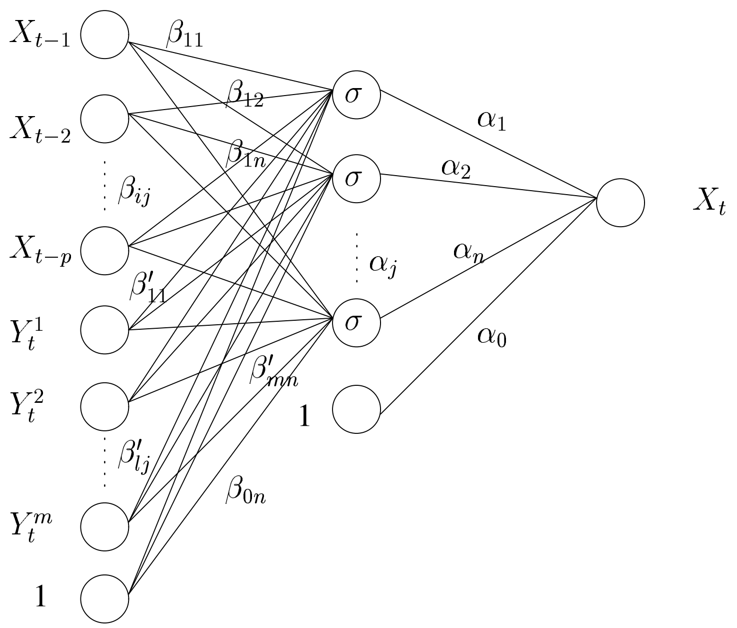


FIG. 4.11 – *Le modèle NARX*$_n(p, m)$. *On utilise les mêmes notations que pour l'équation 4.13.*

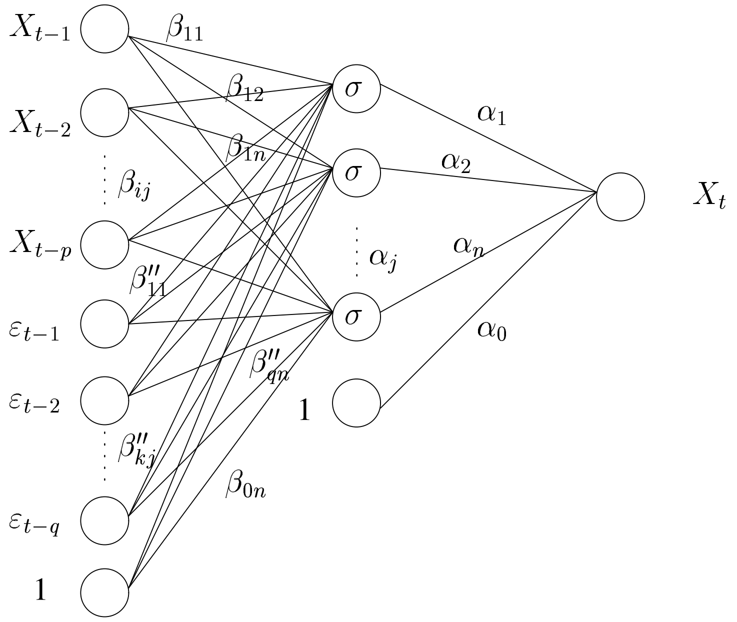


FIG. 4.12 – *Le modèle NARMA*$_n(p, q)$. *On utilise les mêmes notations que pour l'équation 4.14.*

4.3.1 Correspondances avec les modèles ARMA et dérivés

Dans les modèles AR(p) (Box et Jenkins, 1976), on suppose qu'à l'instant t, X_t est fonction linéaire de ses valeurs passées, et d'une perturbation aléatoire ε_t. Si $(a_0, a_1, a_2, \ldots, a_p)$ sont les $p+1$ réels associés au modèle, X_t est alors de la forme :

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + a_0 + \varepsilon_t \quad , \quad t \in \mathbb{Z} \quad . \tag{4.16}$$

Cette classe de modèles permet de représenter à l'aide d'un nombre de paramètres relativement limité, une gamme de phénomènes très variée. En outre, il est possible de proposer des méthodes, dites méthodes d'identification, permettant de choisir dans cet ensemble de modèles celui qui semble le mieux adapté aux données dont on dispose (voir (Box et Jenkins, 1976)). Une fois ce modèle choisi, on peut en estimer les paramètres et déterminer les prévisions optimales à l'horizon k (prévision de X_{t+k}, k entier positif. On se reportera au chapitre 2, définitions 2.2 et 2.3, pour la description des modèles AR(p) et ARMA(p, q).

4.3.1.1 Correspondances

Il est clair que si l'on remplace les fonctions d'activation sigmoïdales des neurones par des fonctions linéaires, le modèle neuronal devient exactement un modèle AR classique (surparamétré s'il y a des couches cachées).

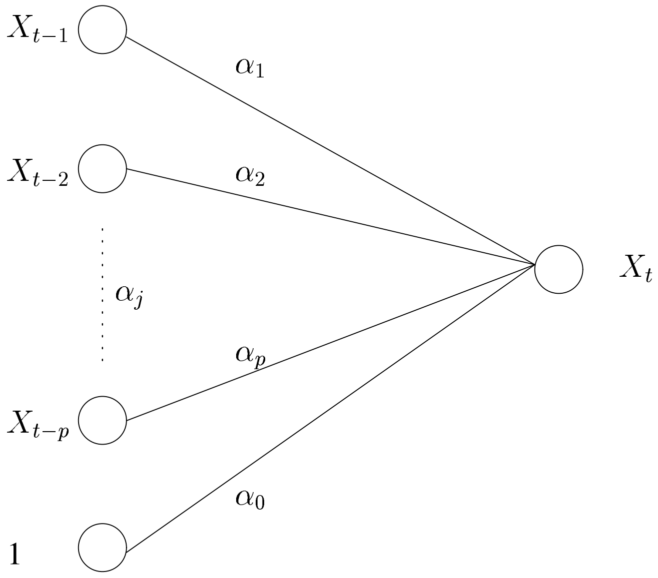


FIG. 4.13 – *Le modèle $AR_n(p, q)$. On utilise les mêmes notations que pour l'équation 4.16.*

Si de plus, on réinjecte les erreurs de prévision, on peut reproduire exactement les fonctionnalités d'un processus ARMA.

En fait, les fonctions sigmoïdes couramment employées telles que la fonction logistique ou la fonction tangente hyperbolique peuvent être approchées par une fonction linéaire au voisinage de zéro.

Comme on essaye d'ajuster au mieux les évolutions d'une série chronologique, on est en droit de penser que si le modèle linéaire est le modèle le plus performant, l'apprentissage du réseau neuronal l'amènera à ne travailler que dans la partie linéaire des fonctions sigmoïdes. On précise cette idée dans le lemme suivant.

Lemme 4.9 :

Soit T et n, deux entiers non nuls et soit $\phi = (\phi_0, \phi_1, \ldots, \phi_p)$ un vecteur de p réels. Alors pour toute suite $(X_t)_{t=1,2,\ldots,T}$ extraite d'un processus autorégressif linéaire $AR(p)$ (voir définition 2.2) de vecteur de paramètre ϕ et de vecteur initial $X_0^{(p)}$, et pour tout $\eta > 0$, il existe un processus $NAR_n(p)$ (voir définition 4.8) noté $(Y_t)_{t\geq 1-p}$, de vecteur de paramètres $\theta = \{(\alpha_i)_{0\leq i\leq n}\ ,\ (\beta_{ij})_{1\leq i\leq n\ ,\ 0\leq j\leq p}\} \in \mathbb{R}^{n\times(p+2)+1}$ et de même vecteur initial $X_0^{(p)}$, tel que :

$$\textit{pour tout} \quad t \in \{1, 2, \ldots, T\}, \quad \mid X_t - Y_t \mid < \eta$$

Preuve. On démontre dans la suite le cas où $p = 1$, $n = 1$, en considérant un processus AR(1) et un modèle $\mathrm{NAR}_1(1)$ avec une seule entrée et une seule unité cachée. La démonstration peut alors être étendue au cas où n et p sont des entiers strictement positifs quelconques.

La preuve consiste donc à montrer que pour $\varepsilon > 0$, pour tout $x \in K$, K compact de $\mathbb{R}$, et pour tout $\gamma \in \mathbb{R}$, il existe $\alpha, \beta \in \mathbb{R}$ tels que

$$\mid \alpha \tanh(\beta x) - \gamma x \mid < \varepsilon \tag{4.17}$$

De par les définitions des processus $\mathrm{NAR}_1(1)$ et AR(1), et en remarquant que, puisque T est fini, $\{\mid X_t \mid, t = 1, 2, \ldots, T\}$ est borné, la démonstration du lemme 4.9 est alors obtenue directement pour $p = 1$ et $n = 1$.

On sait que le développement limité de $x \mapsto \tanh(x)$ dans un voisinage de 0 est :

$$\tanh(x) = x + O(x^3)$$

Il existe donc une constante C positive et un voisinage S de 0, tels que pour tout $x \in S$,

$$\mid \tanh(x) - x \mid \leq C \mid x^3 \mid$$

D'autre part, quelque soit le compact K choisi, il existe β non nul tel que $\beta K = \{\beta x, \quad x \in K\} \subset S$. On peut alors écrire, pour tout $x \in K$, et pour tous α et γ :

$$\begin{aligned} \mid \alpha \tanh(\beta x) - \gamma x \mid &= \mid \alpha \tanh(\beta x) - \alpha\beta x + \alpha\beta x - \gamma x \mid \\ &\leq \mid \alpha\beta x - \gamma x \mid + C|\alpha| \mid \beta x \mid^3 \end{aligned} \tag{4.18}$$

Posons $A = \mid \alpha\beta x - \gamma x \mid$. On peut prendre α tel que $\alpha\beta = \gamma$ ($\alpha = \gamma/\beta$). Pour tout $x \in K$, on a alors $A = 0$, et l'inégalité 4.18 peut s'écrire :

$$\mid \alpha \tanh(\beta x) - \gamma x \mid \leq C|\gamma|\beta^2|x|^3 \tag{4.19}$$

En posant $M = \max_{x \in K} \mid x \mid$ on a alors

$$\mid \alpha \tanh(\beta x) - \gamma x \mid \leq C|\gamma|\beta^2 M^3 \quad . \tag{4.20}$$

En choisissant β suffisamment petit pour que $C|\gamma|\beta^2 M^3 < \varepsilon$ on obtient l'inégalité 4.17. ∎

Notons toutefois que les correspondances entre les modèles ARMA et neuronaux n'ont de sens que pour la prévision à un pas. A un horizon plus lointain, les mécanismes sont sensiblement différents pour des raisons de non-linéarité, et les comportements des modèles ne peuvent pas être équivalents.

4.3.2 Propriétés probabilistes du processus NAR$_n(p)$

On reprend dans ce chapitre les conditions pour que le modèle NAR$_n(p)$ possède des propriétés de stabilité, d'ergodicité et de mélangeance. La propriété de stabilité est utile pour l'obtention de résultats sur la consistance et la normalité asymptotique des estimateurs des moindres carrés des paramètres de ce modèle. Les propriétés d'ergodicités et de $beta$-mélangeance géométriques ne seront pas exploité

4.3.2.1 Stabilité du processus NAR$_n(p)$

On considère dans cette section le modèle du type perceptron (défini en 4.6), dont les fonctions de transfert associées aux neurones de la couche cachée sont des sigmoïdes (définies équation 4.2). L'objectif ici est d'établir un cadre statistique dans lequel le modèle est stable. Dans ce but, on reprend les théorèmes détaillés chapitre 3, dans le cadre plus général des modèles autorégressifs fonctionnels quelconques (ARF$_d(p)$), et on les applique au modèle $ARF_1(p)$ que constitue un modèle NAR$_n(p)$.

Théorème 4.10 :

Soit le modèle $NAR_n(p)$ de la définition 4.8, avec

$$f_\theta(X_{t-1}^{(p)}) = \sum_{j=1}^{n} \alpha_j \sigma_{c,k,r}(\sum_{i=1}^{p} \beta_{ij} X_{t-i} + \beta_{0j}) + \alpha_0$$

et $\sigma_{c,k,r}$ fonction du type sigmoïde (définie en 4.2). On note θ_0 le vrai paramètre $\theta = \{(\alpha_i^*)_{0 \leq i \leq n}\ ,\ (\beta_{ij}^*)_{0 \leq i \leq p\ ,\ 1 \leq j \leq n}\}$. Supposons que, pour $a \geq 1$, l'un ou l'autre ensemble de conditions suivantes soit satisfait :

$$[\mathbf{C.1}] \quad \begin{cases} (a) \text{ le bruit } (\varepsilon_t) \text{ a un moment d'ordre } a\,; \\ (b) \displaystyle\sum_{j=1}^{n}\sum_{i=1}^{p} |\alpha_j^* \beta_{ij}^*| < \frac{2}{ck} \quad ; \end{cases}$$

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

Alors le modèle $NAR_n(p)$ est stable.

Preuve. Le modèle $\mathrm{NAR}_n(p)$ est en fait un modèle $ARF_1(p)$ particulier. La démonstration consiste donc à vérifier que les ensembles de conditions **[C.1]** et **[C.2]** impliquent respectivement les ensembles **[f.1]** et **[f.2]** du théorème 3.1 (avec $d = 1$) pour le vrai modèle en θ_0.

Les conditions relatives aux bruits de **[f.1]** et **[f.2]** sont reprises en *(a)*, *(c)* et *(d)*. En reprenant la définition de $\sigma_{c,k,r}$ (équation 4.2), on voit aussi que $\sigma_{c,k,r}$ est bornée par $|c| + |r|$. Donc pour un vecteur de paramètres θ fixé, $f_\theta(.)$ est bornée, ce qui implique que les conditions **[f.2]** sont toutes satisfaites. On sait de plus, en raison de la preuve du lemme 4.7, que $\sigma_{c,k,r}$ est $(ck)/2$-lipschitzienne. Il suffit donc d'avoir *(b)*, pour que la seconde condition de **[f.1]**, imposant que f soit contractante, soit remplie. ∎

4.3.2.2 Ergodicité et β-mélangeance du processus $\mathrm{NAR}_n(p)$

Reprenons le cadre et les définitions de la section 3.3 qui décrivent les propriétés d'ergodicité et de β-mélangeance géométriques pour un processus $\mathrm{ARF}_d(p)$. On est ici dans le cas $d = 1$.

Théorème 4.11 :

Soit le modèle $NAR_n(p)$ de la définition 4.8, défini pour $t \geq 1-p$, de vecteur initial $X_0^{(p)}$, et tel que

$$f(X_{t-1}^{(p)}, \theta) = \sum_{j=1}^{n} \alpha_j \sigma_{c,k,r}(\sum_{i=1}^{p} \beta_{ij} X_{t-i} + \beta_{0j}) + \alpha_0$$

et $\sigma_{c,k,r}$ fonction du type sigmoïde (définie en 4.2). On note θ_0 le vrai paramètre $\theta_0 = (\alpha_j^*, \beta_{ij}^*)_{0 \leq i \leq p, 0 \leq j \leq n}$. Supposons que, pour $a \geq 1$, l'ensemble de conditions **[C.2]** *du théorème 4.10 soit satisfait, alors le processus $(X_t)_{t \geq 1-p}$ est géométriquement ergodique, quelle que soit la loi initiale de $X_0^{(p)}$. Si de plus cette loi initiale est μ_{θ_0}, la loi invariante du processus, alors le modèle est aussi géométriquement β-mélangeant.*

Preuve. D'après le théorème 3.7, il suffit de remplir les conditions du système **[f.2]** avec $d = 1$ pour que le modèle soit géométriquement ergodique. Or, on a déjà vu dans la preuve du théorème 4.10 que le système de conditions **[C.2]** implique le système de conditions **[f.2]**. La deuxième partie du théorème est issue de la remarque 3.8 et de la fin du théorème 3.7, qui indiquent que si la loi de vecteur initial $X_0^{(p)}$ est la loi invariante de la chaîne de Markov $(X_t^{(p)})_{t \geq 0}$, la chaîne est stationnaire, et le modèle est géométriquement β-mélangeant. ∎

4.3.3 Problème de la prévision d'ordre supérieur à 1

Dans le cadre linéaire, on démontre que la meilleure prévision à k pas, $k \geq 1$, d'une série linéaire quelconque $(X_t)_{t>-p}$ par un modèle AR (équation 4.16) est calculée par récurrence comme une prévision à un pas, où on remplace les réalisations précédentes par leurs prévisions. Soit un processus autorégressif dont l'équation de récurrence s'écrit (on reprend les notations de l'équation 4.16) $X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \phi_0 + \varepsilon_t$, on a :

$$\begin{aligned}
\widehat{X}_t &= E(X_t \mid X_{t-1}, X_{t-2}, \ldots, X_{t-p}) \\
\widehat{X}_{t+1} &= E(X_{t+1} \mid X_{t-1}, X_{t-2}, \ldots, X_{t-p}) \\
&= E(\sum_{i=1}^{p} \phi_i X_{t-i+1} + \phi_0 + \varepsilon_t \mid X_{t-1}, X_{t-2}, \ldots, X_{t-p}) \\
&= \phi_1 \; E(X_t \mid X_{t-1}, X_{t-2}, \ldots, X_{t-p}) + \sum_{i=2}^{p} \phi_i X_{t-i+1} + \phi_0 + E(\varepsilon_t) \\
&= \phi_1 \widehat{X}_t + \sum_{i=2}^{p} \phi_i X_{t-i+1} + \phi_0 \text{ puisque } \varepsilon_t \text{ est centré}
\end{aligned}$$

Ainsi, en posant f_{AR} la fonction relative à un modèle autorégressif : $f_{\text{AR}}(x_0, x_1, \ldots, x_p) = \sum_{i=0}^{p} \phi_i x_i$. on calcule de façon optimale, la prévision de $X_{t+k}, k \geq 0$ de la manière suivante :

$$\begin{array}{rcl} \widehat{X}_t & = & f_{\text{AR}}(X_{t-1}, X_{t-2}, \ldots, X_{t-p}) \\ \widehat{X}_{t+1} & = & f_{\text{AR}}(\widehat{X}_t, X_{t-1}, \ldots, X_{t-p+1}) \\ & \vdots & \\ \widehat{X}_{t+k} & = & f_{\text{AR}}(\widehat{X}_{t+k-1}, \widehat{X}_{t+k-2}, \ldots, \widehat{X}_{t+k-p}) \;\; \text{si} \;\; k > p \end{array} \tag{4.21}$$

Ce processus, itéré un nombre infini de fois, entraîne une convergence de la série $(\widehat{X}_t)_{t\geq 0}$ ainsi engendrée[5] vers sa moyenne. Dans le cadre non-linéaire relatif aux perceptrons multicouches, si on procède de manière similaire en remplaçant dans le système d'équation 4.21 la fonction f_{AR} par une fonction de la classe des perceptron multicouches (voir fig. 4.14), cette propriété n'est plus vérifiée. On se contente ici de soulever et d'expliciter le problème de la prévision à plus d'un pas dans le cadre non-linéaire relatif aux perceptrons multicouches, sans essayer d'y apporter de réponse.

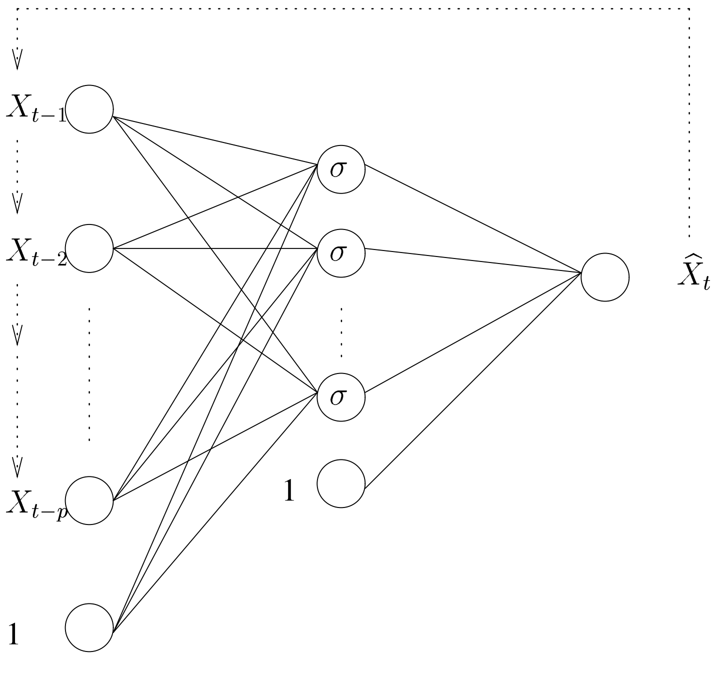


FIG. 4.14 – *Réinjection de la sortie vers l'entrée.*

Ainsi, dès lors que l'on réinjecte en entrée la sortie d'un perceptron multicouches, la fonction récurrente non-linéaire associée peut posséder un ou plusieurs points d'attrac-

5. On suppose alors, conformément à l'équation 4.16, que le polynôme relatif au modèle AR, qui génère les prévisions, a des racines contenues dans le cercle unité.

tion et/ou un ou plusieurs points de répulsion. La valeur limite de la série ainsi générée dépend alors du point de départ de la prévision à k pas (départ en fait de la récurrence).

Prenons par exemple la fonction relative à un perceptron multicouches muni de deux neurones sur la couche cachée (voir fig. 4.15) :

$$\begin{array}{rrcl} f & : & \mathbb{R} \mapsto & \mathbb{R} \\ & & x \mapsto & 2\tanh(-6x) + 3\tanh(3x) \end{array} \tag{4.22}$$

On peut calculer alors les points selles vérifiant $f(x) = x$ (la figure 4.16 en donne une idée approximative). Cette fonction possède 2 points d'attraction[6] ($a_1 = -0,9837$ et, puisque cette fonction est impaire $a_2 = 0,9837$) et 3 points de répulsion ($r_1 = -0,3215$, $r_2 = 0$ et $r_3 = 0,3215$).

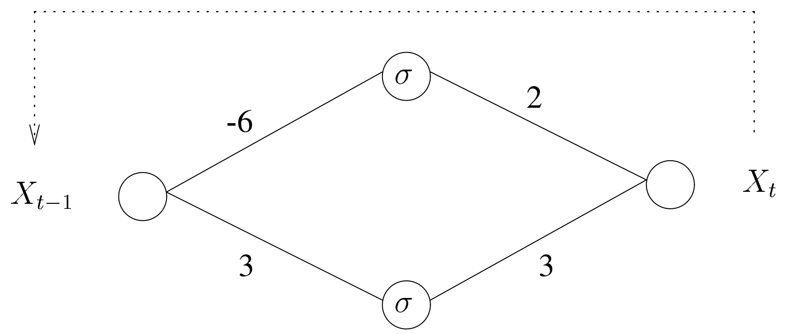


FIG. 4.15 – *Représentation graphique de la fonction f définie équation 4.22.*

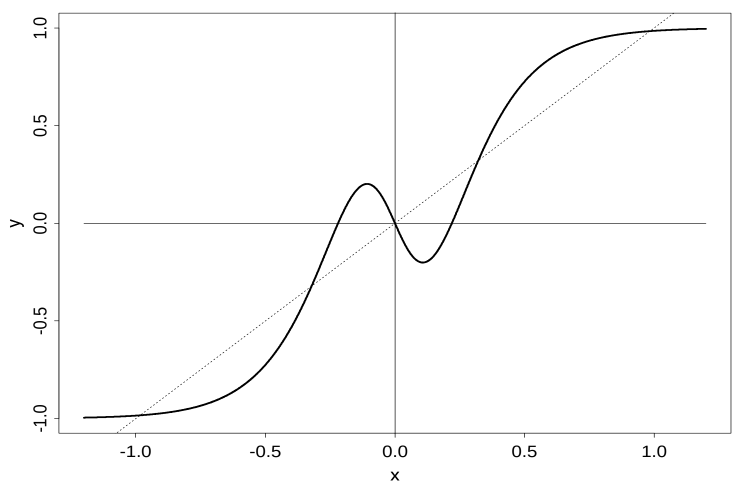


FIG. 4.16 – *Graphe de la fonction $x \mapsto 2\tanh(-6x) + 3\tanh(3x)$. Les valeurs de x vérifiant $x = f(x)$ sont* $\{-0,9837\ ;\ -0,3215\ ;\ 0\ ;\ 0,3215\ ;\ 0,9837\}$.

6. On ne donne que les 4 première décimales.

Comme l'illustrent les figures 4.18 et 4.17, la fonction de récurrence associée à ce modèle, peut converger vers l'un des deux points d'attraction a_1 ou a_2 (fig. 4.17), ou avoir un comportement chaotique et osciller entre les points de répulsion r_1 et r_3 (fig. 4.18) suivant la valeur du point de départ de la récurrence :

$$\begin{aligned}\lim_{n\to+\infty} \widehat{X}_{t+n} &= \begin{cases} a_1 & \text{si } X_t < r_1 \\ a_2 & \text{si } X_t > r_3 \end{cases} \\ \forall n \geq 1,\ \ \widehat{X}_{t+n} &\in\ ]r_1, r_2[\ \text{ si } X_t \in]r_1, r_2[\end{aligned}$$

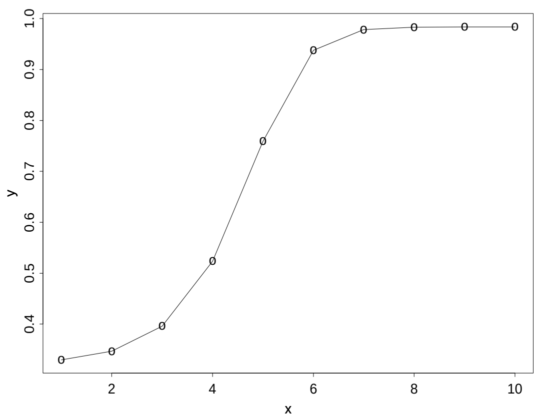


FIG. 4.17 – *Point de départ : 0.33.*

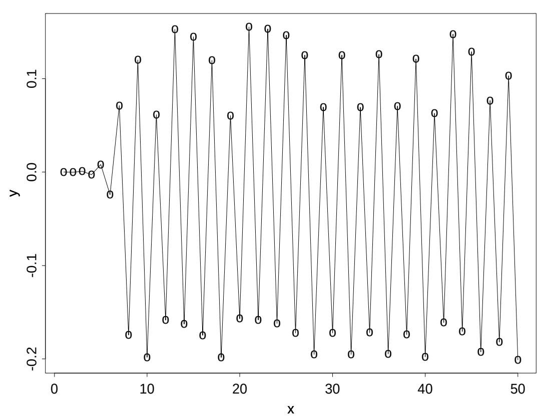


FIG. 4.18 – *Point de départ : 0.001.*

L'une des solutions du problème de la prévision à plus d'un pas est de construire un modèle neuronal qui génère des prévisions vectorielles, en calculant non plus une valeur future de la série mais plusieurs à la fois. Cette méthodologie a néanmoins l'inconvénient de produire un nombre de pas fixé à l'avance, ne laissant pas à l'utilisateur la possibilité de prévoir un horizon supérieur au nombre de sorties du modèle. Un autre inconvénient est dû à l'accroissement de la complexité du problème de prévision, complexité qui croît avec le nombre de sorties du réseau. La résolution de ce problème nécessite alors plus de ressources, et les problèmes d'estimation des paramètres que nous développons chapitre 5 deviennent plus aigus.

Une autre solution serait de maîtriser la fonction de récurrence, mais aucune étude à notre connaissance n'a été effectuée dans ce sens.

Chapitre 5

Estimation des paramètres d'un modèle neuronal

5.1 Introduction

Supposons que l'on tente de modéliser, à l'aide d'un modèle neuronal, un phénomène chronologique quelconque. En reprenant les notations et les principes énoncés chapitre 2, on suppose que ce phénomène peut s'écrire sous une forme autorégressive fonctionnelle :

$$\forall t \in \mathbb{Z}, \quad X_t = f(X_{t-1}^{(p)}) + \varepsilon_t \tag{5.1}$$

avec pour $p \geq 1$, $X_{t-1}^{(p)} = (X_{t-1}, X_{t-2}, \ldots, X_{t-p})$. On considère alors que la fonction f peut être approchée par un modèle neuronal dont la structure est définie et que l'on caractérise par la fonction f_θ par exemple de la classe du perceptron multicouches (voir définition 4.8) où $\theta \in \mathbb{R}^\ell$ est le vecteur des paramètres. L'étape d'estimation consiste alors à estimer les paramètres $(\theta_i)_{i=1,2,\ldots,\ell}$ qui rendent compte du phénomène. L'estimation se fait par la méthode des moindres carrés, bien connue dans le cadre de la régression et l'autorégression. L'étape d'estimation se résume donc à résoudre un problème d'optimisation.

Soit $(X_t)_{1-p \leq t \leq T}$, $T+p$ valeurs successives de la série. On note alors $S_T(\theta)$ la somme des erreurs quadratiques associée, qui constitue la fonction de coût à minimiser en θ :

$$S_T(\theta) = \sum_{t=1}^{T} (X_t - f_\theta(X_{t-1}^{(p)}))^2 \quad . \tag{5.2}$$

Le gradient de cette fonction de coût par rapport au vecteur θ est alors notée :

$$\nabla S_T(\theta) = \frac{\partial S_T(\theta)}{\partial \theta} \tag{5.3}$$

et la matrice des dérivées secondes H (la matrice Hessienne) par

$$H(\theta) = \nabla^2 S_T(\theta) \tag{5.4}$$

Dans le langage connexionniste, l'« estimation des paramètres » s'appelle l'« apprentissage », et la suite de valeurs $(X_t)_{1-p\leq t\leq T}$, la « base d'apprentissage ». Si on veut tester le modèle sur des valeurs de la série n'ayant pas servi à l'apprentissage (par exemple $(X_{T+j})_{j=1,2,\ldots,T'}$ cette nouvelle suite de réels est appelée « base de test ». Si l'apprentissage est mal entrepris, la performance du modèle (la valeur de la fonction de coût) obtenue sur la base d'apprentissage et celle obtenue sur la base de test sont significativment différentes. On parle alors de mauvaise *adéquation* entre le modèle et le phénomène à modéliser, ou de mauvaise *généralisation*.

Pour ce problème d'optimisation, la fonction de coût à minimiser est une fonction quadratique facile à gérer si la fonction f est linéaire. Néanmoins, dans le cas où cette fonction est non-quadratique, on doit faire appel à des algorithmes itératifs basés en général sur le gradient de cette fonction. Dans le cas où le modèle est un $\mathrm{NAR}_n(p)$ (définition 4.8) la fonction de coût à minimiser est non-linéaire et le vecteur des paramètres θ associé est soumis à des conditions de cohérence, en raison de la forme particulière de l'architecture d'un perceptron multicouches. On peut préciser ces conditions comme suit.

Définition 5.1 :

Soit un modèle du type $NAR_n(p)$ (définition 4.8), comprenant p entrées et n neurones cachés. Soit $\theta = \{(\alpha_j)_{0\leq i\leq n}, (\beta_{ij})_{0\leq i\leq p,1\leq j\leq n}\}$ 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 $j = 1,\ldots,n$, α_j est nul ssi $\beta_{ij} = 0$ pour tout $i = 0,1,\ldots,p$;*
- *pour n'importe quel $j = 1,\ldots,n$, si $\beta_{1j} = \cdots = \beta_{pj} = 0$, alors $\beta_{0j} = 0$.*

Ces contraintes sont indispensable pour la cohérence du réseau. Par exemple si il existe i et j tel 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.

On note $\hat{\theta}_T$ l'estimateur des moindres carrés de θ respectant les conditions de cohérence (définition 5.1), soit

$$\hat{\theta}_T := Arg \min_{\theta\in\Theta} S_T(\theta)$$

Dans le cadre de la prévision de séries temporelles, on prefère définir un autre critère de comparaison des performances de l'optimisation, qui est dérivé de l'erreur quadratique brute et normalisé pour être indépendant de l'unité de mesure des valeurs de la série. On dénomme ce critère $NMSE(\theta)$ pour « *Normalize Mean Square Error* ».

Définition 5.2 :

Dans le cadre défini chapitre 2, on appelle $NMSE(\theta)$ le critère de la forme :

$$\mathrm{NMSE}(\theta) = \frac{S_T(\theta)/T}{Var^T(X)} \quad , \tag{5.5}$$

avec

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

Le terme $Var^T(X)$ correspond à la variance empirique des valeurs de la série et peut être associé à la performance d'un modèle trivial M_0 donnant pour chaque temps t une prévision de X_t égal à la moyenne de la série. Ainsi si le critère $NMSE(\theta)$ est égal (resp. supérieur) à 1, le modèle est aussi performant (resp. moins performant) que le modèle M_0. Enfin, plus ce critère est proche de 0, et plus le modèle est performant. Remarquons que $S_T(\theta)$ et $NMSE(\theta)$ sont égaux proportionnelement à une constante positive près, et que leurs minimisations sont deux problèmes rigoureusement équivalents.

5.1.1 Consistance de l'estimateur des moindres carrés

Reprenons les notations et le résultat du théorème 4.10 du chapitre 4. À partir de ces résultats, il est facile, comme nous l'avons vu dans le chapitre 3, d'obtenir la consistance de l'estimateur des moindres carrés du vecteur de paramètres. Notre but ici est de reprendre ces résultats, et de les appliquer à l'estimateur des moindres carrés d'un modèle neuronal basé sur un perceptron multicouches noté $\mathrm{NAR}_n(p)$ (définition 4.8) si celui-ci comporte p entrées.

Rappelons que $\theta = (\theta_1, \theta_2, \ldots, \theta_\ell)$, $\ell \geq 1$ est le vecteur des paramètres ou, pour un modèle neuronal, le vecteur des ℓ poids synaptiques pris dans un ordre quelconque. Les notations suivantes sont les mêmes que dans le chapitre 3 : θ_0 est le vecteur des « vrais paramètres » correspondant au modèle optimal, $\hat{\theta}_T$ est l'estimateur des moindres carrés. On suppose que le modèle est du type correspondant à celui décrit équation 4.12. En particulier, en posant $X_t^{(p)} = (X_t, X_{t-1}, \ldots, X_{t-p+1})$ la série $(X_t^{(p)})$ est une chaîne de Markov homogène dans $\mathbb{R}^p$.

Théorème 5.3 :

Soit un modèle $NAR_n(p)$ correspondant à la définition 2.8 avec les fonctions d'activations ψ sigmoïdes. Supposons que soit satisfait l'ensemble **[R]** *des conditions suivantes :*

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

(ii) ε_1 *a une densité positive par rapport à la mesure de Lebesgue avec* $E(\varepsilon_1) = 0$ *et* $E(\varepsilon_1^2) < \infty$.

(iii) θ *appartient à un sous-ensemble compact* Θ *de* $\mathbb{R}^d$, *tel que* $\theta_0 \in \overset{\circ}{\Theta}$.

(iv) (Condition d'identifiabilité) pour tout θ, $\theta \neq \theta_0$, $f_\theta \neq f_{\theta_0}$ *c'est-à-dire qu'il existe un* $\mathbf{x} \in \mathbb{R}^p$ *tel que* $f_\theta(\mathbf{x}) \neq f_{\theta_0}(\mathbf{x})$.

Alors l'estimateur des moindres carrés $\hat{\theta}_T := \mathrm{Arg} \min_{\theta\in\Theta} S_T(\theta)$ *est fortement consistant.*

Preuve. Le modèle $\mathrm{NAR}_n(p)$ est en fait un modèle $ARF_1(p)$ particulier. La démonstration consiste à vérifier que les conditions **[M]**, **[D]** relatives aux modèle fonctionnel $\mathrm{ARF}_d(p)$ (avec $d = 1$) du théorème 3.10 sont satisfaites.

Conditions [M] : Il est aisé de de voir que les conditions **[R]** du théorème 5.3 satisfont le jeu d'hypothèse **[C.2]** du théorème 4.10 (chapitre 4). En particulier, sous les conditions **[R]**-*(i)* et **[R]**-*(ii)*, la chaîne $X^{(p)}$ issue d'un modèle $\mathrm{NAR}_n(p)$ est stable et a une unique loi invariante μ_{θ_0}, ce qui remplit la condition **[M]**-*(ii)*. Les conditions **[M]**-*(i)-(a)* et **[M]**-*(i)-(b)* sont clairement vérifiées par les conditions **[R]**-*(i)* et **[R]**-*(ii)*. Puisque la fonction $f_\theta : \mathbb{R}^p \mapsto \mathbb{R}$ est lipschitzienne (en x) et bornée (proposition 4.6), et que Θ est compact (condition **[R]**-*(iii)*), $(\mathbf{x}, \theta) \mapsto f_\theta(\mathbf{x}))$ est uniformément continue sur $\mathbb{R}^p \times \Theta$, ce qui satisfait **[M]**-*(iii)-(a)* et **[M]**-*(iii)-(b)*.

Conditions [D] : En ce qui concerne l'identifiabilité du modèle, en raison de **[R]**-*(i)* et **[R]**-*(ii)*, on sait que le bruit (ε_t) possède une densité strictement positive par rapport à la mesure de Lebesgue. On sait alors que pour le modèle $\mathrm{NAR}_n(p)$, la loi stationnaire μ_{θ_0} de la chaîne $X^{(p)}$ (sous $\mathbb{P}_{\theta_0}$) a aussi une densité strictement positive par rapport à la mesure de Lebesgue. La condition **[R]**-*(iv)* suffit alors pour satisfaire la condition d'identifiabilité. ∎

Ce résultat assure donc la convergence presque sûre de l'estimateur vers le vrai paramètre lors de l'optimisation de la fonction de coût (équation 5.2).

5.1.2 La rétropropagation du gradient

Lorsque la fonction f_θ est non-linéaire, la minimisation de la fonction de coût $S_T(\theta)$ fait appel à des algorithmes itératifs[1] presque tous basés sur un calcul du gradient. Dans le cas où cette fonction est de la classe des perceptrons multicouches, le problème sur lequel se sont penchés un certain nombre de chercheurs, a été de trouver une manière de calculer ce gradient dans un temps raisonnable. Un algorithme très simple, la *rétroprogation du gradient*, a été ainsi proposé indépendamment par plusieurs équipes de chercheurs dans les années 80 : (Rumelhart et al., 1986) et (LeCun, 1985). Cet algorithme devenu célèbre, plus connu sous le terme anglais « *backpropagation* », a permis ainsi de relancer la recherche sur les réseaux de neurones à couches. Le calcul de base qui est la clef de la méthode n'est en fait rien d'autre qu'un calcul astucieux basé sur des dérivées de composés de fonctions. Cet algorithme permet d'obtenir la valeur exacte du gradient, en rétropropageant l'erreur de la sortie vers l'entrée. Sa complexité est, en nombre de calculs, du même ordre que la *propagation* qui permet de calculer la sortie du réseau.

Nous n'entrerons pas ici dans les détails, l'algorithme est présenté dans la plupart des livres et thèses consacrés aux réseaux de neurones (voir par exemple (Hertz et al., 1991)). Notons simplement qu'il existe des algorithmes similaires permettant de calculer les dérivées secondes (Cf. (Buntine et Weigend, 1994)).

5.2 Méthodes d'optimisation

On décrit dans cette partie les algorithmes qui exploitent le gradient pour effectuer l'optimisation de la fonction de coût (voir équation 5.2). Cette fonction étant non-linéaire et pouvant avoir un relief très « vallonné » (voir section 5.3), ces méthodes n'ont pas toutes les mêmes possibilités en terme de rapidité de convergence et de capacité à éviter les minima locaux.

5.2.1 Le gradient total/stochastique

Le principe des méthodes de gradient consiste à modifier les paramètres de la fonction à minimiser dans la direction de la plus forte pente, donc dans la direction opposée au vecteur gradient. Minimiser la fonction de coût (5.2) consiste alors à modifier itérativement le vecteur de paramètres. Si on modifie les paramètres dans la direction opposée au vecteur gradient de la fonction de coût globale, calculée sur toutes les données, cette méthode est appelée le *gradient total*. À la $k^{ème}$itération, on a

$$\theta^k = \theta^{k-1} - \eta_k \nabla S_T(\theta^{k-1}) \tag{5.6}$$

1. Certains sont détaillés dans la section suivante.

où η_t est un terme « petit » dont la valeur est à régler, et qui est appelé le *pas du déplacement*.

On utilise, en général, une version différente de l'algorithme, où on effectue une modification du vecteur des paramètres dans la direction opposée au vecteur gradient associé à un seul terme d'erreur quadratique de la fonction de coût (5.2), tiré aléatoirement suivant une loi uniforme. Pour un $t \in \{1, 2, \ldots, T\}$ choisi aléatoirement à l'étape k, on a

$$\theta^k = \theta^{k-1} - \eta_k \frac{\partial (X_t - f_{\theta^{k-1}}(X_{t-1}^{(p)}))^2}{\partial \theta} \tag{5.7}$$

Cette variante de la méthode du gradient total est appelée méthode du « gradient stochastique » (voir (Duflo, 1990) pour plus de détails) et permet dans une certaine mesure, d'éviter les pièges des minima locaux. Elle permet aussi la minimisation de la fonction de coût en « temps réel » quand les observations ne sont pas toutes simultanément disponibles.

Partant de la technique du gradient total, de récentes recherches (Duflo, 1996) ont permis de mettre au point une technique d'optimisation presque sûre vers le minimum global. Cette méthode consiste à ajouter un bruit stochastique au terme de descente de gradient. On se contente ici de donner un exemple de modification itérative du vecteur des paramètres assurant la convergence presque sûre (pour plus de détails et pour connaître les conditions exactes, cf (Duflo, 1996) section 7.IV) :

$$\theta^k = \theta^{k-1} - \eta_k \nabla S_T(\theta^{k-1}) + \gamma_k \epsilon_k \tag{5.8}$$

avec

- $\forall k \in \mathbb{Z}, \quad \epsilon_k \sim \mathcal{N}(0, 1)$.
- $\gamma_k = c/\ell n(k)$ avec c « petit ».
- $\eta_k = 1/\sqrt{k}$

Les premiers essais ont montré un comportement satisfaisant de l'algorithme. Il s'avère néanmoins que le coefficient c s'avère difficile à régler dans le cadre de la modélisation neuronale. En particulier, les temps de convergence peuvent être allongés de manière significative, ce qui peut être considéré comme un lourd handicap dans le cas où la dimension du vecteur des paramètres est grand.

Inconvénient des méthodes de gradient. Les utilisateurs des méthodes du gradient (ou méthodes du premier ordre) s'accordent pour trouver que le temps de convergence du réseau est long. De plus, la vitesse de convergence dépend étroitement de la valeur du pas de déplacement, et on ne connaît pas, a priori, la valeur la plus adéquate. Il est aussi difficile de déterminer le moment où on peut considérer que le modèle a convergé. Un critère d'arrêt fiable consisterait à stopper la convergence lorsque la norme du gradient de la fonction de coût est à zéro, mais l'imprécision de la méthode permet difficilement l'utilisation d'un tel critère, le calcul du gradient s'effectuant rarement au point minimum exact. En ce qui concerne la méthode du gradient stochastique, son efficacité à éviter les minima globaux dépend aussi de la valeur du pas de déplacement.

5.2.2 Méthodes quasi-newtoniennes

L'idée des méthodes quasi-newtoniennes est de d'utiliser l'information issue des dérivées du second ordre. Ces dérivées sont cependant estimées en général, à partir des dérivées premières, afin d'éviter des temps de calcul trop importants. On décrit ici brièvement le principe des algorithmes du second ordre.

Reprenons les notations de la section 5.1.1. Soit $h \in \mathbb{R}^\ell$, tel que $\theta + h$ appartenant à un voisinage de θ et considérons le développement de Taylor de la fonction de coût définie équation 5.2 :

$$S_T(\theta + h) = S_T(\theta) + {}^{\mathrm{t}}h\, \nabla S_T(\theta) + \frac{1}{2}{}^{\mathrm{t}}h\; H\; h + O(\| \; h \; \|^3) \qquad (5.9)$$

En supposant que la fonction $S_T(\theta + h)$ est presque quadratique en θ au voisinage d'un minimum, le terme $O(\| \; h \; \|^3)$ n'est pas significatif. En dérivant (5.9) par rapport à h, on obtient alors :

$$\nabla S_T(\theta + h) = \nabla S_T(\theta) + H(\theta)\; h$$

On sait alors que la valeur h^* qui minimise $S_T(\theta + h)$ est telle que : $\nabla S_T(\theta + h^*) = 0$, soit

$$H(\theta)\; h^* = -\nabla S_T(\theta) \quad .$$

Dans le cas où $S_T(\theta)$ est de classe $\mathcal{C}^2$ et admet un minimum en θ^*, le gradient s'annule en θ^*, et la matrice Hessienne $H(\theta)$ est définie non négative dans un voisinage de θ^*. On cherche alors θ^* en utilisant l'itération de Newton appliquée au gradient de $S_T(\theta)$:

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

Si le point de départ θ n'est pas suffisamment proche de θ^*, il peut arriver que l'algorithme ne converge pas, ou qu'il converge vers un point stationnaire qui n'est pas un minimum (maximum ou point selle). Si au cours de l'itération $H(\theta^{k-1})$ reste définie positive, le vecteur $-H^{-1}(\theta^{k-1}) \nabla S_T(\theta^{k-1})$ est une direction de descente et il existe η_k tel que

$$S_T\left(\theta^{k-1} - \eta_k\; H^{-1}(\theta^{k-1}) \nabla S_T(\theta^{k-1})\right) < S_T(\theta^{k-1}) \qquad (5.10)$$

D'où l'idée des méthodes quasi-newtoniennes qui consistent à modifier le vecteur des paramètres à la $k^{ème}$itération de la façon suivante :

$$\theta^k = \theta^{k-1} - \eta_k\; \widehat{H}^{-1}(\theta^{k-1}) \nabla S_T(\theta^{k-1})$$

où $\widehat{H}(\theta^k)$ est une matrice définie positive qui converge vers la matrice Hessienne $H(\theta^*)$ lorsque k tend vers l'infini. La suite de réels η_k est obtenue par une minimisation unidirectionnelle de $S_T(\theta)$ dans la direction de $-H(\theta^{k-1})^{-1}\nabla S_T(\theta^{k-1})$.

Les différentes méthodes quasi-newtoniennes diffèrent alors par le choix des matrices $\widehat{H}(\theta)$. La méthode BFGS (Broyden-Fletcher-Goldfarb-Shanno) a été retenue parce qu'elle n'impose pas de forme particulière à la fonction $S_T(\theta)$ et parce qu'elle est peu sensible à l'exactitude de la minimisation unidirectionnelle (calcul de $(\eta_k)_{k>1}$). La matrice $\widehat{H}(\theta^k)$ est alors calculée à la $k^{ème}$itération suivant la formule :

$$\begin{aligned}\widehat{H}(\theta^k) &= \widehat{H}(\theta^{k-1}) + \frac{(\theta^k - \theta^{k-1})(\theta^k - \theta^{k-1})}{(\theta^k - \theta^{k-1})(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k))} \\ &- \frac{[\widehat{H}(\theta^{k-1})(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k))] \times [\widehat{H}(\theta^{k-1})(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k))]}{(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k))\widehat{H}(\theta^{k-1})(\nabla S_T(\theta^{k-1}) - \nabla S_T(\theta^k))}\end{aligned}$$

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

Pour un réseau de neurones, les paramètres à déterminer sont les poids synaptiques et la fonction à minimiser est la fonction de coût en sortie de réseau (en général l'erreur quadratique). À l'instar de la méthode du gradient total, cette méthode est basée sur le gradient de la fonction de coût de l'ensemble des données.

5.2.3 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-newtonienne utilisant l'inverse de la matrice Hessienne. Elle est actuellement la méthode standart dès lors que la fonction de coût est une somme d'erreurs quadratiques.

Reprenons les notations de la section précédente et posons

$$\varepsilon_t = X_t - f_\theta(X_{t-1}^{(p)}), \quad \forall t \in \{1, 2, \ldots, T\} \quad .$$

On a alors $S_T(\theta) = \sum_{i=1}^{T} \varepsilon_t^2$ et, pour $i \in \{1, 2, \ldots, \ell\}$ (on rappelle que $(\theta_i)_{i=1,2,\ldots,\ell}$ sont les composantes du vecteur des paramètres θ) :

$$\begin{aligned}\frac{\partial S_T(\theta)}{\partial \theta_i} &= -2\sum_{t=1}^{T} \varepsilon_t \frac{\partial f_\theta(X_{t-1}^{(p)})}{\partial \theta_i} \\ \frac{\partial^2 S_T(\theta)}{\partial \theta_i \partial \theta_j} &= 2\sum_{i=t}^{T}(\frac{\partial f_\theta(X_{t-1}^{(p)})}{\partial \theta_i}\frac{\partial f_\theta(X_{t-1}^{(p)})}{\partial \theta_j} - \varepsilon_t \frac{\partial^2 f_\theta(X_{t-1}^{(p)})}{\partial \theta_i \partial \theta_j})\end{aligned} \tag{5.11}$$

Quand l'on est proche du minimum, les $(\varepsilon_t)_{1\leq i\leq T}$ tendent à être une suite de variables aléatoires indépendantes, centrées, et de variance constante (bruit blanc). Le deuxième terme de l'équation 5.11 tend alors vers 0 en probabilité (il peut cependant entraîner au début de l'algorithme des perturbations importantes). Le principe de la méthode LM consiste alors à exploiter cette approximation. Posons :

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

et

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

Conformément aux méthodes du second ordre, on peut alors modifier le vecteur des paramètres en utilisant l'itération de Newton :

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

Cependant au lieu d'estimer $H^{-1}(\theta)$ suivant la méthode décrite section 5.2.2, on peut tenter de résoudre le système d'équation linéaire :

$$\sum_{i=l}^{\ell} \alpha_{ij} \delta_j = \beta_i \tag{5.12}$$

où δ_i est la $i^{ème}$composante de $\theta^k - \theta^{k-1}$. Cette méthode a cependant l'inconvénient d'être instable. L'astuce de LM consiste alors à remplacer les $(\alpha_{ij})_{1\leq i,j\leq \ell}$ par : $(\alpha'_{ij})_{1\leq i,j\leq \ell}$ définis par :

$$\begin{cases} \alpha'_{jj} & := & \alpha_{jj}(1+\lambda) \quad \forall j \in \{1,2,\ldots,\ell\} \\ \alpha'_{ij} & = & \alpha_{ij} \quad \text{si} \quad i,j \in \{1,2,\ldots,\ell\}, \quad i \neq j \end{cases} \quad .$$

où λ est un réel positif.

Quand λ est grand, la matrice $(\alpha'_{ij})_{1\leq i,j\leq \ell}$ devient fortement diagonal, et on se rapproche de la méthode du gradient de la plus forte pente : $\delta_i \approx \frac{1}{\alpha_{ii}}\beta_i$ et $\frac{1}{\alpha_{ii}}$ peut être vu comme le pas du gradient. Quand λ est « petit », on se rapproche d'une méthode du second ordre utilisant la matrice Hessienne H : $\theta^k - \theta^{k-1} \approx -H^{-1}(\theta^{k-1})\nabla S_T(\theta^{k-1})$

Notons que cette méthode n'est plus valide dès lors que la fonction de coût n'est plus la somme des erreurs quadratiques du modèle. Dans les exemples de modélisations présentés tout au long de ce mémoire on utilise en général soit la méthode BFGS soit la méthode LM pour l'estimation des paramètres. On peut trouver plus de détails sur ces méthodes d'optimisation dans (Press et al., 1988) et (Minoux, 1983).

5.3 Problèmes de l'estimation des paramètres

On résume dans cette section les problèmes pratiques et théoriques de la modélisation neuronale. Cette liste ne se veut pas exhaustive, mais montre néanmoins les difficultés que l'on peut rencontrer lors de l'étape d'estimation des paramètres.

- **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 surapprentissage.** Du fait des capacités d'approximation universelle des modèles neuronaux, l'apprentissage peut mener à un surajustement (*overfitting* en anglais) de la série. Utiliser un modèle comportant un grand nombre de paramètres, pour modéliser un problème de trop faible complexité peut entraîner ce genre de problème. Dans ce cas, en raison de la mauvaise évaluation de la variance de la perturbation aléatoire associée au phénomène, il y a mauvaise adéquation entre le modèle et le problème de prévision réel. On peut voir figure 5.1 un effet du surajustement lors d'une étape d'estimation des paramètres. En particulier, on observe au delà d'un certain nombre d'itérations de la phase d'optimisation, une croissance de la somme des erreurs quadratiques sur des valeurs ne servant pas à l'apprentissage.

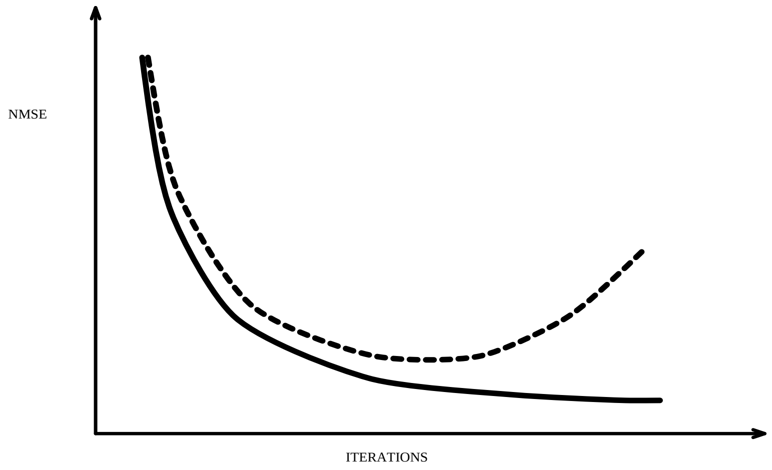


FIG. 5.1 – *Effet du suraprentissage. En abscisse le nombre d'itérations de la méthode d'optimisation utilisée. Le trait plein correspond à la somme des erreurs quadratique normalisé (voir définition 5.2) calculée sur la base d'apprentissage. Le trait en pointillé correspond à ce même critère calculé sur des valeurs de la série non utilisées pour l'apprentissage (base de test).*

Afin d'illustrer les problèmes d'estimation des paramètres cités ci-dessus, on propose une série d'exemples d'optimisation, basés sur un problème de prévision concret. La série choisie est une série réelle au comportement reputé complexe : la série des tâches solaires *sunspots*. Les tâches solaires (*sunspots*) sont des tâches sombres qui apparaissent sur la surface du Soleil. Elles ont souvent un diamètre supérieur à celui de la Terre et ont été observées dès 1610, peu après l'invention du télescope. Des données correspondant au nombre moyen d'apparition de ces tâches par an ont été collectées depuis 1700. La période moyenne s'écoulant entre deux maxima est de 11 ans. Néanmoins, elle peut varier de 7 à 15 ans. Le mécanisme engendrant les apparitions de ces taches n'est encore que partiellement connu, on suppose notamment qu'il est lié au champ magnétique solaire qui change en moyenne tous les 22 ans (voir figure 5.2).

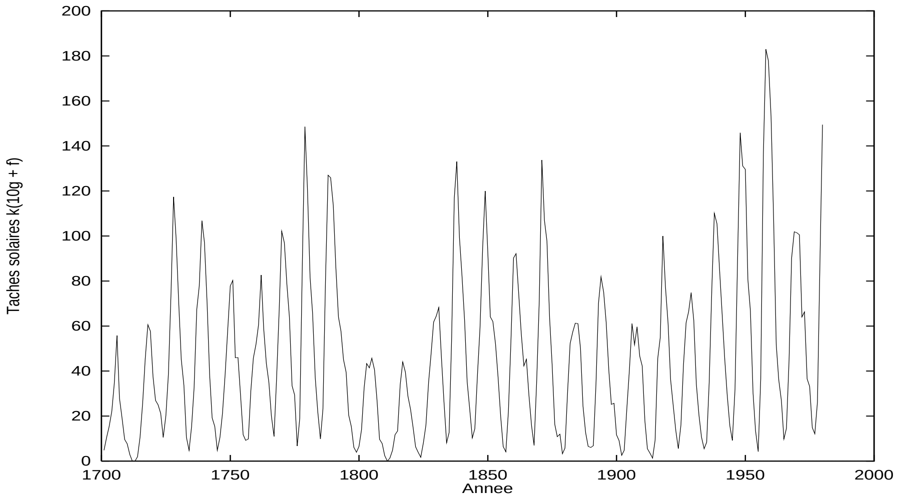


FIG. 5.2 – *La série sunspots*

On partage les données en deux parties : une base d'apprentissage et une base de test. La base d'apprentissage comprend 2/3 des points de la série, tirés aléatoirement, la base de test est composée des points restants. On tente ici de déterminer, parmi toutes les architectures possibles de modèles $\mathrm{NAR}_n(p)$ (voir définition 4.8) pour $n \in \{1, 2, \dots, 15\}$ et $p \in \{1, 2, \dots, 12\}$, laquelle semble la plus adaptée pour modéliser la série *sunspots*. Pour chaque couple (n, p), $n \in \{1, 2, \dots, 15\}$, $p \in \{1, 2, \dots, 12\}$, une centaine d'estimations du vecteur des paramètres du modèle $\mathrm{NAR}_n(p)$ sont effectuées en utilisant les données de la *base d'apprentissage*. Les 100 optimisations associées sont obtenues en prenant pour chacune d'elles des paramètres initiaux différents, choisis aléatoirement suivant une loi uniforme dans l'intervalle [-1,1]. On note $(\hat{\theta}^i)_{1\leq i\leq 100}$ les 100 vecteurs de paramètres estimés obtenus. Les différentes modélisations sont évaluées en utilisant le critère $\mathrm{NMSE}(\theta)$ (voir définition 5.2). Si on évalue le modèle $\mathrm{NAR}_n(p)$ ayant pour vecteur de paramètre θ sur la base d'apprentissage, on note l'évaluation $\mathrm{NMSE}^{app}_{n,p}(\theta)$, et $\mathrm{NMSE}^{tst}_{n,p}(\theta)$ si on évalue ce même modèle sur la base de test.

Pour chaque couple (n, p), $n \in \{1, 2, \dots, 15\}$, $p \in \{1, 2, \dots, 12\}$ on calcule alors

- La moyenne des évaluations $(\mathrm{NMSE}^{app}_{n,p}(\hat{\theta}^i))_{1\leq i\leq 100}$ (notée $\mathrm{moy}^{n,p}(\mathrm{app})$) :

$$\mathrm{moy}_{n,p}(\mathrm{app}) = \frac{1}{T}\sum_{i=1}^{100} \mathrm{NMSE}^{app}_{n,p}(\hat{\theta}^i) \quad ,$$

et la moyenne des évaluations $(\mathrm{NMSE}^{app}_{n,p}(\hat{\theta}^i))_{1\leq i\leq 100}$ (notée $\mathrm{moy}_{n,p}(\mathrm{test})$[2]) :

$$\mathrm{moy}_{n,p}(\mathrm{tst}) = \frac{1}{T}\sum_{i=1}^{100}\mathrm{NMSE}^{tst}_{n,p}(\hat{\theta}^i) \quad .$$

- Le minimum des évaluations $(\mathrm{NMSE}^{app}_{n,p}(\hat{\theta}^i))_{1\leq i\leq 100}$ (noté $\mathrm{min}_{n,p}(\mathrm{app})$) :

$$\mathrm{min}_{n,p}(\mathrm{app}) = \min_{\theta\in\{\hat{\theta}^1,\hat{\theta}^2,\ldots,\hat{\theta}^{100}\}}\mathrm{NMSE}^{app}_{n,p}(\theta) \quad ,$$

  et le minimum des évaluations $(\mathrm{NMSE}^{tst}_{n,p}(\hat{\theta}^i))_{1\leq i\leq 100}$ (noté min(tst)) :

$$\mathrm{min}_{n,p}(\mathrm{tst}) = \min_{\theta\in\{\hat{\theta}^1,\hat{\theta}^2,\ldots,\hat{\theta}^{100}\}}\mathrm{NMSE}^{tst}_{n,p}(\theta) \quad .$$

- La variance des critères $(\mathrm{NMSE}^{app}_{n,p}(\hat{\theta}^i))_{1\leq i\leq 100}$ (noté $\mathrm{var}_{n,p}(\mathrm{app})$) :

$$\mathrm{var}_{n,p}(\mathrm{app}) = \frac{1}{T}\sum_{i=1}^{100}(\mathrm{NMSE}^{app}_{n,p}(\hat{\theta}^i) - \mathrm{moy}_{n,p}(\mathrm{app}))^2 \quad ,$$

  et la variance des critères $(\mathrm{NMSE}^{tst}_{n,p}(\hat{\theta}^i))_{1\leq i\leq 100}$ (noté $\mathrm{var}_{n,p}(\mathrm{tst})$) :

$$\mathrm{var}_{n,p}(\mathrm{tst}) = \frac{1}{T}\sum_{i=1}^{100}(\mathrm{NMSE}^{tst}_{n,p}(\hat{\theta}^i) - \mathrm{moy}_{n,p}(\mathrm{tst}))^2 \quad .$$

Les graphiques 5.3, 5.4, 5.5, 5.6, 5.7 et 5.8 représentent respectivement leurs valeurs pour $(n \in \{1,2,\ldots,15\}, p \in \{1,2,\ldots,12\})$.

On peut alors donner les analyses suivantes :

- Les valeurs de $(\mathrm{moy}_{n,p}(\mathrm{app}))_{n\in\{1,2,\ldots,15\},p\in\{1,2,\ldots,12\}}$ (figure 5.3) diminuent quand le nombre de retards p et le nombre de neurones de la couche cachée n augmentent. Ces valeurs peuvent être très petites (le perceptron multicouches est un approximateur universel).
- Les valeurs de $(\mathrm{moy}_{n,p}(\mathrm{tst}))_{n\in\{1,2,\ldots,15\},p\in\{1,2,\ldots,12\}}$ (figure 5.4) sont fortes pour $n = 1$ et $p = 1$ (on observe un sous-ajustement des données) et très fortes pour approximativement $p \geq 8$ et $n \geq 5$ (on observe un sur-ajustement des données). Ici, le sous-ajustement et le sur-ajustement sont liés respectivement à une sous-paramétrisation et une sur-paramétrisation du modèle.
- Les modèles qui donnent les meilleurs performances sur la base de test ont des structures très variées (voir figure 5.6).

2. On rappelle que les vecteurs de paramètres estimés sont obtenus en utilisant uniquement les données de la base d'apprentissage.

- Pour $p \geq 10$ et $n \geq 12$, les variances des différents critères calculés sur la base de test sont fortes, alors qu'elles ne le sont pas en ce qui concerne les critères relatifs, calculés sur la base d'apprentissage. Donc, pour ces valeurs, des évaluations de performances proches sur la base d'apprentissage, peuvent être sensiblement différentes sur la base de test.

Toutes ces analyses montrent la difficulté de déterminer le couple {structure de modèle du type perceptron multicouches ; paramètres estimés} capable de fournir les meilleures performances de modélisation de prévision de la série *sunspots*.

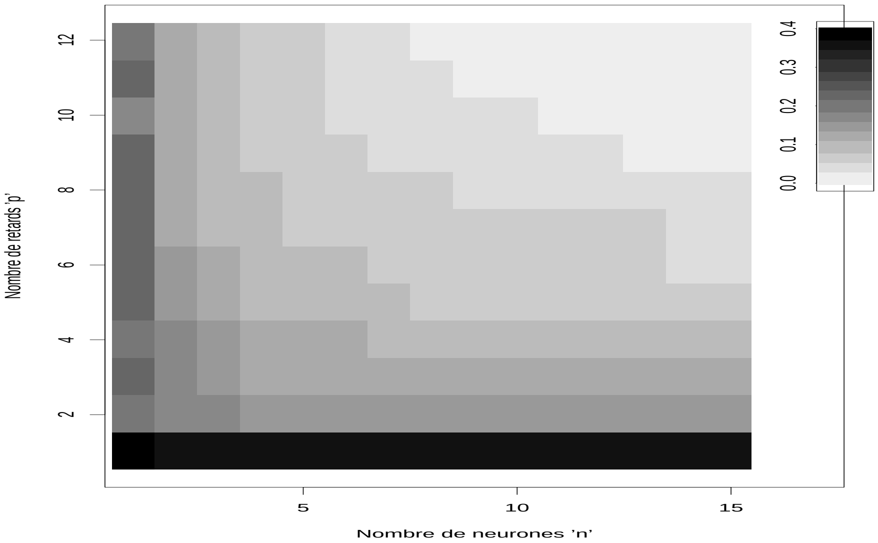


FIG. 5.3 – *Représentation graphique des valeurs* $(moy_{n,p}(app))_{n\in\{1,2,\ldots,15\},p\in\{1,2,\ldots,12\}}$. *Les valeurs faibles sont en claires, les valeurs fortes en sombres.*

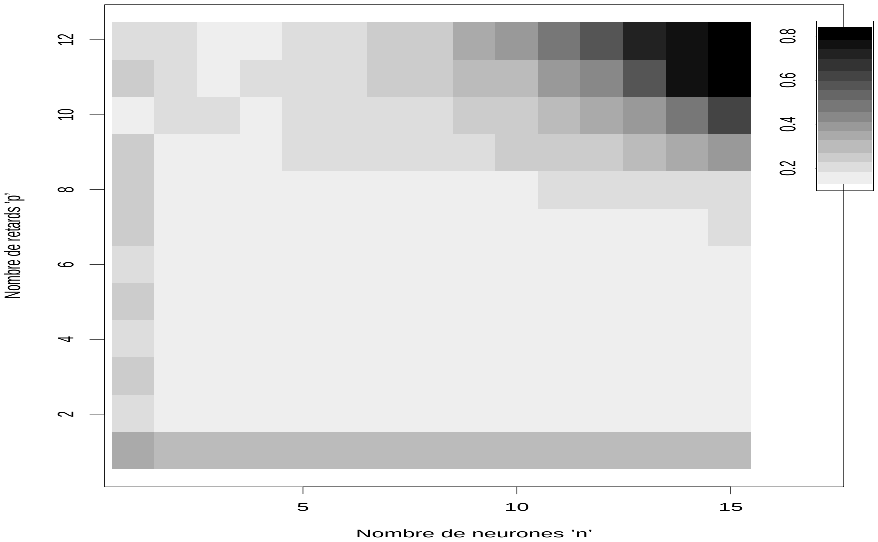


FIG. 5.4 – *Représentation graphique des valeurs* $(moy_{n,p}(tst))_{n\in\{1,2,\ldots,15\},p\in\{1,2,\ldots,12\}}$. *Les valeurs faibles sont en claires, les valeurs fortes en sombres.*

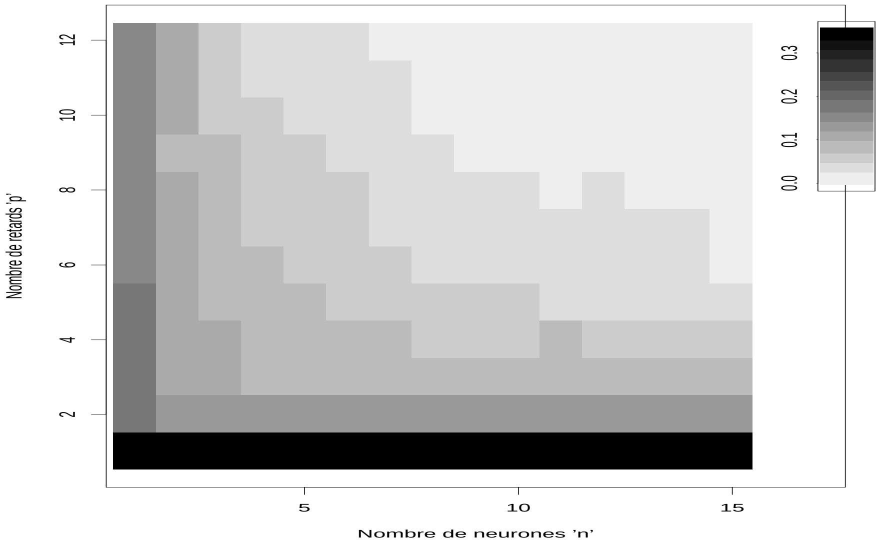


FIG. 5.5 – *Représentation graphique des valeurs* $(min_{n,p}(app))_{n\in\{1,2,\ldots,15\},p\in\{1,2,\ldots,12\}}$. *Les valeurs faibles sont en claires, les valeurs fortes en sombres.*

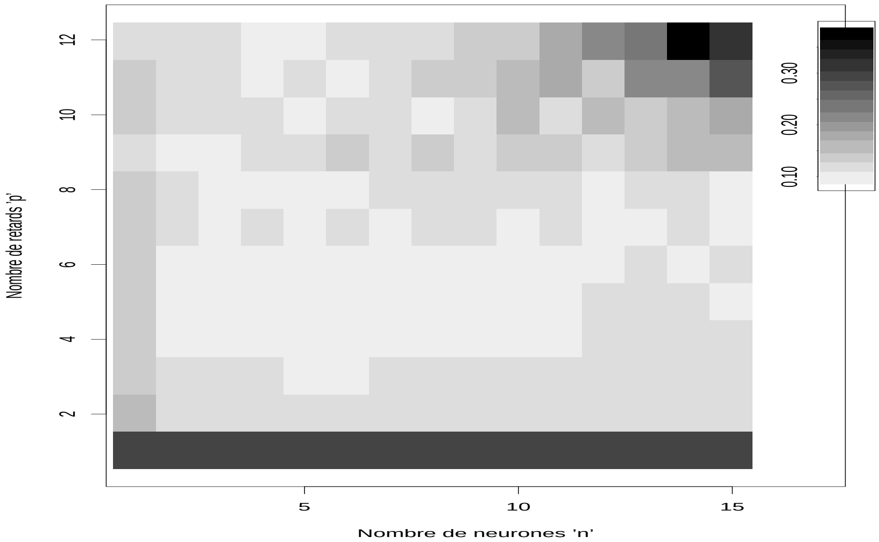


FIG. 5.6 – *Représentation graphique des valeurs* $(min_{n,p}(tst))_{n\in\{1,2,\ldots,15\},p\in\{1,2,\ldots,12\}}$. *Les valeurs faibles sont en claires, les valeurs fortes en sombres.*

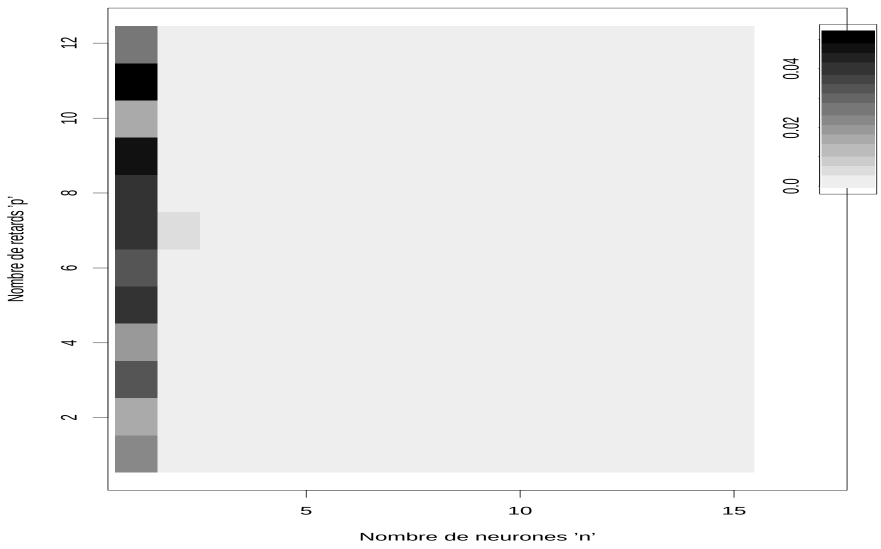


FIG. 5.7 – *Représentation graphique des valeurs* $(var_{n,p}(app))_{n\in\{1,2,\ldots,15\},p\in\{1,2,\ldots,12\}}$. *Les valeurs faibles sont en claires, les valeurs fortes en sombres.*

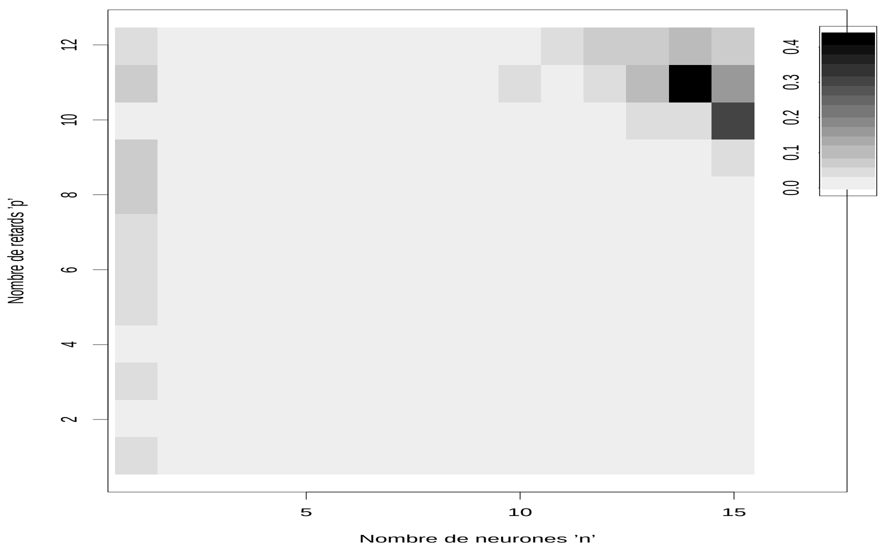


FIG. 5.8 – *Représentation graphique des valeurs* $(var_{n,p}(tst))_{n\in\{1,2,\ldots,15\},p\in\{1,2,\ldots,12\}}$. *Les valeurs faibles sont en claires, les valeurs fortes en sombres.*

Chapitre 6

Méthodes statistiques d'identification de modèles neuronaux

6.1 Introduction

Dans le protocole de modélisation paramétrique décrit chapitre 2, l'étape qui suit celle de l'estimation du vecteur des paramètres consiste à *identifier* le modèle. Cette étape réside en fait à découvrir la structure du modèle susceptible d'approcher la dynamique globale du phénomène à modéliser. Dans le cadre des modèles neuronaux du type perceptron multicouches décrits chapitre 4, cette étape se révèle à la fois complexe et indispensable :

- Complexe parce les non-linéarités des modèles neuronaux rendent caduques les méthodes classiques d'identification dans le cadre linéaire, et la théorie s'avère souvent en retard sur les heuristiques employées.
- Indispensable car les modèles non-linéaires en général et neuronaux en particulier ont de grandes capacités d'approximation. Il est donc important de découvrir une structure de modèle adéquate au problème, afin de réduire les risques d'une mauvaise adéquation entre le modèle et le phénomène à modéliser.

Ainsi, pour la mise au point d'un modèle paramétrique non-linéaire, deux questions doivent toujours être présentes à l'esprit du concepteur :

1. La complexité du modèle choisi est-elle suffisante pour modéliser correctement le phénomène ? (*problème dit de biais*)

2. Y-a-t-il adéquation entre le phénomène et le modèle trouvé ? Ou, plus précisément, les résultats de modélisation obtenus sur les données ayant permis la mise au point du modèle (la base d'apprentissage) sont-ils généralisables à l'ensemble des données disponibles ou à venir ? (*problème dit de variance*)

La recherche de la meilleure solution se trouve dans la détermination du compromis *biais/ variance* ou *simplicité/ complexité*. Pour un présentation plus complète et plus formelle de ce compromis, on pourra consulter l'article de (Geman et al., 1992).

Citons aussi les résultats de (Vapnik, 1982; Vapnik, 1992) qui introduit la notion de *capacité de la classe des fonctions* réalisables par les modèles. Dans le cas des perceptrons multicouches, cette capacité, qui n'est rien d'autre qu'un quantifieur de la capacité d'approximation de fonction, dépend :

– de la dimension du vecteur des paramètres, i.e. du nombre de connexions du réseau.

– de la norme des paramètres ; plus cette norme est petite, et plus les signaux parvenant aux neurones se situent dans un voisinage de 0, et plus les fonctions sigmoïdes utilisées sont proches de fonctions linéaires ce qui rend le modèle de plus faible capacité d'approximation.

Pour résoudre le problème relatif à la première question, une méthode simple consiste à utiliser toutes les variables disponibles et à choisir par exemple un perceptron multicouches avec une seule couche cachée comportant un grand nombre de neurones. Le théorème d'approximation universelle énoncé chapitre 4 section 4.2.1 donne alors une réponse satisfaisante à la première question. Encore faut-il pouvoir définir ce qu'est « un grand nombre de neurones », ce nombre dépendant à la fois de la dimension du problème traité et surtout de sa nature.

Néanmoins, lors de l'application de cette méthode, l'utilisateur s'aperçoit souvent que les résultats enregistrés sur la base d'apprentissage sont significativement différents de ceux enregistrés sur des données n'ayant pas servi à la modélisation. Ce problème connu sous le nom de surapprentissage (*overfitting* en anglais) est décrit en détail chapitre 5 section 5.3. Pour répondre correctement à la deuxième question, il est donc nécessaire de limiter la dimension du vecteur des paramètres.

Afin de réduire ce risque de mauvaise adéquation entre la modélisation et le phénomène réel, il existe plusieurs techniques plus ou moins fondées théoriquement.

Citons trois différentes techniques largement utilisées dans le domaines des réseaux de neurones :

1. La technique consistant à stopper l'algorithme d'optimisation employé pour estimer les paramètres lorsque le surapprentissage survient sur des données n'appartenant pas à la base d'apprentissage. Cette technique est connue sous le nom

de *validation croisée* (les dénominations anglaises étant *cross-validation* ou *early stopping*).

2. La technique consistant à introduire un terme de « régularisation » dans la fonction de coût utilisée lors de l'étape d'estimation des paramètres afin de restreindre progressivement l'espace du vecteur des paramètres dans un voisinage de 0.

3. La technique d'*élagage* qui consiste à éliminer des paramètres sans véritable utilité pour la modélisation, plus connue dans un contexte neuronal sous l'appellation anglaise de *pruning*.

Ces trois techniques sont sensiblement différentes. La première nécessite en particulier un partage des données disponibles entre une base d'apprentissage et une base de validation. Si ces données sont en nombre trop faible, ce partage peut entraîner une perte d'information non-négligeable. Notons aussi qu'elle n'est pas fondée théoriquement, et que différents partages des données entraînent souvent différents résultats de modélisation. Cette méthode ne semble donc pas d'un très grand intérêt, et on ne l'étudie pas dans ce mémoire.

La technique de régularisation est basée sur l'ajout à la fonction de coût d'un terme de pénalisation. Ainsi on écrit la fonction de coût sous la forme d'une combinaison linéaire :

$$\text{Fonction de coût} = \text{Erreur Quadratique} + \alpha \text{Pénalisation} \quad .$$

L'idée est de favoriser les modèles de complexité faible :(un modèle linéaire sera considéré comme de faible complexité). Le terme α est vu comme un coefficient pondérant le terme de régularisation. Plus ce terme est grand, et plus on considère qu'il est important d'avoir un modèle de complexité faible. On utilise généralement des heuristiques pour régler ce coefficient. Le terme de pénalisation qui a l'adhésion de bon nombre d'utilisateurs est la somme des carrés des poids: $\sum_i \theta_i^2$. Utiliser ce terme favorise la convergence vers une solution où les pa sont « petits ». En fait, ajouter ce terme revient a faire l'*hypothèse implicite* que la distribution des vrai paramètres est gaussienne et centrée autour de 0. Cette hypothèse peut paraître forte et surprenante dans certains cas. Nous n'en dirons pas plus ici. Signalons simplement les travaux de (MacKay, 1992a; MacKay, 1992d; MacKay, 1992c; MacKay, 1992b; MacKay, 1993) qui propose un cadre statistique des *techniques de régularisation* basé sur la théorie bayesienne.

La technique d'élagage est une méthode qui existe déjà dans la littérature traitant des problèmes de régression linéaire. C'est une méthode de *stepwise* descendant qui tend à diminuer la dimension du vecteur des paramètres. Dans le domaine des modèles neuronaux, plusieurs heuristiques sont proposées. Ainsi, (Mozer et Smolensky, 1989) suggèrent une méthode de détermination des poids et unités cachées significatifs, alors que (LeCun, 1989; LeCun et al., 1990) proposent une méthode d'élimination des poids d'un perceptron multicouches fortement connecté, basé sur le calcul de la matrice des dérivées secondes de la fonction de coût en fonction des paramètres. Pour un compte rendu sur cette technique voir (Reed, 1993), (Fambon et Jutten, 1994) et (Gorodkin et al., 1993). Une technique d'élagage statistique originale, appelée *SSM* (*Statistical Stepwise*

Method), basée sur les résultats théoriques du chapitre 3, fait l'objet de la section 6.2. On y décrit en particulier le cadre théorique, quelques applications et une comparaison avec l'algorithme « *Optimal Brain Damage* » de (LeCun et al., 1990).

Les méthodes d'élagage, si elles sont utilisées rigoureusement, semblent intéressantes pour diminuer, à partir d'une architecture donnée, le nombre de paramètres (voire de neurones cachés et de variables d'entrée). Néanmoins le problème du choix de l'architecture initiale reste entier. Une solution à ce problème sera proposée chapitre 8.

6.2 Cadre théorique de la méthode d'élimination de poids synaptiques : *SSM*

6.2.1 Consistance et normalité asymptotique de l'estimateur d'un modèle NAR$_n(p)$

Reprenons les notations et le résultat du théorème 5.3 du chapitre 5. À partir de ces résultats, il est possible, comme nous l'avons vu dans le chapitre 3, en renforçant un peu les hypothèses, d'obtenir la normalité asymptotique de l'estimateur des moindres carrés du vecteur de paramètres. Notre but ici est de reprendre ces résultats, et de les appliquer à l'estimateur des moindres carrés d'un modèle neuronal basé sur un perceptron multicouches (ce modèle est noté NAR$_n(p)$ si celui-ci comporte p entrées et n neurones sur la couche cachée).

Rappelons que $\theta = (\theta_1, \theta_2, \ldots, \theta_\ell)$, $\ell \geq 1$, est le vecteur des paramètres ou, pour un modèle neuronal, le vecteur des ℓ poids synaptiques pris dans un ordre quelconque. Les notations suivantes sont les mêmes que dans les chapitres précédents : θ_0 est le vecteur des « vrais paramètres » correspondant au modèle optimal, $\hat{\theta}$ est l'estimateur des moindres carrés. On suppose que le modèle est du type de la définition 4.8. En particulier, en posant $X_t^{(p)} = (X_t, X_{t-1}, \ldots, X_{t-p+1})$, la série $(X_t^{(p)})$ est une chaîne de Markov homogène dans $\mathbb{R}^p$.

Théorème 6.1 :

Soit un modèle $NAR_n(p)$ correspondant à la définition 2.8 avec des fonctions d'activation sigmoïdes. Supposons que soit satisfait l'ensemble **[P]** *des conditions suivantes :*

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

(ii) ε_1 *a une densité positive par rapport à la mesure de Lebesgue avec* $E(\varepsilon_1) = 0$ *et* $E(\varepsilon_1^6) < \infty$.

(iii) θ *appartient à un sous-ensemble compact* Θ *de* $\mathbb{R}^\ell$, *tel que* $\theta_0 \in \overset{\circ}{\Theta}$.

(iv) (Condition d'identifiabilité) pour tout θ, $\theta \neq \theta_0$, $f_\theta \neq f_{\theta_0}$ *c'est-à-dire qu'il existe un* $\mathbf{x} \in \mathbb{R}^p$ *tel que* $f_\theta(\mathbf{x}) \neq f_{\theta_0}(\mathbf{x})$.

(v) La matrice $\ell \times \ell$

$$\Sigma_0 = \int_{\mathbb{R}^p} \left[\frac{\partial}{\partial\theta_i} f_\theta(\mathbf{x}) \frac{\partial}{\partial\theta_j} f_\theta(\mathbf{x})\right]_{1\leq i,j\leq \ell} \mu_{\theta_0}(d\mathbf{x}), \tag{6.1}$$

est définie positive.

Alors,

(a) L'estimateur des moindres carrés $\hat{\theta}_T$ *est fortement consistant.*

(b) Pour n'importe quelle loi initiale de la chaîne de Markov $X^{(p)}$,

$$\sqrt{T}\left[\hat{\theta}_T - \theta_0\right] \xrightarrow{\mathbf{CL}} \mathcal{N}(0, \sigma^2\Sigma_0^{-1}) \quad .$$

Preuve.

La démonstration consiste à vérifier que les conditions **[D]**, **[M]** et **[N]** relatives aux modèle fonctionnel ARF$_d(p)$ (avec $d = 1$) des théorèmes 3.10 et 3.12 sont satisfaites.

Les conditions **[P]**-*(i)*-(ii)-*(iv)* sont les mêmes que les conditions **[R]**-*(i)*-(ii)-*(iv)* du théorème 5.3, et la condition **[P]**-*(iii)* est plus restrictive que la condition **[R]**-*(iii)*. De par la démonstration du théorème 5.3, les conditions **[M]** sont donc vérifiés, ainsi que les conditions **[D]**.

Conditions [N] : En considérant $V = \overset{\circ}{\Theta}$, avec Θ compact, et en remarquant que f_θ est $\mathcal{C}^\infty$, **[N]**-*(i)* est facilement vérifiée. Il est de même aisé de montrer, puisque les fonctions sigmoïdes ainsi que toutes leurs dérivées par rapport à x sont bornées (lemme 4.7), que pour tout $\mathbf{x} \in \mathbb{R}^p$:

$$\begin{aligned} |D_i f_{\theta_0}(\mathbf{x})| &\leq \text{Cste}(1+|\mathbf{x}|), \quad i=1,2,\ldots,\ell \\ |D^2_{ij} f_{\theta_0}(\mathbf{x})| &\leq \text{Cste}(1+|\mathbf{x}|^2), \quad i,j=1,2,\ldots,\ell \\ |D^2_{ij} f_{\theta}(\mathbf{x}) - D^2_{ij} f_{\theta_0}(\mathbf{x})| &\leq \text{Cste} \parallel \theta - \theta_0 \parallel (1+|\mathbf{x}|^3), \quad i=1,2,\ldots,\ell \quad . \end{aligned}$$

Le contrôle du moment d'ordre 6 de (ε_t) (condition **[P]**-*(ii)*) et ces dernières inégalités satisfont les conditions **[N]**-*(i)-(ii)*, qui entraîne finalement le théorème 6.1.

∎

À noter que ces résultats permettent de construire des intervalles de confiance et des tests de nullité des paramètres, qui sont au coeur de l'algorithme d'élimination des poids décrit plus loin. On rappelle que $S_T(\theta)$ (voir équation 5.2) est la somme des erreurs quadratiques sur une série de longueur $T+p$. On estime σ^2, la variance du bruit $(\varepsilon_t)_{t>0}$, par $\hat{\sigma}^2$ la variance résiduelle :

$$\hat{\sigma}^2 = \frac{1}{T} S_T(\hat{\theta}_T) \quad , \tag{6.2}$$

et Σ_0 par :

$$\widehat{\Sigma}_0 = \frac{1}{2T} \nabla^2 S_T(\hat{\theta_T}) \tag{6.3}$$

Une approximation asymptotique de Σ_0 est aussi donnée par (voir par exemple (Seber et Wild, 1989)) : $\frac{1}{T}\sum_t [\nabla f_{\hat{\theta}_T}(X_t^{(p)})]^t [\nabla f_{\hat{\theta}_T}(X_t^{(p)})]$ .Signalons enfin que si une méthode du second ordre est utilisée pour estimer les paramètres, la matrice Hessienne inverse Σ_0^{-1} est déjà calculée et est directement utilisable.

6.2.2 Identification presque sûre du « vrai modèle »

En s'appuyant sur l'étude décrite chapitre 3, le théorème énoncé précédemment entraîne que l'estimateur $\hat{\theta}_T$ vérifie une loi du logarithme itéré. Ainsi, en appliquant les résultats sur la sélection de modèles de (Senoussi, 1990); (Guyon, 1992), on peut établir une identification presque sûre du vrai modèle, en supposant que l'on dispose d'un modèle dominant commun à un nombre fini de modèles candidats.

Plus précisément, reprenons les notations relatives à l'équation 4.6 du chapitre 4 et supposons que la dimension des modèles candidats soit bornée par un entier strictement positif ℓ (hypothèse $(\mathbf{H_1})$). On suppose donc que $\Theta \subset \mathbb{R}^\ell$ et on note $M_{\max}$ le modèle dominant dont le vecteur des paramètres, de dimension ℓ, est noté $\theta_{\max}$ (voir figure 6.1) :

$$\theta_{\max} = \{(\alpha_j)_{0\leq i\leq n}, (\beta_{ij})_{0\leq i\leq p, 1\leq j\leq n}\}$$

avec $\ell = n \times (p+2) + 1$.

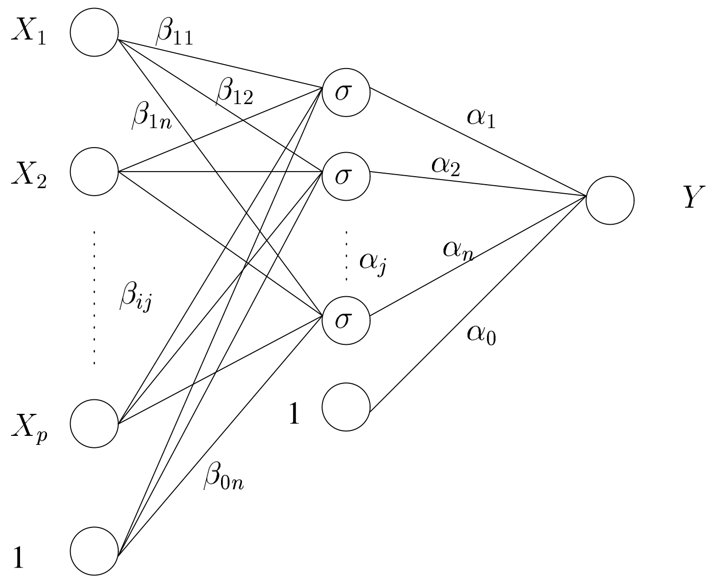


FIG. 6.1 – *Réseau de neurones du type perceptron multicouches. On utilise les notations relatives à l'équation 4.6 du chapitre 4.*

Considérons la famille finie de modèles :

$$\mathcal{M} = \{\theta = (\theta_1, \theta_2, \dots, \theta_\ell)/\text{il existe un ou plusieurs } i \in \{1, 2, \dots, \ell\} \text{ tel que } \theta_i = 0\}, \tag{6.4}$$

qui respecte les *conditions de cohérence* du modèle $\text{NAR}_n(p)$ (voir définition 5.1).

Soit $M \in \mathcal{M}$ un sous-modèle de $M_{\max}$. On note $d(M)$ le nombre de composantes non-nulles du vecteur de paramètres associé θ, i.e. la dimension de θ, et Θ_M l'ensemble des valeurs possibles de θ, $\Theta_M \subset \Theta$. On note aussi M_0, le vrai modèle associé au vecteur des vrais paramètres θ_0 de dimension $d(M_0)$. Conformément à l'hypothèse $(\mathbf{H_1})$ on a alors $M_0 \in M_{\max}$ et on suppose de plus que M_0 est minimal: si $M \subset M_0$, et $M \neq M_0$, alors $\theta_0 \notin M$.

Soit $\hat{\theta}_{T,M}$ l'estimateur des moindres carrés de θ vecteur des paramètres du modèle M :

$$\hat{\theta}_{T,M} = \text{Arg } \min_{\theta \in \theta_M} S_T(\theta) \quad .$$

Soit $(c(t))_{t\in\mathbb{N}}$ une suite positive croissante dans $\mathbb{R}$. Alors le *contraste des moindres carrés pénalisé* de vitesse de pénalisation $(c(t))$ est de la forme :

$$\text{CP}(T, M) = \frac{S_T(M)}{T} + \frac{c(T)}{T} d(M). \tag{6.5}$$

On note aussi $\hat{M}_T = \text{Arg} \min_{M \in \mathcal{M}} \text{CP}(T, M)$ le modèle estimé, résultat de deux minimisations successives pour un T fixé :

- une minimisation sur $\mathbb{R}^{d(M)}$ pour calculer $\hat{\theta}_{T,M}$,

- une minimisation sur l'espace fini $\mathcal{M}$ pour calculer $\hat{M}_T$.

En utilisant le théorème 3.15 énoncé chapitre 2, on a alors le résultat suivant :

Théorème 6.2 :

Supposons les conditions du théorème 6.1 remplies. Supposons de plus que la vitesse de pénalisation $c(T)$ est telle que

$$\lim_T \frac{c(T)}{T} = 0, \quad \textit{et} \quad \liminf_T \frac{c(T)}{2\,\ell n\,\ell n\, T} > \sigma^2 \frac{\Lambda}{\lambda} \tag{6.6}$$

où Λ (resp. λ) est la plus grande (resp. la plus petite) valeur propre de la matrice Σ_0.

Alors le couple $(\hat{M}_T, \hat{\theta}_{T,\hat{M}_T})$ converge presque sûrement vers le vrai modèle et le vrai vecteur des paramètres (M_0, θ_0).

Soit γ une constante positive (du même ordre de grandeur en pratique que la variance σ^2). Un terme de pénalisation logarithmique tel que $c(t) = \gamma\,\ell n\, t$ satisfait clairement l'ensemble des conditions du théorème 6.2.

Prendre une telle vitesse de pénalisation conduit alors à un critère de sélection de modèle du type :

$$\mathrm{BIC}^* = \mathrm{BIC}^*(T, M) = \frac{S_T(M)}{T} + \frac{\gamma\,\ell n\, T}{T} d(M) \tag{6.7}$$

On peut noter la différence entre ce critère et le critère habituel BIC :

$$\mathrm{BIC} = \mathrm{BIC}(T, M) = \ell n\, \frac{S_T(M)}{T} + \frac{\ell n\, T}{T} d(M). \tag{6.8}$$

Ces deux critères comprennent un terme de pénalisation logarithmique en T mais le critère BIC, basé sur la méthode du maximum de vraisemblance, comprend un premier terme qui est le logarithme du premier terme du critère BIC*.

En utilisant les résultats des théorème 6.1 et 6.2, on est maintenant en mesure de proposer une méthodologie pour

- trouver un modèle *dominant*,

- réduire la dimension du vecteur des paramètres afin de déterminer le *vrai modèle* à l'aide d'une méthode d'élimination des connexions.

6.2.3 Recherche du modèle dominant

Afin de déterminer un modèle de complexité suffisante, susceptible de modéliser correctement la dynamique du phénomène, on utilise la méthodologie suivante :

1) On emploie toutes les variables (et leurs retards) supposés être corrélés à la variable étudiée.

2) On utilise un modèle neuronal avec une unité sur la couche cachée et on initialise les poids aléatoirement suivant une loi uniforme par exemple d'intervalle [-1,1] ; on estime les paramètres avec l'une des méthodes exposées chapitre 5, puis on calcule le critère BIC* de l'équation 6.7.

3) On ajoute une unité à la couche cachée et on initialise les poids correspondants aléatoirement.

4) On estime de nouveau les paramètres puis on calcule le critère BIC* de l'équation 6.7.

5) Si la différence entre deux critères BIC* successifs est strictement positive on revient au 3). Sinon, on arrête la recherche et le dernier modèle est considéré comme le *modèle dominant*.

Ce simple algorithme fonctionne assez bien pour des processus peu complexes. Partir d'un modèle dominant dont la structure est bien adaptée peut faciliter grandement l'étape d'estimation et celle de l'identication. On propose au chapitre 8 une méthode de recherche de modèle plus élaborée que celle présentée ci-dessus. Cette méthode implique en particulier des modèles neuronaux à propagation directe sans notion de couches, de structure plus complexes que les modèles à couches utilisés dans cette exemple.

6.3 Algorithme d'élimination des poids (*SSM*)

Supposons que l'on dispose d'un modèle dominant $M_{\max}$, déterminé par exemple en utilisant la technique exposée dans la section précédente. Et rappelons que $\theta_{\max}$, le vecteur des paramètres associé, est de dimension ℓ.

En théorie, dans le but d'évaluer le vrai modèle, on pourrait tenter d'explorer exhaustivement l'ensemble des sous-modèles $M \in \mathcal{M}$ en calculant pour chacun d'eux le critère BIC*. Le nombre de modèles candidats est cependant exponentiellement grand (de l'ordre de 2^{ℓ}). On propose donc une méthode statistique d'élimination originale selon un principe connu dans le domaine linéaire sous l'appelation *stepwise descendant*. Cette stratégie de recherche du meilleur modèle est basé sur le résultat du théorème 6.1 (normalité asymptotique de l'estimateur des moindres carrés $\hat{\theta}_T$). Cette méthode, appelé

Statistical Stepwise Method (*SSM*), a fait l'objet de plusieurs articles dont (Mangeas et al., 1993), (Muller et al., 1994) et (Cottrell et al., 1995)).

La technique est en fait basée sur un test de nullité des paramètres. Tenter d'éliminer la $i^{ème}$connexion consiste alors à tester l'hypothèse « $\theta_i = 0$ » contre l'hypothèse alternative « $\theta_i \neq 0$ » suivant un test de Student (en fait un test gaussien puisque que T est grand).

Effectuer une succession d'éliminations de poids $\theta_{k_1}, \theta_{k_2}, \ldots, \theta_{k_L}$ consiste à tester itérativement $\theta_{k_1} = 0, \theta_{k_1} = \theta_{k_2} = 0, \ldots$, et donc à construire une série de L modèles $M_{\max}, M_{\max}^{k_1}$, $M_{\max}^{k_1,k_2}, \ldots$, $M_{\max}^{k_1,k_2,\ldots,k_L}$. La difficulté ici est de calculer le niveau des tests et nous utiliserons simultanément le test de Student et le critère de sélection BIC*pour s'assurer d'une bonne convergence de l'algorithme.

Étudions plus en détail le calcul du test d'élimination du $i^{ème}$poids. Soit M le modèle courant et M^i le sous-modèle obtenu en annulant le $i^{ème}$poids θ_i. La différence des deux critères de selection s'écrit (en gardant les notations de la section 6.2.2):

$$\mathrm{BIC}^*(M^i) - \mathrm{BIC}^*(M) = \frac{1}{T}\left\{S_T(M^i) - S_T(M)\right\} - \frac{\gamma\,\ell n T}{T}. \tag{6.9}$$

Pour diminuer autant que possible le critère BIC*, il faudrait déterminer l'indice $i \in \{1, 2, \ldots, \ell\}$ qui minimise le plus $S_T(M^i) - S_T(M)$, et donc calculer tous les $S_T(M^i)$ possibles. On peut cependant faire l'économie de l'évaluation de l'ensemble des modèles candidats M^i en effectuant les calculs des statistiques $(Q_i)_{i\in\{1,2,\ldots,\ell\}}$ utilisées pour tester « $\theta_i = 0$ » contre « $\theta_i \neq 0$ ». On a :

$$Q_i = \frac{\hat{\theta}_i}{\hat{\sigma}(\hat{\theta}_i)} \tag{6.10}$$

où

$$\hat{\sigma}(\hat{\theta}_i) = \frac{\hat{\sigma}}{\sqrt{T}}\sqrt{(\widehat{\Sigma}_0^{-1})_{i,i}} \tag{6.11}$$

représente l'écart-type de $\hat{\theta}_i$ et $\widehat{\Sigma}_0$ est défini équation 6.3. Notez que Q_i est connu dès que $\hat{\theta}_{T,M}$ est estimé dans le modèle M suivant les équations 6.2 et 6.3, sans réestimer les paramètres du sous-modèle M^i. On sait aussi que la statistique de Fisher du test de M^i contre M s'écrit :

$$Q_i^2 = \frac{S_T(M_i) - S_T(M)}{S_T(M)/(T - d(M))}. \tag{6.12}$$

Mais, comme habituellement dans le cas asymptotique, $S_T(F)/(T - m(F))$ converge presque sûrement vers σ^2 et peut être considéré comme constant pour un T grand. Ainsi, maximiser BIC*, ou encore minimiser la différence $S_T(M^i) - S_T(M)$, revient à annuler le poids θ_i tel que Q_i^2 soit minimum.

L'algorithme complet s'écrit donc:

1. Chercher un modèle dominant $M_{\max}$, de la manière décrite section 6.2.3 et estimer les poids. On note K le nombre de poids.

2. Calculer, pour chaque poids d'indice $i \in \{1, 2, \dots, K\}$, le quotient $Q_i = \hat{\theta}_i / \hat{\sigma}(\hat{\theta}_i)$ correspondant (voir équation 6.10).
3. Déterminer l'indice $k \in \{1, 2, \dots, K\}$ réalisant le minimum de ces quotients tout en maintenant la cohérence du réseau (voir définition 5.1).
4. Éliminer la $k^{ème}$connexion et réestimer les paramètres (refaire un apprentissage en partant des poids du modèle précédent).
5. Accepter l'élimination de θ_k ssi le critère BIC* décroît.
6. En cas de rejet de l'hypothèse nulle, arrêter l'élimination et garder le meilleur modèle. En cas d'acceptation, réordonner les poids pour que les indices soient compris entre 1 et $K-1$, poser $K := K-1$ et revenir à l'étape 2.

Ce type d'algorithme fait partie de la famille des *stepwises descendant*, largement utilisés dans le domaine de la régression linéaire (Draper et Smith, 1981)). La méthode est liée à celle mise au point par (LeCun et al., 1990) (appelée « *Optimal Brain Damage* ») dont la technique d'élimination est une version simplifiée de celle employée dans l'algorithme *SSM*. Ainsi, reprenons les notations employées précédemment et considérons que l'on dispose d'un modèle dominant $M_{\max}$ d'un problème donnée. Après l'étape d'estimation, l'idée sous-jacente à la méthode *OBD* est d'éliminer les paramètres dont la variation perturbe le moins la fonction de coût $S_T(\hat{\theta}_{T,M_{\max}})$. Pour cela, on calcule le développement de Taylor de $S_T(\theta)$ autour du point $\hat{\theta}_{T,M_{\max}}$. On a, pour $h, \theta \in \mathbb{R}^\ell$ tel que $\theta + h \in \Theta$:

$$\begin{aligned} S_T(\theta+h) - S_T(\theta) &= {}^t h \, \frac{\partial S_T(\theta)}{\partial \theta} + \frac{1}{2} {}^t h \, \frac{\partial S_T(\theta)}{\partial \theta} \, h + O(\| h \|^3) \\ &= \sum_{i=1}^{\ell} h_i \, \frac{\partial S_T(\theta)}{\partial \theta_i} + \frac{1}{2} \sum_{i=1}^{\ell} h_i^2 \, \frac{\partial^2 S_T(\theta)}{\partial \theta_i^2} + \sum_{i,j=1/i \neq j}^{\ell} h_i h_j \, \frac{\partial^2 S_T(\theta)}{\partial \theta_i \partial \theta_j} + O(\| h \|^3) \quad . \end{aligned}$$

En prenant $\theta = \hat{\theta}_{T,M_{\max}}$ les termes de dérivées premières disparaissent puisque ce vecteur des paramètres est un minimum de $S_T(\theta)$. De plus, en posant les hypothèses suivantes :

- $S_T(\theta)$ est presque quadratique en θ et le terme $O(\| h \|^3)$ n'est pas significatif,
- La matrice Hessienne est diagonale (les termes croisés sont nuls),

on arrive à une expression simplifiée de l'influence de la variation :

$$S_T(\theta) - S_T(\theta+h) = \frac{1}{2} \sum_{i=1}^{\ell} h_i^2 \frac{\partial^2 S_T(\theta)}{\partial \theta_i^2} \quad .$$

Tenter d'éliminer la $i^{ème}$connexion est en fait équivalent à annuler le paramètre associé θ_i. Ceci peut être fait en prenant une variation h telle que $h_i = -\theta_i$ et $h_j = 0$ pour tout

$j \neq i$, $j \in \{1, 2, \ldots, \ell\}$. On arrive alors à la définition d'un critère d'élimination pour le $i^{ème}$poids, appelé *saliency*, et défini par :

$$s_i = \theta_i^2 \; \frac{\partial^2 S_T(\theta)}{\partial \theta_i^2} \quad .$$

La technique consiste alors, suivant un protocole proche de celui de *SSM*, à éliminer successivement les poids dont les critères $(s_i)_{i=1,2,\ldots,\ell}$ sont les plus petits. Il est alors facile de vérifier que, si l'hypothèse de la matrice Hessienne diagonale est exacte, le terme s_i est strictement équivalent au terme Q_i défini équation 6.10. Cette hypothèse apparaît cependant très forte et non fondée dans la grande majorité des cas.

L'avantage de l'algorithme *SSM*, basé sur des résultats statistiques, par rapport à des algorithmes tel que *OBD*, basé sur des résultats numériques, tient en fait au contrôle de l'élimination et à la détermination d'un critère d'arrêt :

- En utilisant les résultats sur la normalité asymptotique de l'estimateur des moindres carrés du vecteur des paramètres, on dispose d'une « mesure » statistique de l'utilité du paramètre dans la modélisation, sa « significativité ».

- En utilisant le résultat sur l'identification presque sûre du modèle neuronal, on dispose d'un critère d'arrêt de l'algorithme d'élimination fiable et fondé.

Il est à noter enfin que certaines recherches vont dans le sens d'une méthode de construction de l'architecture du type « *stepwise ascendant* » basée sur les mêmes principes théoriques (voir (Jutten et Chentouf, 1995)).

6.4 Simulations

Les simulations suivantes ont pour double intérêt de démontrer (par l'exemple) que l'algorithme élimine correctement les connexions (voire les neurones) qui n'appartiennent pas à un modèle donné, et de confirmer le fait que certaines séries ayant des caractéristiques non-linéaires sont mieux ajustées avec des perceptrons multicouches que par des modèles ARMA.

6.4.1 Simulation 1

Pour obtenir les conditions de convergence asymptotique, la série simulée contient 1000 point pour un nombre de paramètres égal à 7. La perturbation aléatoire est caractérisée par un bruit blanc ε_t gaussien.

Le modèle s'écrit :

$$\left\{\begin{array}{lcl} X_t & = & f^1_{\theta^*}(X^{(2)}_{t-1}) + \varepsilon_t \\ & = & \theta^*_1 \tanh(\theta^*_2 X_{t-1} + \theta^*_3) + \theta^*_4 \tanh(\theta^*_5 X_{t-2} + \theta^*_6) + \theta^*_7 + \varepsilon_t \\ \varepsilon_t & \sim & \mathcal{N}(0, 1/4) \end{array}\right. \tag{6.13}$$

$$\text{avec} \left\{\begin{array}{rrrr} \theta^*_1 = & 1 & \theta^*_2 = & 0.7 \\ \theta^*_3 = & -1.3 & \theta^*_4 = & 1 \\ \theta^*_5 = & 0.9 & \theta^*_6 = & -0.4 \\ \theta^*_7 = & 1 & & \end{array}\right.$$

Les retards utilisés étant d'ordre 2, on peut classer cette série dans les modèles NAR(2). Voir la figure 6.2 pour une représentation graphique du réseau de neurones associé. Les valeurs des poids synaptiques sont indiquées sur les connexions. La fonction de transfert de chaque neurone de la couche cachée est la fonction $x \mapsto \tanh(x)$; la fonction de transfert est linéaire pour le neurone de sortie, et il est à noter que l'activité des neurones de la couche cachée est fortement non-linéaire. La sortie au temps t, la sortie du réseau $f^1_{\theta^*}(X^{(2)}_{t-1})$ est bruitée par une perturbation aléatoire ε_t avant d'être injectée dans la couche d'entrée pour calculer la valeur suivante.

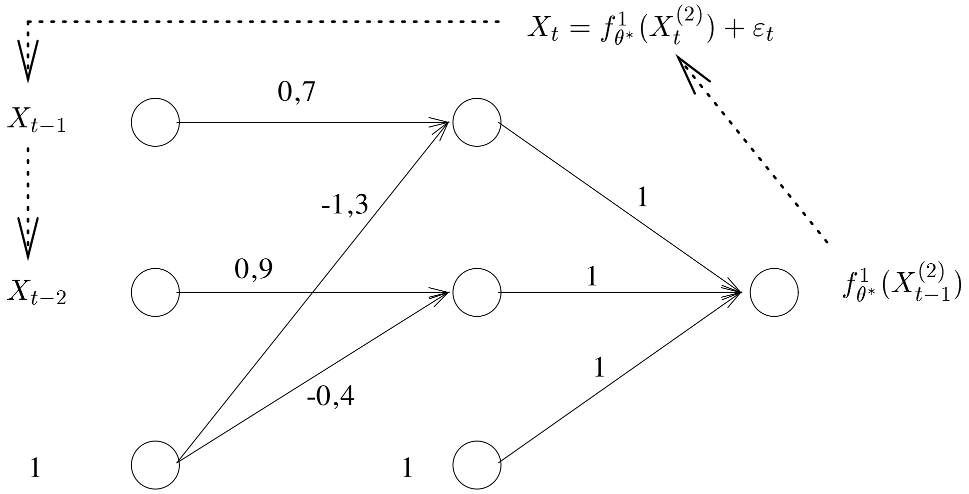


FIG. 6.2 – *Simulation 1. Réseau correspond au modèle de l'équation 6.13.*

A partir d'un réseau dominant et surparamétré par rapport à ce modèle, on estime les paramètres et on applique l'algorithme d'élimination des connexions superflues *SSM*.

Le réseau dominant choisi (voir figure 6.3) comporte 16 connexions ; la première couche et la couche cachée comportent 3 neurones plus un neurone de seuil. On remarque que la variable caractérisant le retard d'ordre 3 est inutile dans le modèle simulé décrit précédemment, ainsi que l'un des neurones de la couche cachée et certaines connexions.

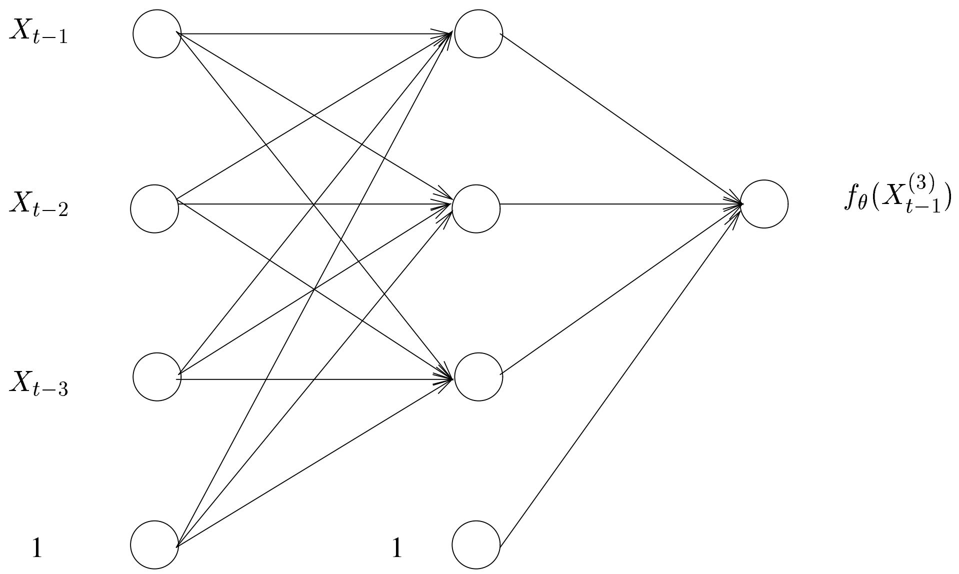


FIG. 6.3 – *Simulation 1. Réseau dominant surparamétré, modèle de départ avant élimination des connexions inutiles au problème de prévision.*

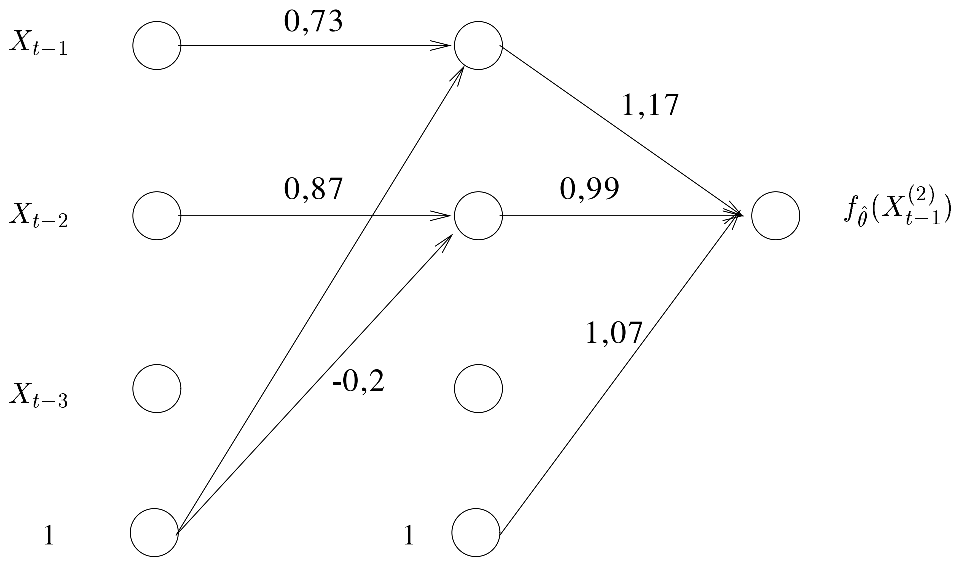


FIG. 6.4 – *Simulation 1. Réseau obtenu après application de l'algorithme* SSM.

Après initialisation aléatoire des poids synaptiques, on applique l'algorithme d'élimination SSM expliqué section 6.3.

Dans ce cas précis, la partie déterministe du processus simulé décrit équation 6.13 est non-linéaire et calculée à l'aide de tangentes hyperboliques, parfaitement reproductible par un réseau de neurones de ce type. Il est à signaler qu'a priori, il n'y a pas unicité de la solution et que l'on peut sans doute trouver d'autres combinaisons linéaires de tangentes hyperboliques répondant au problème. On peut de plus obtenir des valeurs de poids ayant des signes différents mais de même valeur absolue, sachant qu'avec les multiplications successives opérées, on peut trouver une même sortie de réseau. Dans tous les cas, cela dépend fortement des valeurs aléatoires d'initialisation. Le réseau obtenu pour un jeu de poids initialisés aléatoirement suivant une loi uniforme dans l'intervalle [-1,1] est représenté figure 6.4.

On peut alors faire un certain nombre de constatations :

- Constatation n°1 : On retrouve l'architecture ayant servi à construire la série simulée. Le neurone supplémentaire de la couche cachée disparaît, ainsi que les connexions superflues.

- Constatation n°2 : La variable X_{t-3} qui n'apparaissait pas dans le modèle initial a disparu.

- Constatation n°3: La variance résiduelle $\sum_{i=1}^{n}(X_t - f_{\hat{\theta}}(X_{t-1}^{(2)}))^2/1000)$est égale à 0,247. Elle est donc pratiquement équivalente à celle du bruit blanc ayant servi à la simulation (0,25), et les résidus $(X_t - f_{\hat{\theta}}(X_{t-1}^{(2)})_{1<t<1000})$ sont pratiquement égaux terme à terme avec les valeurs du bruit blanc. Le nombre de paramètres étant équivalent, ainsi que la variance, les valeurs des critères BIC* associés au modèle simulé et au modèle trouvé sont égales.

Cette expérience a été menée 100 fois de suite avec des poids initiaux différents et les remarques citées ci-dessus sont vraies dans 95 % des cas. Pour les 5 % restants, les constatations (2) et (3) restent vraies, seule l'architecture comporte plus de connexions que l'architecture initiale. L'écart-type des variances résiduelles des différents modèles obtenus est égal à 0,008 et la moyenne est égale à 0,248.

Essai de modélisation ARMA : On a tenté de modéliser à l'aide de processus ARMA, le processus simulé décrit équation 6.13. Le logiciel *Mandrake* (Azencott et al., 1991), intègre une option de modélisation automatique, adéquate pour ce genre de problème. Alors que le modèle neuronal obtenu par l'algorithme SSM obtient un critère BIC* de 9.64, le meilleur modèle linéaire (un AR(2)) obtient un BIC* de 10,6. La différence, due évidemment aux fortes non-linéarités inhérentes aux processus simulés est significative.

6.4.2 Simulation 2

En suivant le même protocole de construction du processus que celui de la simulation 1, on tente de modéliser un processus autorégressif d'ordre 2. La perturbation aléatoire est caractérisée par un bruit blanc ε_t gaussien.

Le modèle linéaire s'écrit :

$$\left\{ \begin{array}{lcl} X_t & = & f^2_{\theta^*}(X^{(2)}_{t-1}) + \varepsilon_t \\ & = & \theta^*_1 X_{t-1} + \theta^*_2 X_{t-2} + \theta^*_3 + \varepsilon_t \\ \varepsilon_t & \sim & \mathcal{N}(0,1/4) \end{array} \right. \tag{6.14}$$

avec $\{\theta^*_1 = 0,7 \; ; \; \theta^*_2 = -0.3 \; ; \; \theta^*_3 = 0.9\}$.

Les retards utilisés étant d'ordre 2, on peut classer cette série dans les modèles AR(2). Voir la figure 6.5 pour une représentation graphique du réseau de neurones associé. La fonction de transfert est linéaire pour le neurone de sortie et ce réseau représente donc exactement le modèle autorégressif de l'équation 6.13. La sortie au temps t, la sortie du réseau $f^2_{\theta^*}(X^{(2)}_{t-1})$ est bruitée par une perturbation aléatoire ε_t avant d'être injectée dans la couche d'entrée pour obtenir la valeur suivante.

Le réseau dominant initial est décrit figure 6.6. Il comporte 2 neurones cachés munis de fonctions d'activation sigmoïdes. Après avoir initialisées aléatoirement les valeurs des poids suivant une loi uniforme d'intervalle [-1,1], on estime les paramètres et on applique l'algorithme d'élimination des connexions SSM. Le réseau obtenu est représenté figure 6.7.

On remarque que la fonction d'activation travaille essentiellement dans un voisinage proche de 0, ce qui implique que le modèle est presque linéaire. La variance résiduelle obtenue : $\sum_{i=1}^{n}(X_t - f_{\hat{\theta}}(X^{(2)}_{t-1}))^2/1000)$est égale à 0,253 ce qui donne une différence de 1,2% avec la variance réelle.

Cette expérience à été menée 100 fois de suite avec des poids initiaux différents et les remarques citées ci-dessus sont vraies dans 97 % des cas. Pour les 3 % restants, seule l'architecture comporte plus de connexions que l'architecture initiale. L'écart-type des variances résiduelles des différents modèles obtenus est égal à 0,005 et la moyenne est égale à 0,249.

6.4.3 Simulation 3

Il est évident que s'il existe dans « la nature » des séries avec des particularités non-linéaires, il n'y a aucune raison que celles-ci soient construites à base de tangentes hyperboliques. Néanmoins, les capacités d'approximateur universel des perceptrons multicouches assurent théoriquement qu'il est possible d'approcher n'importe quel type de

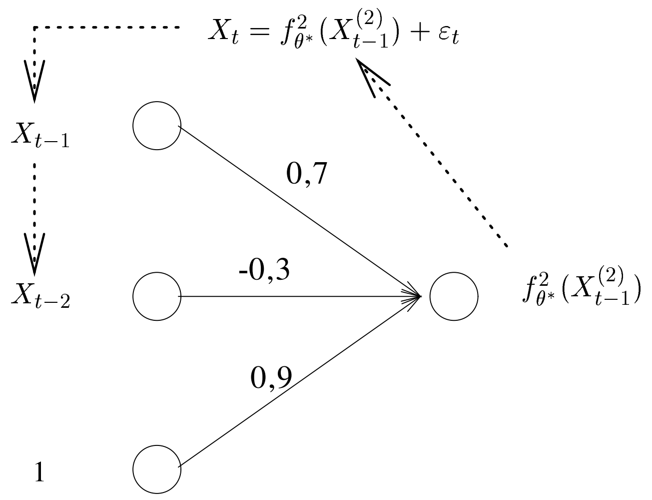


FIG. 6.5 – *Simulation 2. Réseau correspond au modèle de l'équation 6.14.*

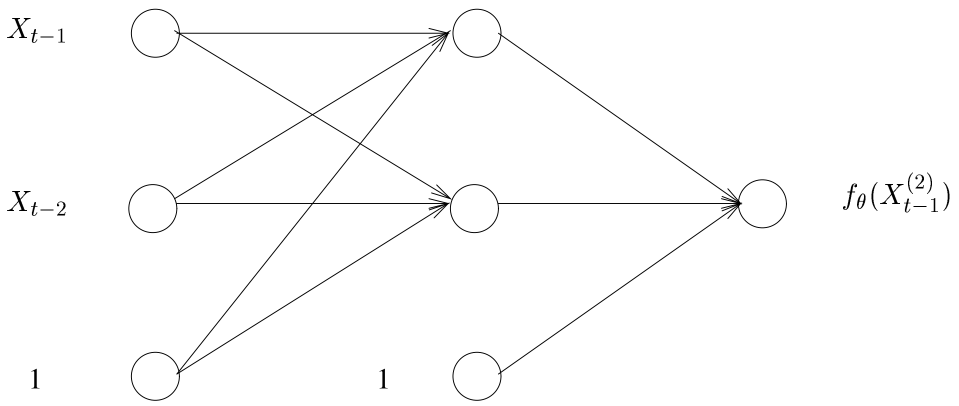


FIG. 6.6 – *Simulation 2. Réseau dominant surparamétré, modèle de départ avant élimination des connexions inutiles au problème de prévision.*

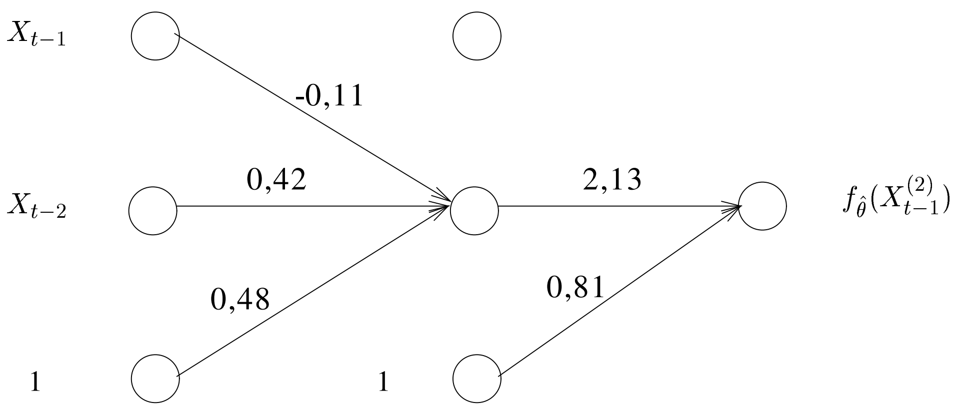


FIG. 6.7 – *Simulation 2. Réseau obtenu après application de l'algorithme* SSM.

dynamique. L'objectif de cette deuxième simulation a donc pour but de tester les possibilités de la méthodologie *SSM* sur un processus non-linéaire de nature différente que celle des perceptrons multicouches. On a choisi pour cela une série construite à partir de d'un modèle *RBF* (définition 2.9).

Le modèle est de la forme :

$$\begin{cases} X_t &= f^3_{\theta^*}(X^{(2)}_{t-1}) + \varepsilon_t \\ X_t &= \theta^*_1 e^{-(\theta^*_2 X_{t-1}+\theta^*_3)^2} + \theta^*_4 e^{-(\theta^*_5 X_{t-2}+\theta^*_6)^2} + \theta^*_7 + \varepsilon_t \\ \varepsilon_t &\sim \mathcal{N}(0,1/4) \end{cases} \tag{6.15}$$

$$\text{avec} \begin{cases} \theta^*_1 = & 1 & \theta^*_2 = & 4,7 \\ \theta^*_3 = & -1.3 & \theta^*_4 = & 1 \\ \theta^*_5 = & 0.9 & \theta^*_6 = & -0.4 \\ \theta^*_7 = & 1 & & \end{cases}$$

Le modèle choisi comporte 5 neurones cachés et 3 entrées correspondant aux trois retards de la série (soit 26 connexions). Ce modèle est donc fortement surparamétré par rapport au modèle simulé équation 6.15. Après avoir estimé les paramètres et appliqué l'algorithme *SSM*, le réseau final obtenu comporte 4 neurones cachés et toutes les connexions provenant de l'entrée correspond au retard d'ordre 3 ont été éliminées. Pour un nombre de paramètres égal à 10, la variance résiduelle associée au modèle neuronal final est égale à 0,251, soit une différence de 0,04% avec la variance de la perturbation du processus simulé.

Sur 100 expériences menées suivant le même protocole, mais avec des poids aléatoires initiaux différents, l'écart-type des variances résiduelles des différents modèles obtenus est égal à 0,012 et la moyenne est égale à 0,249.

6.4.4 Conclusion

Ces différents essais de modélisation à l'aide de l'algorithme *SSM* et de très nombreux autres effectués ont permis de *valider* les faits suivants :

- L'algorithme *SSM* élimine correctement les retards superflus, que ces retards soient proches dans le temps ou non.
- Si on injecte en entrée des variables complètement indépendantes de la variable étudiée, elles sont systématiquement éliminées.
- Si on essaye de modéliser un processus autorégressif linéaire, le réseau obtenu, dans la plupart des cas, ne conserve qu'un seul neurone dans sa couche cachée, sa fonction de transfert ne travaillant qu'au voisinage de zéro. Pour les autres cas, il arrive qu'une combinaison linéaire de fonctions sigmoïdes puisse approcher une forme linéaire.

Autres cas : D'autres tests du même type ont tous confirmé le bon comportement de l'algorithme. Il est cependant possible d'observer différents résultats qui dépendent de l'initialisation aléatoire des poids synaptiques. Il est en effet clair que la fonction de coût (l'erreur quadratique) que l'on minimise possède de nombreux minima locaux dans le cas d'un perceptron multicouches (voir chapitre 5). Néanmoins, si l'algorithme ne fournit pas l'architecture désirée, il converge vers une solution proche, en terme de variance résiduelle. L'algorithme remplit donc bien son rôle d'élimination des paramètres inutiles. Dans le cas de séries simulées artificiellement, il retrouve le plus souvent l'architecture idéale et les variables d'entrée significatives. À noter qu'une modélisation d'une série non-artificielle, la célèbre série $SUNSPOTS$ [1], a été effectué, à l'aide de l'algorithme SSM, dans l'article de (Cottrell et al., 1995) que l'on peut trouver en annexe A. La performance de cette modélisation est comparée à celle de modélisations linéaires et neuronales à base d'un perceptron multicouches, et confirme le bon comportement de l'algorithme.

1. Cette série représente le nombre de tâches solaires annuelles de l'année 1700 à nos jours. Plus de détails peuvent être trouvés section 5.3, et on peut voir son évolution figure 5.2.

Chapitre 7

Modèles neuronaux modulaires et prévision de séries stationnaires par morceaux

7.1 Introduction

La plupart des modélisations appliquées aux séries temporelles tentent de reproduire la structure et la dynamique globale des séries. Elles utilisent en particulier l'hypothèse sous-jacente que la dynamique associée à la série est de nature stationnaire. Cette hypothèse peut néanmoins s'avérer fausse pour un bon nombre de séries existantes dans la nature. Une importante sous-classe de séries non-stationnaires est constituée des séries stationnaires par morceaux, pour lesquelles la série entre à divers moments dans des régimes de différentes natures. En d'autres termes, la variable aléatoire associée à la série change de distribution pour différents intervalles de temps. La série des consommations électriques, par exemple, change de régime en fonction des saisons ou/et des caractéristiques calendaires (telles que les vacances). Une série financière peut aussi posséder une dynamique stationnaire par morceaux, liés à certaines périodes économiques telles que la récession, la reprise ou la stabilité (Granger, 1994). Dans ce cas, même si en théorie un modèle non-linéaire global, tel que celui basé sur un perceptron multicouches, est capable d'approcher n'importe quelle fonction continue sur un compact, y compris celles de la classe « changement de régimes », il s'avère être peu adapté à ce type de problème. En particulier, ne pas tenir compte des différences de niveaux de bruits associés à chaque régime entraîne une mauvaise répartition des capacités d'apprentissage du modèle. Le réseau apprenant avec la même précision les périodes bruitées et les périodes non-bruitées, il surapprend les périodes bruitées. Il peut aussi être intéressant d'utiliser un modèle qui trie explicitement les entrées qui lui servent à un moment donné. Ainsi, différents sous-modèles peuvent se spécialiser sur différentes entrées afin de minimiser le parasitage inévitable des entrées non-corrélées à la sortie. Ces problèmes deviennent particulièrement aigus quand on est confronté à un phénomène multivarié ou quand le nombre de variables exogènes à prendre en considération est important.

7.1.1 Modèle neuronal modulaire

On présente ici une classe de modèle appelé *modèle neuronal modulaire* (MNM), introduit dans la communauté des réseaux de neurones par (Jacobs et al., 1991) sous le nom de « *mixture of experts* ». L'idée de base, inhérent aux modèles modulaires, est de diviser un problème complexe en un certain nombre de sous-problèmes plus simples et plus spécifiques. De plus, puisque l'on ne connaît pas la partition à l'avance, la résolution des sous-problèmes et du partitionnement s'opèrent de manière simultanée. Dans l'approche que l'on a choisie, on utilise des réseaux du type perceptron multicouches pour émuler à la fois les sous-modèles (appelés *experts*) et le modèle de partitionnement (appelé *réseau contrôleur*). Il est important de souligner que la partition peut être non-linéaire, et que les sous-problèmes à résoudre peuvent l'être tout autant. Cette démarche est différente de celle de (Jordan et Jacobs, 1994) , qui considèrent que même avec des non-linéarités sous-jacentes, si la partition est assez fine, les sous-problèmes à résoudre sont de nature linéaire. On suppose ici, en termes plus probabilistes, que la partition ne se fait pas en fonction des non-linéarités existantes mais en fonction des différentes lois de probabilité qui régissent les données. Le modèle cherché peut alors être décrit complètement par un certain nombre de différents sous-modèles.

Pour chaque sous-modèle, on associe enfin un niveau de bruit. A travers certaines expériences, cette caractéristique apparaît être essentielle pour deux raisons :

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

7.1.2 Travaux précédents

L'approche modulaire est construite sur des bases statistiques solides. Elle peut être comparée à différents travaux précédents qui tentent d'effectuer un partitionnement à travers un modèle global. Ainsi (Elman, 1990) obtient des segmentations en évaluant la taille des erreurs, (Doutriaux et Zipser, 1990) les obtiennent en analysant les activations des différentes unités cachées, alors que (Levin, 1991) ajoute un jeu d'unités cachées pour coder les différents régimes. Pour tous ces essais, la tâche du modèle global est rude, puisqu'il lui faut apprendre à prévoir et à partitionner l'espace à travers un même ensemble de neurones cachés. L'idée de partitionner l'espace n'est pas nouvelle. L'un des pionniers dans le domaine de la prévision temporelle, H. Tong, a mis au point le modèle TAR (*Threshold autoregressif*) (Tong et Lim, 1980) (voir chapitre 2, définition 2.5) qui propose un partitionnement linéaire sans réelle justification théorique. Citons aussi les

« *mélanges de modèles* » introduit dans la communauté des économistes par Hamilton (Hamilton, 1990; Hamilton, 1994) (proches dans leur organisation des modèles neuronaux modulaires mais où le modèle de partitionnement et les sous-modèles de prévision sont linéaires).

On généralise donc ici, en s'inspirant des modèles linéaires et en proposant un modèle à la fois flexible (introduction de niveaux de bruits[1] et apprentissage simultané de la partition et des prévisions) et incluant des capacités de prise en compte des non-linéarités. Le modèle que nous adaptons pour effectuer de la prévision de séries temporelles a été introduit par (Jacobs et al., 1991) dans le cadre de la régression pure. (Jordan et Jacobs, 1994) ont, par la suite, développé une architecture adéquate relative au modèle théorique connu sous le nom de « hierarchical mixture of experts » (experts linéaires avec variance de bruit unique). Une étude de l'algorithme d'optimisation EM (« Expectation-Maximisation »), adapté à cette architecture, a été menée par (Jordan et Xu, 1995) alors que (Waterhouse et Robinson, 1995) l'appliquent à la prévision de la célèbre série $SUNSPOTS$ (Weigend et al., 1990; Nowlan et Hinton, 1992) et pour une régression non-linéaire sur un exemple de série avec bruit hétérogène (Weigend et Nix, 1994). D'autres travaux ont été menés par (Xu, 1994) qui applique cette modélisation à un mélange de deux processus $AR(2)$, et (Müller et al., 1994) qui utilisent une méthode appelée « hard competition » pour une tâche similaire. Citons enfin (Rabiner, 1989; Ivanova et al., 1994; Ivanova et al., 1994) qui utilisent des modèles du type « chaîne de Markov cachée » pour prévoir le régime dans lequel évolue la série, en se basant sur la succession des régimes précécents.

7.1.3 Organisation du chapitre

Dans la section 7.2, on donne les bases mathématiques menant à la construction de ce modèle et le cadre statistique qui le décrit et qui mène à la fonction de coût et à la construction de l'algorithme d'apprentissage. La section 7.3 est consacrée aux diverses expériences numériques et séries étudiées. Les trois que l'on étudie ici sont:

- une série simulée sur ordinateur,
- une série chaotique basée sur les variations d'intensité d'un laser,
- la série des consommations journalières d'électricité marquée par de fortes saisonnalités et de fortes corrélations avec différentes variables exogènes.

1. Dans le modèle neuronal modulaire, à chaque sous-modèle est associé un niveau de bruit différent. On peut noter la différence de cette démarche avec celle des modèles $ARCH$ de la définition 2.6.

7.2 Description du modèle neuronal modulaire

Cette section décrit les ingrédients nécessaires à la spécification et au fonctionnement d'un modèle neuronal modulaire. On détaille en particulier :

- la théorie
- l'architecture de base
- la fonction de coût (dérivée du maximum de vraisemblance)
- l'algorithme d'optimisation utilisé pour minimiser la fonction de coût.

7.2.1 Théorie du modèle modulaire

Le but du modèle est de favoriser la segmentation de l'espace des entrées. Cette méthodologie est à distinguer des modèles du type *additif* (Perrone, 1994) qui pondèrent les sous-modèles de manière statique (une même pondération pour toutes les entrées) alors que le modèle décrit ici le fait de manière dynamique (les pondérations changent en fonction des variables d'entrées). Ceci autorise les experts à se spécialiser au cours de l'apprentissage dans des régions spécifiques de l'espace des entrées alors qu'un modèle additif entraîne une implication de la totalité des sous-modèles sur la totalité de l'espace.

Dans cette section, on détaille la formalisation mathématique du modèle modulaire. De nature aléatoire, celle-ci est essentiellement basée sur la notion de distribution conditionnelle. La part la plus importante de la théorie, qui s'applique à tous les modèles de régression basés sur des modèles exponentiels, revient à (Jordan et Jacobs, 1994) ainsi d'ailleurs que l'adaptation de l'algorithme d'optimisation associé (appelé « Expectation-Maximisation »). Notre travail consiste donc essentiellement à adapter ce type de modélisation à l'autorégression et à la prévision de séries temporelles stationnaires par morceaux. On écrit dans la suite les équations et l'algorithme d'optimisation associé.

On considère un modèle entrée-sortie (d'entrée $\mathbf{x}$ et de sortie Y) composé d'un contrôleur et de K sous-modèles appelés experts (voir figure 7.1).

La sortie Y est une variable aléatoire à valeurs[2] dans $\mathbb{R}$, conditionnée par une variable discrète I à valeurs dans $\{1, 2, \dots, K\}$. La loi de la variable I dépend de l'entrée $\mathbf{x} \in \mathbb{R}^d, d \geq 1$ et est donnée par les probabilités $P_{\mathbf{x}}(I = j),\ j = 1, 2, \dots, K$. Le contrôleur calcule en sortie ces K probabilités.

La sortie Y s'écrit alors de manière additive suivant la formule générale suivante :

$$Y = f_j(\mathbf{x}) + \eta_j, \quad \text{si } I = j \tag{7.1}$$

2. Les valeurs de Y sont des scalaires, la généralisation au cas vectoriel est sans complication.

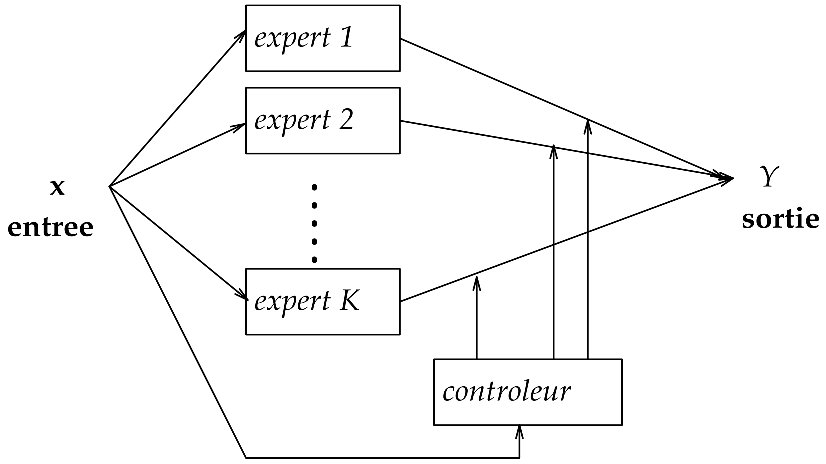


FIG. 7.1 – *Schéma d'modèle modulaire comprenant K experts.*

où f_j, fonction de $\mathbb{R}^d$ dans $\mathbb{R}$, constitue le $j^{ème}$expert et où η_j est une v.a. centrée. $f_j(\mathbf{x})$ est alors l'espérance de Y si $I = j$.

La loi de Y (pour une certaine entrée $\mathbf{x}$) peut donc s'écrire :

$$P_{\mathbf{x}}(Y = y) = \sum_{j=1}^{K} P_{\mathbf{x}}(I = j) P_{\mathbf{x}}(Y = y \mid I = j) \tag{7.2}$$

et l'espérance de Y :

$$E_{\mathbf{x}}(Y = y) = \sum_{j=1}^{K} P_{\mathbf{x}}(I = j) f_j(\mathbf{x}) \quad . \tag{7.3}$$

Le but est de construire un modèle paramétrique permettant de modéliser le module contrôleur et les K modules experts[3].

On modélise ici le contrôleur par une fonction paramétrique, de paramètre θ_g. Cette fonction réalise une approximation de la fonction : $\mathbf{x} \longrightarrow (P_{\mathbf{x}}(I = j))_{j=1,2,\ldots,K}$.
Soit $(g_j(\mathbf{x}, \theta_g))_{j=1,2,\ldots,K}$ le vecteur sortie de cette fonction.

De même, chaque module (ou expert) j est modélisé par une fonction paramétrique de vecteur de paramètre θ_j, pour tout $j = 1, 2, \ldots, K$. La sortie (scalaire) du $j^{ème}$module est notée $f_j(\mathbf{x}, \theta_j)$ (qui est une approximation de $f_j(\mathbf{x})$ définie en 7.1). Le modèle modulaire s'écrit alors :

$$Y = f_j(\mathbf{x}, \theta_j) + \varepsilon_j, \text{ si } I = j \tag{7.4}$$

3. On utilise dans la section suivante des réseaux de neurones du type *perceptron multicouches avec une couche cachée* pour modéliser les différents modules (aussi bien le contrôleur que les experts). Néanmoins, dans toute cette section, ces modules peuvent être approchés par n'importe quelle famille de fonctions paramétriques, possédant la propriété d'approximation universelle.

où $f_j(\mathbf{x}, \theta_j)$, est la sortie de $j^{ème}$expert et ε_j est un bruit centré et de variance σ_j^2.

On considère uniquement dans la suite le cas où les bruits $(\varepsilon_j)_{j=1,2,\ldots,K}$ sont gaussiens ($\varepsilon_j \sim \mathcal{N}(0, \sigma_j^2)$).

Le modèle est alors entièrement spécifié, et on peut estimer les paramètres du modèle donnés par $\Theta = \{\theta_g, \theta_1, \theta_2, \ldots, \theta_K, \sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2\}$, en utilisant la méthode classique du maximum de vraisemblance.

Afin d'alléger l'écriture, on utilise par la suite la notation $P(y \mid .)$ pour $P(Y = y \mid .)$. La probabilité d'obtenir $Y = y$, connaissant $\mathbf{x}$ et sachant que $I = j$ (Y a alors pour loi celle associée au $j^{ème}$expert) s'écrit alors :

$$P_{\mathbf{x}}(y \mid I = j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(y - f_j(\mathbf{x}, \theta_j))^2}{2\,\sigma_j^2}\right) \tag{7.5}$$

et la loi globale de Y dans le modèle modulaire (voir équation 7.2 et fig.7.3) :

$$P_{\mathbf{x}}(y) = \sum_{j=1}^{K} g_j(\mathbf{x}, \theta_g)\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(y - f_j(\mathbf{x}, \theta_j))^2}{2\,\sigma_j^2}\right) \quad . \tag{7.6}$$

La prévision de y est alors l'espérance de la sortie connaissant l'entrée $\mathbf{x}$. On la note :

$$\begin{aligned} \widehat{y}(\mathbf{x}) &= E_{\mathbf{x}}[Y] \\ &= \sum_{j=1}^{K} g_j(\mathbf{x}, \theta_g) f_j(\mathbf{x}, \theta_j) \end{aligned} \tag{7.7}$$

Supposons que l'on dispose d'un échantillon de N couples $(\mathbf{x}^{(t)}, y^{(t)})_{t=1,2,\ldots,N}$ auquel on associe N variables aléatoires $(I^{(t)})_{t=1,2,\ldots,N}$ à valeurs dans $\{1, 2, \ldots, K\}$. On suppose aussi que $I^{(t)}$ a une loi qui dépend de $\mathbf{x}^{(t)}$.

On obtient alors, par calcul classique, la vraisemblance (on pose $\mathcal{Y} = \{(y^{(t)})_{t=1,2,\ldots,N}\}$, $\mathcal{X} = \{(\mathbf{x}^{(t)})_{t=1,2,\ldots,N}\}$ et par commodité, $\mathcal{L}_{\mathcal{X}}(\mathcal{Y}, \Theta)$ est noté $\mathcal{L}(\mathcal{Y}, \Theta, \mathcal{X})$) :

$$\begin{aligned} \mathcal{L}(\mathcal{Y}, \Theta, \mathcal{X}) &= \prod_{t=1}^{N} P_{\mathbf{x}^{(t)}}(y^{(t)}) \\ &= \prod_{t=1}^{N} \sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g)\; P_{\mathbf{x}^{(t)}}(y^{(t)} \mid I^{(t)} = j) \end{aligned}$$

$$= \prod_{t=1}^{N} \sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g) \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(y^{(t)} - f_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\,\sigma_j^2}\right) \qquad (7.8)$$

Maximiser cette vraisemblance, revient alors à minimiser la fonction de coût suivante, qui est l'opposée du log-vraisemblance :

$$\begin{aligned} \mathcal{C}(\mathcal{Y}, \Theta, \mathcal{X}) &= -\ell n\, \mathcal{L}(\mathcal{Y}, \Theta, \mathcal{X}) \\ &= \sum_{t=1}^{N} -\ell n \left[\sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g) \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(y^{(t)} - f_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\,\sigma_j^2}\right)\right] \quad . \end{aligned}$$

La section suivante indique comment on peut construire, de manière pratique, une structure neuronale permettant de répondre au problème de la modélisation modulaire.

7.2.2 Architecture

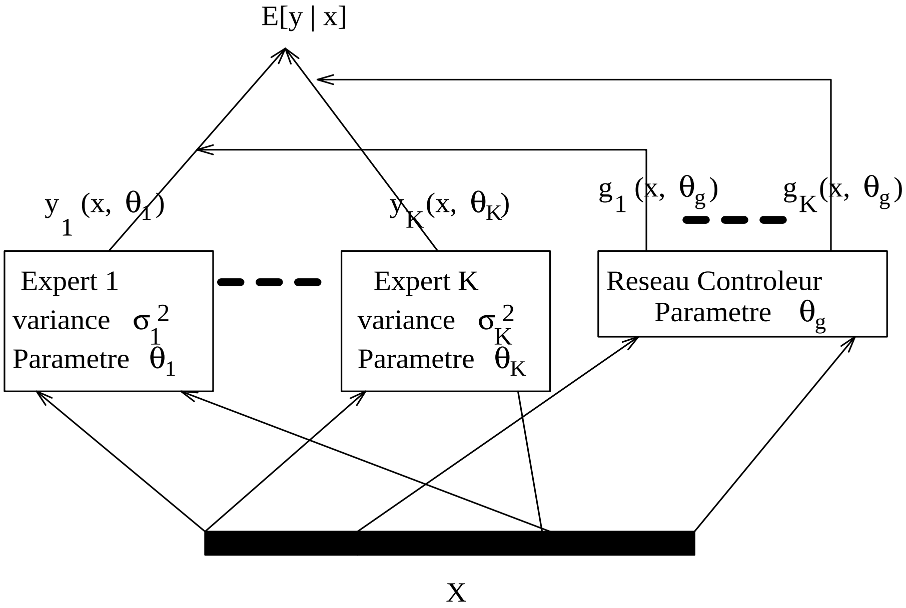


FIG. 7.2 – *Architecture du MNM. Les entrées* $\mathbf{x}$ *sont en bas de la figure. Les experts ainsi que le contrôleur sont des réseaux de neurones à couches. Les* K *sorties du contrôleur* $(g_j(\mathbf{x}, \theta_g))_{1 \leq j \leq K}$ *pondèrent les sorties des experts ; la sortie globale étant* $\sum_{j=1}^{K} g_j(\mathbf{x}, \theta_g) f_j(\mathbf{x}, \theta_j)$. *On peut fournir le même vecteur d'entrée aux experts et au contrôleur, ou distribuer l'information de manière non-exhaustive suivant les connaissances a priori du modèle.*

La figure 7.2 montre un schéma de l'architecture d'un modèle neuronal modulaire. Le modèle est constitué ici de K réseaux experts et d'un réseau contrôleur. Par la suite,

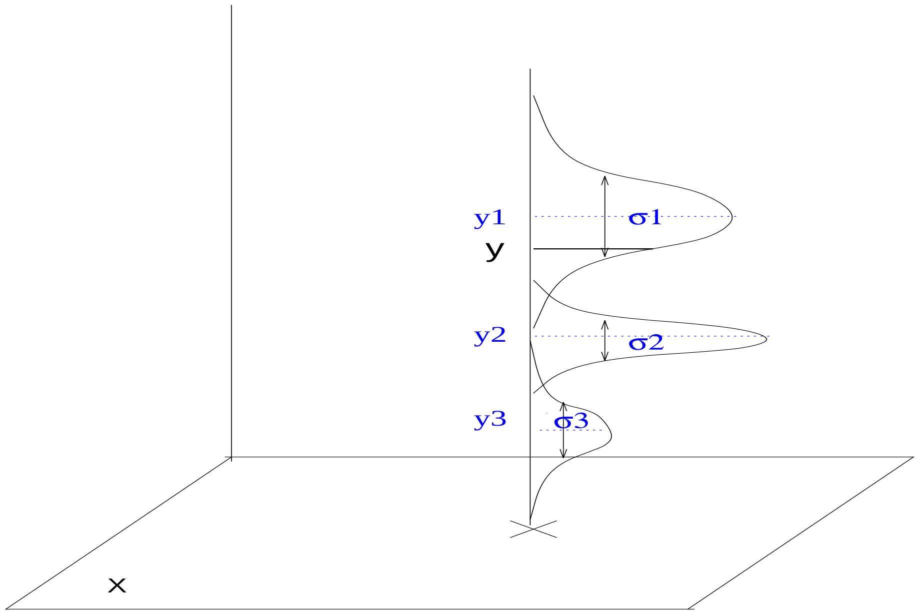


FIG. 7.3 – *Densité de probabilité associée à la distribution des mélanges de gaussiennes (équation 7.6). Les sorties de chaque expert f_j correspondent aux centres de chaque gaussienne et varient en fonction des entrées. Les variances des gaussienne (définies par σ_j) sont indépendantes des entrées et constantes après convergence du modèle (en phase application). Les centres des trois gaussiennes représentées ici sont pondérés par les sorties du contrôleur, représentant les probabilités que l'expert et sa gaussienne associée calque le « vrai » modèle.*

par souci de concision, nous n'emploierons plus le terme réseau et nous nommerons ces modules simplement « experts » et « contrôleur ». Dans le modèle que nous décrivons plus précisément ici, les experts, ainsi que le contrôleur, sont représentés par des réseaux de neurones avec une couche cachée. Ces réseaux sont tous munis de fonctions de transfert sigmoïdes associées aux neurones de la couche cachée. En ce qui concerne la couche de sortie, les experts ont un neurone de sortie muni d'une fonction de transfert linéaire, alors que le contrôleur possède K sorties munies de fonctions « softmax » que nous définirons par la suite.

Experts et contrôleur ont accès aux variables d'entrées. Ces entrées $\mathbf{x}$ sont le plus souvent communes aux experts et au contrôleur, mais peuvent aussi être distribuées de manière non-exhaustive suivant la connaissance a priori des variables. Ainsi, certaines peuvent contenir des informations utiles au partitionnement mais inutiles pour la prévision au sein de chaque partition.

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

Précisons le fonctionnement du contrôleur :

le contrôleur possède un neurone de sortie pour chaque expert. La $j^{ème}$sortie du contrôleur représente l'estimation de la probabilité que le $j^{ème}$expert soit activé sachant l'entrée. Pour cela et puisque ces probabilités sont de somme 1 (voir équation 7.2), on utilise en dernier traitement du réseau une fonction « exponentielle normalisée » aussi appelée « softmax » :

Soit $(s_j)_{j=1,2,\ldots,K}$ le vecteur formé par les sorties classiques d'un perceptron multicouches, avec une couche cachée munie de fonctions de transfert sigmoïdes et d'une couche de sortie linéaire. Le vecteur $(s_j)_{j=1,2,\ldots,K}$ est alors transformé par la fonction « *softmax* », afin de fournir des valeurs positives, de somme égale à 1,

$$g_j(\mathbf{x}^{(t)}, \theta_g) = \frac{e^{s_j}}{\sum_{k=1}^{K} e^{s_k}}, \quad \forall j = 1, 2, \ldots, K \quad . \tag{7.9}$$

La sortie globale du modèle (l'estimation de la sortie désirée fournie par le réseau) devient ainsi (conformément à l'éq. 7.7) une combinaison linéaire des sorties des experts pondérées par les sorties du contrôleur :

$$
\begin{aligned}
\widehat{y}(\mathbf{x}) &= \sum_{j=1}^{K} g_j(\mathbf{x},\theta_g) f_j(\mathbf{x},\theta_j) \\
&= \sum_{j=1}^{K} \frac{e^{s_j}}{\sum_{k=1}^{K} e^{s_k}} f_j(\mathbf{x},\theta_j)
\end{aligned}
$$

Notons que ce type de modèle n'appartient pas à la classe des réseaux de neurones à propagation classique, puisque la sortie du modèle modulaire est une somme de *produits* de sorties de réseaux à couches[4].

Ayant décrit l'ensemble des composants du modèle, il reste à déterminer les mécanismes d'estimations des paramètres $\Theta := \{\theta_g, \theta_1, \theta_2, \ldots, \theta_K, \sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2\}$.

7.2.3 L'algorithme d'optimisation: « Expectation-Maximization »

La fonction de coût décrite dans l'équation 7.9 est composée d'une somme de logarithmes népériens appliqués à des combinaisons linéaires d'exponentielles. Cette fonction s'avère difficile à minimiser. On peut cependant aborder le problème différemment, en introduisant des variables dites cachées qui représentent les segmentations inconnues a priori. Cette méthodologie, connue sous le nom d'algorithme « Expectation-Maximisation » (**EM**) permet alors de simplifier la fonction de coût et aide à déterminer les segmentations sous-jacentes. Supposons, de même que pour la section 7.2.1, que l'on dispose d'un échantillon de N couples $(\mathbf{x}^{(t)}, y^{(t)})_{t=1,2,\ldots,N}$ auquel on associe N variables aléatoires $(I^{(t)})_{t=1,2,\ldots,N}$ à valeurs dans $\{1, 2, \ldots, K\}$. On suppose aussi que $I^{(t)}$ a une loi qui dépend de $\mathbf{x}^{(t)}$. On construit alors K variables cachées binaires, de somme égale à 1, qui caractérisent la valeur de la variable aléatoire $I^{(t)}$. Ces variables ont pour loi :

$$
J_j^{(t)} = \begin{cases} 1 & \text{si } I^{(t)} = j \\ 0 & \text{sinon.} \end{cases} \tag{7.10}
$$

Ainsi, conformément à l'équation 7.1, $J_j^{(t)} = 1$ (équivalent à $I^{(t)} = j$) implique que la sortie Y a pour loi celle associée au $j^{ème}$expert.

Ceci permet de reformuler la vraisemblance, en transformant la somme sur les experts en produit (nous verrons par la suite comment). Evidemment, on ne connaît pas a priori la valeur de ces variables binaires caractérisant la segmentation. Le prix à payer pour cette simplification de la fonction de vraisemblance se manifeste alors par une étape d'estimation de ces variables (en fixant les paramètres) qui se passe en amont de

4. La classe de fonctions implémentées par des réseaux de neurones à propagation directe munie de l'opérateur addition est stable, alors qu'elle ne l'est plus munie de l'opérateur multiplication.

chaque pas d'optimisation (estimation des paramètres en considérant connues ces variables cachées). C'est ici qu'apparaît l'algorithme **EM** (« Expectation-Maximisation ») avec ses deux étapes caractéristiques pour chaque itération (calcul de d'espérance - maximisation). Plus formellement, on peut écrire la distribution nouvellement définie du couple {variables connues, variables cachées} par la fonction de vraisemblance suivante.

Avec $\mathcal{Y}_{\text{cachée}} = \{(J_j^{(t)})_{1\leq j\leq K, 1\leq t\leq N}\}$:

$$\mathcal{L}_2(\mathcal{Y}, \mathcal{Y}_{\text{cachée}}, \Theta, \mathcal{X}) = \prod_{t=1}^{N}\prod_{j=1}^{K}\left[g_j(\mathbf{x}^{(t)}, \theta_g)\ P_{\mathbf{x}^{(t)}}(y^{(t)} \mid I^{(t)} = j)\right]^{J_j^{(t)}} \tag{7.11}$$

A noter que la distribution des variables cachées a été choisie de manière à s'accorder avec la distribution des données « complètes » $(\mathcal{Y}, \mathcal{Y}_{\text{cachée}})$. Ainsi si on intègre la densité de probabilité des données « complètes », on trouve la densité marginale décrite par l'équation 7.8 :

$$\begin{aligned}
\int P_{\mathcal{X}}(\mathcal{Y}, \mathcal{Y}_{\text{cachée}}) d\mathcal{Y}_{\text{cachée}} &= \int \prod_{t=1}^{N}\prod_{j=1}^{K}\left[g_j(\mathbf{x}^{(t)}, \theta_g)\ P_{\mathbf{x}^{(t)}}(y^{(t)} \mid I^{(t)} = j)\right]^{J_j^{(t)}} d(J_k^{(t)})_{k=1,2,\ldots,K} \\
&= \sum_{k=1}^{K}\prod_{t=1}^{N}\left[g_k(\mathbf{x}^{(t)}, \theta_g)\ P_{\mathbf{x}^{(t)}}(y^{(t)} \mid I^{(t)} = k)\right] \\
&= P_{\mathcal{X}}(\mathcal{Y}) \quad .
\end{aligned}$$

Comme souligné précédemment, on ne peut pas utiliser la nouvelle fonction de coût (équation 7.11) puisque qu'on ne connaît pas les valeurs des variables cachées. L'idée de l'algorithme **EM** est alors de remplacer les variables cachées $J^{(t)}$ par leurs espérances h_j. Ces estimations sont calculées lors de l'étape **E** :

$$\begin{aligned}
h_j(\mathbf{x}^{(t)}, y^{(t)}, \theta_g, \theta_1, \theta_2, \ldots, \theta_K) &= E_{\mathbf{x}^{(t)}}\left[J_j^{(t)} \mid y^{(t)}\right] = P_{\mathbf{x}^{(t)}}(J_j^{(t)} = 1 \mid y^{(t)}) \\
&= P_{\mathbf{x}^{(t)}}(I^{(t)} = j \mid y^{(t)}) \\
&= \frac{P_{\mathbf{x}^{(t)}}(y^{(t)}, I^{(t)} = j)}{P_{\mathbf{x}^{(t)}}(y^{(t)})} = \frac{P_{\mathbf{x}^{(t)}}(I^{(t)} = j)\ P_{\mathbf{x}^{(t)}}(y^{(t)} \mid I^{(t)} = j)}{P_{\mathbf{x}^{(t)}}(y^{(t)})} \\
&= \frac{g_j(\mathbf{x}^{(t)}, \theta_g)\ P_{\mathbf{x}^{(t)}}(y^{(t)} \mid I^{(t)} = j)}{\sum_{k=1}^{K} g_k(\mathbf{x}^{(t)}, \theta_g)\ P_{\mathbf{x}^{(t)}}(y^{(t)} \mid I^{(t)} = k)}
\end{aligned}$$

Par la suite, cette notation étant lourde, on utilise simplement le terme $h_j^{(t)}$ pour $h_j(\mathbf{x}^{(t)}, y^{(t)}, \theta_g, \theta_1, \theta_2, \ldots, \theta_K)$. En associant une distribution gaussienne à chaque expert, on peut définir complètement $h_j^{(t)}$, en utilisant les expressions des vecteurs $\left(g_j(\mathbf{x}^{(t)}, \theta_g)\right)_{j=1,2,\ldots,K}$, $\left(f_j((\mathbf{x}^{(t)}, \theta_j)\right)_{j=1,2,\ldots,K}$ et $y^{(t)}$:

$$h_j^{(t)} = \frac{g_j(\mathbf{x}^{(t)},\theta_g)\ \dfrac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\dfrac{\left(y^{(t)} - f_j(\mathbf{x}^{(t)},\theta_j)\right)^2}{2\sigma_j^2}\right)}{\displaystyle\sum_{k=1}^{K} g_k(\mathbf{x}^{(t)},\theta_g)\ \dfrac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(\dfrac{\left(y^{(t)} - f_k(\mathbf{x}^{(t)},\theta_k)\right)^2}{2\sigma_k^2}\right)} \tag{7.12}$$

En prenant l'opposé du logarithme de l'éq. 7.11 et en remplaçant le vecteur $(J_j^{(t)})_{j=1,2,\ldots,K}$ par le vecteur $(h_j^{(t)})_{j=1,2,\ldots,K}$ (son espérance), on obtient la fonction de coût à optimiser ($\mathcal{L}_2$ est définie dans l'équation 7.11). On pose $\mathcal{H} = \{\left(h_j^{(t)}\right)_{1\leq j\leq K, 1\leq t\leq N}\}$

$$\begin{aligned}
\mathcal{C}_{\mathrm{EM}}(\mathcal{Y},\mathcal{H},\Theta \mid \mathcal{X}) &= -\ell n(\mathcal{L}_2) \qquad (7.13)\\
&= -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)}\ \ell n\left[g_j(\mathbf{x}^{(t)},\theta_g)\ P_{\mathbf{x}^{(t)}}(y^{(t)} \mid I^{(t)} = j)\right]\\
&= -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)}\ \ell n\left[g_j(\mathbf{x}^{(t)},\theta_g)\ \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(y^{(t)} - f_j(\mathbf{x}^{(t)},\theta_j)\right)^2}{2\sigma_j^2}\right)\right]\\
&= -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)}\left[\ell n\left(g_j(\mathbf{x}^{(t)},\theta_g\right) - \frac{\left(y^{(t)} - f_j(\mathbf{x}^{(t)},\theta_j)\right)^2}{2\sigma_j^2} - \frac{1}{2}\,\ell n\, 2\pi\sigma_j^2\right]
\end{aligned}$$

L'algorithme complet s'obtient en passant d'une étape à une autre, la succession de l'étape **E** à **M** constituant une itération.

Dans l'étape **M**, on ajuste le vecteur des paramètres du modèle, que l'on note ici $\Theta = \{\theta_g,\theta_1,\theta_2,\ldots,\theta_K,\sigma_1^2,\sigma_2^2,\ldots,\sigma_K^2\}$, pour minimiser cette fonction de coût.

Les variances associées aux experts s'ajustent de manière directe[5]. La variance du $j^{ème}$expert représente en fait la somme des erreurs quadratiques des experts, pondérées par les probabilités a posteriori que ces experts génèrent la sortie.

$$\sigma_j^2 := \frac{\displaystyle\sum_{t=1}^{N} h_j^{(t)}\left(y^{(t)} - f_j(\mathbf{x}^{(t)},\theta_j)\right)^2}{\displaystyle\sum_{t=1}^{N} h_j^{(t)}} \quad . \tag{7.14}$$

Soulignons qu'il est nécessaire pour certains problèmes d'introduire une borne inférieure pour les variances, écartant le risque d'obtenir des valeurs très petites et de

5. Du fait de la place particulière de ces variances dans la fonction de coût, on peut résoudre explicitement l'équation $\partial\mathcal{C}_{\mathrm{EM}}/\partial\sigma_j = 0$ et calculer les meilleures estimations sans utiliser de méthode itérative.

voir « exploser » la fonction de coût et surtout les gradients. Cette limite imposée aux variances, correspond en fait à une distribution a priori qui bloquerait la variable au-dessus d'une certaine valeur. Cette distribution a priori peut aussi être redéfinie en utilisant des fonctions moins brutales dans leurs transitions.

Puisqu'on utilise des réseaux de neurones munis de fonctions de transfert non-linéaires, les poids ne peuvent être ajustés directement. En particulier, on ne peut résoudre explicitement le système d'équations $\partial\mathcal{C}_{\mathrm{EM}}/\partial\theta_j = 0,\ \forall j = 1, 2, \ldots, K$. On utilise donc des méthodes d'optimisation itératives du premier ou du deuxième ordre, basées sur le gradient (voir chapitre 5). Un des problèmes que l'on se pose, est donc le calcul du gradient à l'instant (t) en fonction des paramètres. En ce qui concerne le $j^{ème}$expert, on a :

$$\frac{\partial\mathcal{C}^{(t)}_{\mathrm{EM}}}{\partial\theta_j} = \frac{\partial\mathcal{C}^{(t)}_{\mathrm{EM}}}{\partial f_j(\mathbf{x}^{(t)}, \theta_j)} \times \frac{\partial f_j(\mathbf{x}^{(t)}, \theta_j)}{\partial\theta_j} \quad . \tag{7.15}$$

Le calcul de $\partial\mathcal{C}^{(t)}_{\mathrm{EM}}/\partial f_j(\mathbf{x}^{(t)}, \theta_j)$ mène à une valeur proportionnelle à la différence entre la sortie désirée et la sortie de l'expert, pondérée par les h_j et divisée par les variances associées :

$$\frac{\partial\mathcal{C}^{(t)}_{\mathrm{EM}}}{\partial f_j(\mathbf{x}^{(t)}, \theta_j)} = -h^{(t)}_j \frac{1}{\sigma^2_j} \left(y^{(t)} - f_j(\mathbf{x}^{(t)}, \theta_j) \right) \quad . \tag{7.16}$$

Le calcul de $\partial f_j(\mathbf{x}^{(t)}, \theta_j)/\partial\theta_j$ se fait de manière classique par rétropropagation [6].

On peut noter dans l'équation 7.16 les deux termes supplémentaires qui viennent s'ajouter au gradient classique.

En ce qui concerne le réseau superviseur, on peut exprimer aisément la formule du gradient de la fonction de coût en fonction des sorties du réseau, avant l'application de la fonction softmax (voir équation 7.9). On a alors :

$$\frac{\partial\mathcal{C}^{(t)}_{\mathrm{EM}}}{\partial\theta_g} = \frac{\partial\mathcal{C}^{(t)}_{\mathrm{EM}}}{\partial s_j} \times \frac{\partial s_j}{\partial\theta_g} \tag{7.17}$$

avec,

$$\frac{\partial\mathcal{C}^{(t)}_{\mathrm{EM}}}{\partial s_j} = -\left(h^{(t)}_j - g_j(\mathbf{x}^{(t)}, \theta_g) \right) \tag{7.18}$$

Le vecteur des paramètres (θ_g au sein de la fonction $\partial\mathcal{C}^{(t)}_{\mathrm{EM}}/\partial s_j$) est ajusté afin d'uniformiser les variables $P_{\mathbf{x}^{(t)}}(I^{(t)} = j) = g_j(\mathbf{x}^{(t)}, \theta_g)$ et $P_{\mathbf{x}^{(t)}}(I^{(t)} = j \mid y^{(t)}) = h^{(t)}_j$. A noter la différence entre les h_j et les g_j. Le terme h_j est la *probabilité a posteriori* d'utiliser

6. Cf. chapitre 5 section 5.1.2.

le $j^{ème}$expert, et son calcul est basé sur l'information issue de l'entrée et de la sortie. Le terme g_j est la *probabilité a priori*, et son calcul s'effectue durant l'apprentissage, sans connaissance a priori de la sortie (g_j n'est basé que sur l'entrée) et peut ainsi être utiliser en généralisation [7]. De même que pour le gradient relatif aux poids des experts, le calcul de $\partial f_j(\mathbf{x}^{(t)}, \theta_j)/\partial\theta_j$ se fera de manière classique par rétropropagation [8].

Dans toutes nos expérimentations, le réseau contrôleur et les réseaux experts sont des perceptrons multicouches avec une couche cachée et on utilise une méthode d'optimisation du second ordre pour optimiser la fonction de coût de l'étape **M** (la méthode *BFGS* (Press et al., 1992)).

7.2.4 Comparaison avec les autres fonctions de coût

Nous terminons cette section en explicitant la fonction de coût définie précédemment et en rappelant les définitions des fonctions de coût classiques pour les réseaux de neurones (par souci de concision, on n'écrit plus la somme sur l'ensemble des exemples de la base d'apprentissage et on écrit simplement l'un des termes de cette somme). La plus standard, associée à des modèles non-modulaires, est l'erreur quadratique (EQ) ($\widehat{y}(\mathbf{x})$ représente la sortie du réseau) :

$$\mathcal{C}_{\text{EQ}} = \frac{1}{2}\left(y - \widehat{y}(\mathbf{x})\right)^2 \quad . \tag{7.19}$$

Cette fonction de coût, très simple, provient du calcul du maximum de vraisemblance (en éliminant les constantes) dans le cas d'un processus fortement stationnaire, ergodique et associé à un bruit gaussien [9].

Dans le cas où la variance dépend des entrées (cas *hétéroscédastique*), on peut utiliser un modèle développé dans (Weigend et Nix, 1994) et comportant deux sorties, l'une pour l'estimation de la *moyenne* (en fait la prédiction de la valeur désirée) et l'autre pour l'estimation de la *variance* de cette prédiction :

$$\mathcal{C}_{\text{BEL}} = \frac{1}{2}\left[\frac{\left(y - \widehat{y}(\mathbf{x})\right)^2}{\sigma^2(\mathbf{x})} + \ell n\, \sigma^2(\mathbf{x})\right] \quad . \tag{7.20}$$

où BEL signifie « barres d'erreurs locales » (en fait le modèle comporte 2 sorties, l'une pour la prévision, l'autre pour sa variance). A noter que lors de la minimisation de cette fonction, un équilibre s'opère entre le terme contenant l'inverse de la variance et le logarithme de celle-ci.

7. On pourra donner un diagnostic de bon apprentissage de ces probabilités, en traçant le graphe des h_j versus g_j.

8. Voir section 5.1.2 pour plus de détails.

9. Voir la section 2.1.1 sur les modèles gaussiens.

Nous présentons maintenant des fonctions de coût plus adaptées à des séries ayant des distributions du genre mélange de gaussiennes. Celle issue de la théorie des réseaux de neurones modulaires :

$$\mathcal{C}_{\text{MNM}} = -\,\ell n \left[\sum_{j=1}^{K} g_j(\mathbf{x}, \theta_g)\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(y - f_j(\mathbf{x}, \theta_j))^2}{2\,\sigma_j^2} \right) \right] \quad . \tag{7.21}$$

et celle adaptée pour l'algorithme EM, après l'introduction de la probabilité a posteriori h_j utilisant aussi l'information de la valeur désirée (voir section 7.2.3):

$$\mathcal{C}_{\text{EM}} = \sum_j -h_j^{(t)}\, \ell n\, g_j(\mathbf{x}, \theta_g) + \frac{1}{2} \sum_j h_j^{(t)} \left[\frac{(y - f_j(\mathbf{x}, \theta_j))^2}{\sigma_j^2} + \ell n\, \sigma_j^2 \right] \quad . \tag{7.22}$$

Le premier terme peut être assimilé à un terme d'entropie (les variables g_j et h_j ont des valeurs proches, sauf que la deuxième variable utilise plus d'information ; voir section 7.2.3. Cette entropie mesure en fait le désordre des experts. Il devient faible quand le désordre est faible (les sorties du superviseur sont alors binaires ou presque binaires) et atteint son maximum dans le cas d'un désordre total (quand $g_j = 1/K, 1 \leq j \leq K$, les experts se partagent de manière égale la tâche de prévision).

7.3 Expériences numériques

Afin d'évaluer les aptitudes des réseaux modulaires à effectuer de la prévision de séries temporelles, on applique cette technique à trois jeux de données distincts. Le premier jeu est une série engendrée artificiellement sur ordinateur, mélangeant processus déterministe et stochastique. Ces données ont l'avantage d'être adaptées parfaitement à la théorie sous-jacente aux réseaux modulaires. On est en effet en présence d'un mélange de différents processus associés à des bruits de différentes amplitudes. Cette série étant de plus créée artificiellement, on connaît à la fois les segmentations, les niveaux de bruit et la nature de ce mélange, permettant une évaluation efficace des sorties du modèle et de l'algorithme **EM**. Le deuxième jeu est issu d'un phénomène physique chaotique construit en laboratoire, caractérisant l'intensité d'un laser. Cette série est composée d'une phase de relative stabilité, d'une phase d'effondrement très perturbée et d'une phase intermédiaire de restabilisation. Cette série est soumise aux erreurs de relevés, mais aussi, puisque chaotique, à des perturbations particulièrement délicates à modéliser. Le troisième jeu enfin caractérise la consommation d'électricité journalière de la France métropolitaine. On dispose aussi de variables corrélées à la consommation électrique, telles que la température, la nébulosité ou les particularités calendaires. Ces données sont soumises à des erreurs de relevés mais aussi à de multiples comportements inhérents à la clientèle et aux aléas climatiques. On peut observer ainsi des données appartenant à certains intervalles qui sont faciles à modéliser quand la conjoncture est connue et que le nombre de données est suffisant, mais aussi d'autres plus « pathologiques » liées à des jours fériés ou des vacances.

Les principaux résultats qui se dégagent de ces études sont:

1. Les réseaux modulaires retrouvent correctement la partition de l'espace dans le cas de la série simulée sur ordinateur, et fournissent des solutions de partitionnement cohérentes avec les connaissances a priori dans le cas des autres séries.
2. Les performances au sens de la somme des erreurs quadratiques sont sensiblement meilleures qu'une modélisation classique pour la série artificielle et du même ordre pour les autres.
3. Grâce à l'introduction de différents niveaux de variance associés aux groupes de données, on observe une résistance au sur-apprentissage.
4. Du fait des partitions mises en évidences, des sorties du contrôleur et des variances citées plus haut, des analyses a posteriori fines sont possibles.

Les expériences décrites ci-dessous sont toutes des tentatives de prévision à un pas, et seront détaillées suivant le même schéma: description des données, description de l'architecture et de l'apprentissage, analyses.

7.3.1 Série simulée sur ordinateur

Afin de tester l'aptitude des modèles modulaires à séparer les différentes densités de distribution, on crée artificiellement une série issue d'une combinaison d'un processus déterministe et d'un processus aléatoire.

7.3.1.1 Description des données

Le premier des deux processus utilisés ici est le célèbre processus appelé en anglais « quadratic map » (voir section 2.1.1 pour plus de détails). Ce processus déterministe bien que généré par une équation de base très simple (éq. 7.23, processus (1)) évolue de manière chaotique pour peu que le point initial appartienne à un intervalle défini (ici]-1,1[). Le deuxième processus est une composition d'un autorégressif de degré 1 (avec l'ajout d'un bruit gaussien de variance 0.1) et d'une tangente hyperbolique (éq. 7.23, processus (2)). On choisit de passer d'un processus à l'autre avec un commutateur c qui passe de 0 à 1 ou de 1 à 0 avec une probabilité de 0.02 suivant une loi uniforme (i.e., le temps moyen de changement est environ de 50 pas).

$$x_t = \begin{cases} 2(1 - x_{t-1}^2) - 1 & \text{si } c = 0 \quad \text{(processus (1))} \\ \tanh(-1.2 x_{t-1} + \varepsilon_t) & \text{sinon} \quad \text{(processus (2))} \end{cases} \tag{7.23}$$

où ε_t est un bruit blanc gaussien de variance 0.1. On génère 2000 points, 1000 pour l'apprentissage, 1000 pour le test. A noter que les deux processus sont bornés dans l'intervalle]-1,1[, et que le bruit associé au second processus est additionné à l'AR(1) avant l'application de la fonction $x \mapsto \tanh(x)$ ce qui implique en sortie un bruit qui n'est plus réellement gaussien (en particulier, il est borné de variance finale 0.071). On considérera néanmoins par la suite ce bruit comme gaussien.

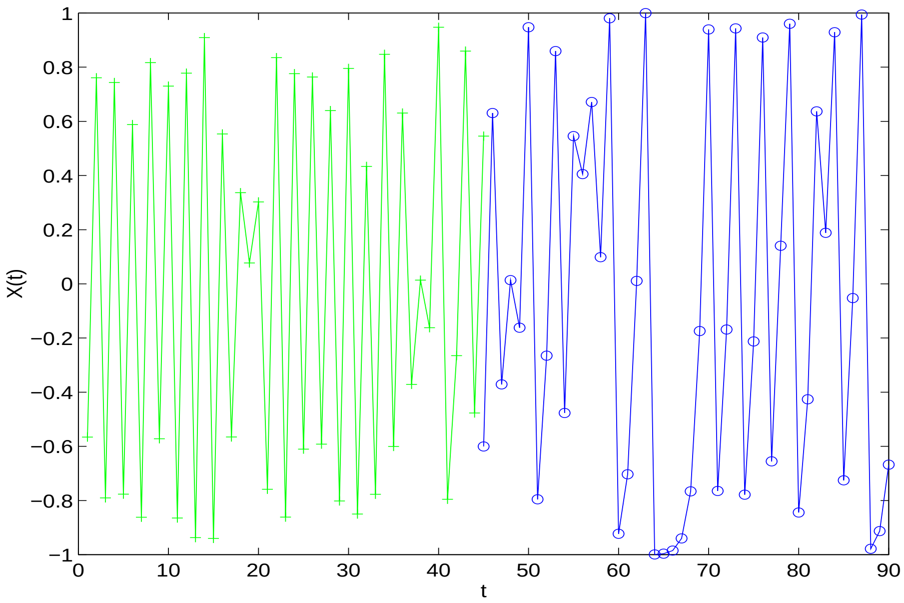


FIG. 7.4 – *Évolution de la série sur 90 pas de temps. La partie gauche représente la composition de l'AR(1) avec la* tanh, *la partie droite le processus quadratique.*

7.3.1.2 Architecture et apprentissage du modèle

L'architecture est constituée de quatre réseaux de neurones du type perceptron multicouche, 3 experts [10] et 1 contrôleur. Chaque expert possède 10 neurones cachés [11] munis de fonctions de transfert sigmoïdes. Le contrôleur en possède 20, et, puisque le problème abordé ici est du type autorégression, les neurones de sortie sont tous munis de fonction de transfert linéaire. Chacune des 50 simulations effectuées à partir de différents poids initiaux aboutit aux mêmes résultats :

– après convergence, un expert n'est plus utilisé (pour chaque exemple de la base, la sortie associée du réseau contrôleur est nulle (figure 7.7)) ;

10. Deux experts sont suffisants pour émuler ce mélange de deux distributions, mais nous nous situons dans un cadre où nous ne connaissons pas le nombre de distributions mélangées. On utilise donc un modèle muni d'un grand nombre d'experts.

11. Bien qu'un neurone caché soit suffisant pour émuler le processus bruité, on ne connaît pas a priori le nombre de neurones nécessaires. Dans la pratique, on utilise donc un nombre jugé suffisant pour la modélisation. Du reste 10 neurones cachés sont nécessaires pour modéliser le processus quadratique, et nous avons décidé d'utiliser le même nombre de neurones pour chaque expert.

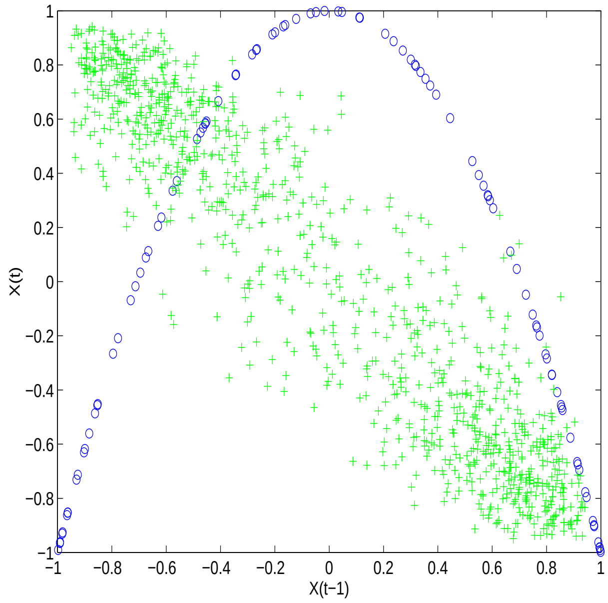


FIG. 7.5 – *Représentation bivariée du processus généré par ordinateur (x_t en fonction de x_{t-1}). Le processus quadratique est représenté par 'o' et la composition de l'AR(1) avec la* tanh *est noté '+'.*

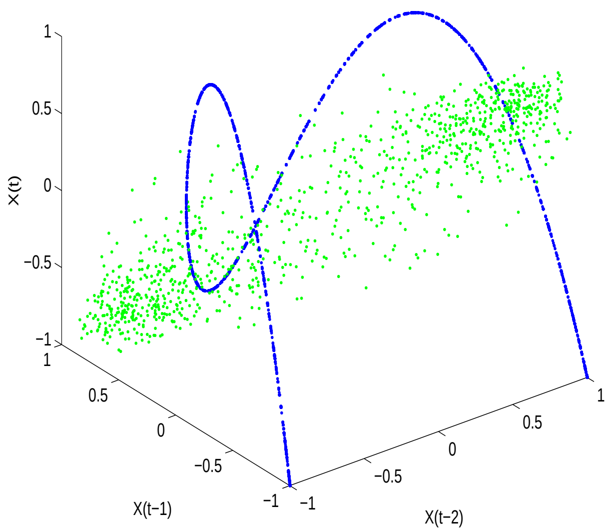


FIG. 7.6 – *Représentation graphique de x_t fonction de x_{t-1} et x_{t-2}. Comme pour la figure 7.5, on peut voir l'evidente différence de distribution des deux processus.*

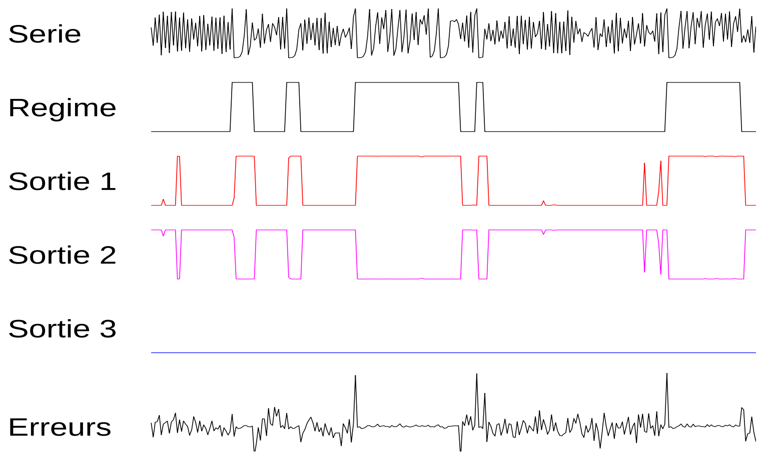


FIG. 7.7 – *Sorties du contrôleur. On compare sur cette figure les réels changements de régimes et ceux produits par le réseau. La somme des trois sorties est égale à 1, et on peut voir que la plupart du temps, les sorties sont binaires (la troisième sortie est toujours nulle). Cet ensemble de points provient de la base de test.*

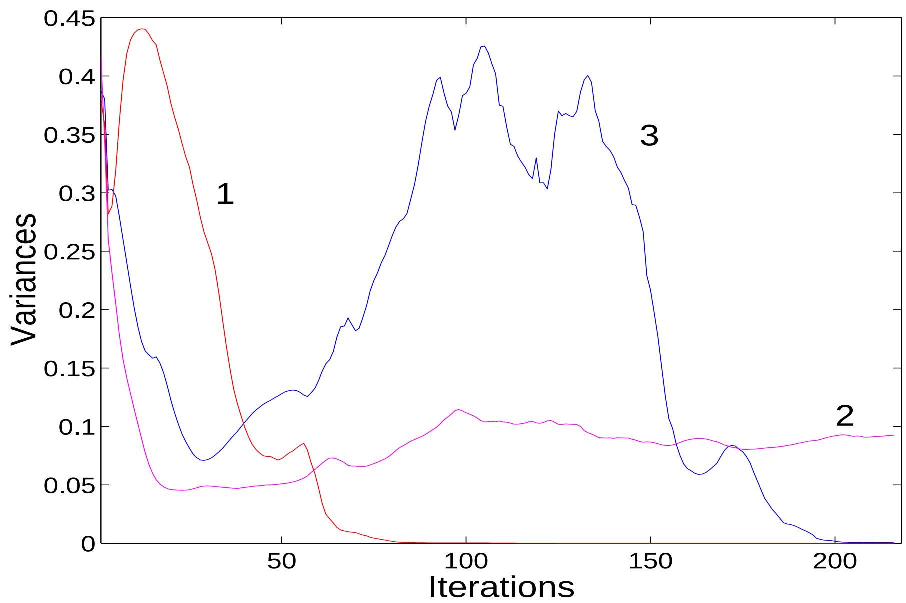


FIG. 7.8 – *Évolution des variances durant l'apprentissage. En fin de convergence, l'expert 1 est spécialisé dans la prévision du processus quadratique. Après 60 iterations, la variance associée devient pratiquement nulle. L'expert 2 apprend à gérer les prévisions du processus bruité, et la valeur de sa variance s'approche de 0.1. Le troisième expert est écarté après l'itération 200.*

– les partitions sont déterminées et apprises correctement (figure 7.7)

– les changements de régimes sont repérés après un ou deux pas (les points exacts de rupture des distributions sont déterminés aléatoirement et ne peuvent pas, par conséquent, être prédits).

– la valeur de la variance résiduelle associée à l'expert qui gère les prédictions du processus quadratique (figure 7.8) est petite (de l'ordre 0.001) en comparaison de celle associée à l'expert gérant les prévisions du processus bruité. Cette dernière est d'ailleurs très proche du niveau de bruit artificiel ajouté (équation 7.23).

– les performances du modèle modulaire sont du même ordre ou meilleures que celles des différents modèles neuronaux testés, basés sur des perceptrons multicouches classiques avec 10, 20 , 30, 40, 50 neurones cachés.

7.3.1.3 Analyse de la modélisation

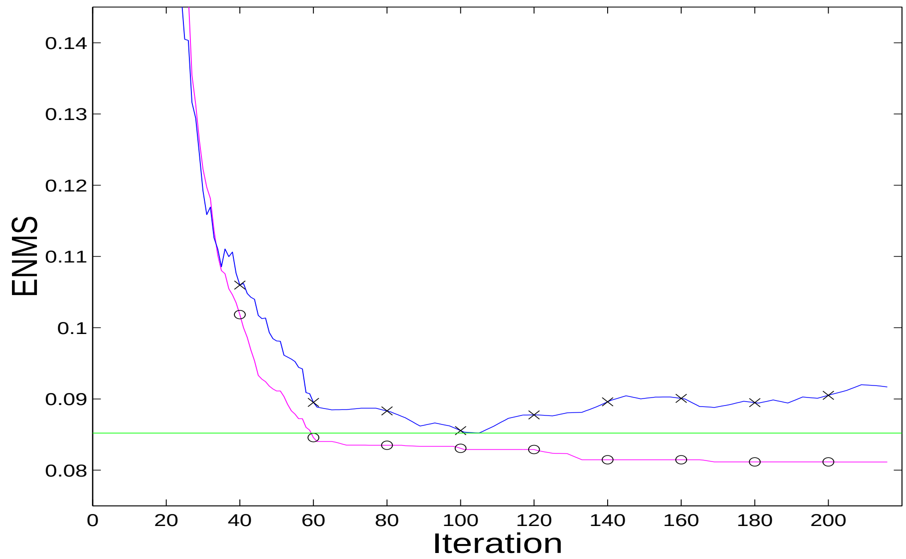


FIG. 7.9 – *Courbes de performances durant l'apprentissage du modèle neuronal modulaire. On note 'o' la courbe correspondant à la base d'apprentissage, '+' celle associée à la base de test. Le minimum sur la base de test est 0.085.*

De l'apprentissage (figure 7.9), résulte un très faible sur-apprentissage[12]. L'association d'une segmentation de l'espace, et des variances de différents niveaux associés à chaque segmentation, joue un rôle crucial dans ce phénomène.

12. Le sur-apprentissage entraîne une mauvaise généralisation. Ce phénomène est dû à une prise en considération de perturbations aléatoires indépendantes de la dynamique réelle du processus (indépendantes de la loi de probabilité qui régit le processus).

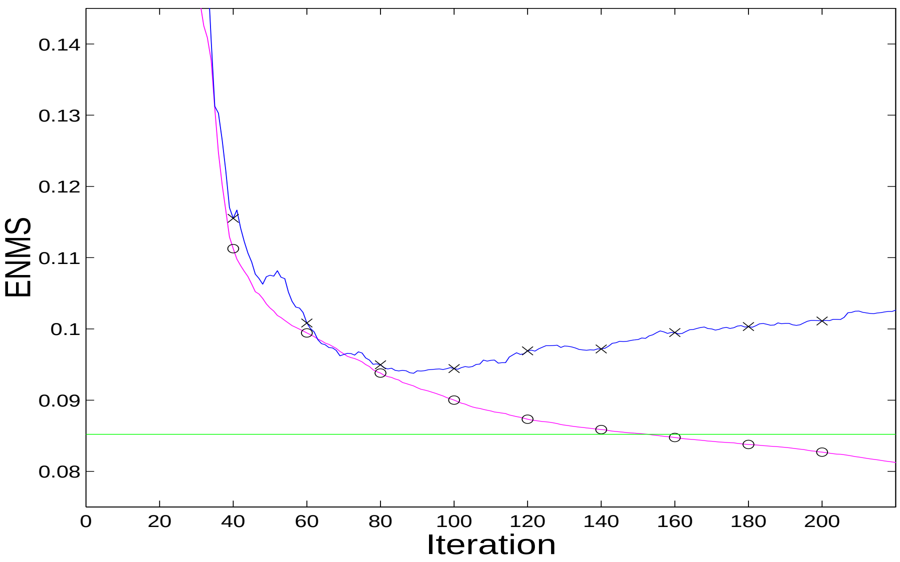


FIG. 7.10 – *Courbes de performances durant l'apprentissage d'un perceptron à une couche cachée munie de 50 neurones. On note 'o' la courbe correspondant à la base d'apprentissage, '+' celle associée à la base de test. Le minimum sur la base de test est 0.095.*

Supposons que la partition soit correctement apprise (et donc que les sorties du contrôleur soient binaires) la fonction de coût (équation 7.14) devient la somme des erreurs quadratiques divisées par les variances assignées à chaque distribution :

$$\begin{aligned} -2\,\ell n\,\mathcal{L} &= \frac{1}{\sigma_1^2}\sum_{t\in\Omega_1}(x_t-\hat{x}_t)^2+\frac{1}{\sigma_2^2}\sum_{t\in\Omega_2}(x_t-\hat{x}_t)^2+\ell n(\sigma_1^2)+\ell n(\sigma_2^2)+2\,\ell n(2\pi) \\ &= \frac{1}{\sigma_1^2}(SSE_1+\frac{\sigma_1^2}{\sigma_2^2}SSE_2)+\ell n(\sigma_1^2\sigma_2^2)+\text{const.} \end{aligned}$$

où Ω_1 est l'ensemble des exemples assignés au processus quadratique déterministe, Ω_2 est le complémentaire de Ω_1 (ensemble des exemples attribués au second processus), σ_1^2 et σ_2^2 leurs variances respectives et SSE la somme des carrés des erreurs. Dans ce cas, minimiser $-\,\ell n\,\mathcal{L}$ revient à minimiser dans un même temps SSE_1 et SSE_2 pondérés par un coefficient σ_1^2/σ_2^2 (petit puisque $\sigma_1^2 \ll \sigma_2^2$) et le logarithme de leurs variances respectives. Par conséquent, les erreurs ne sont pas prises en considération de manière similaire, et l'apprentissage s'opère plus sur les parties de la série jugées peu bruitées par le modèle que sur les parties jugées plus bruitées et difficiles à modéliser. L'apprentissage s'arrête lorsqu'un minimum est trouvé, basé sur *les erreurs quadratiques* des différents sous-modèles, sur leurs partitions respectives et sur *leurs variances*. D'un autre côté, tenter de modéliser la série en utilisant un modèle neuronal simple, type perceptron multicouches implique un sur-apprentissage évident (figure 7.10). En fait, essayer de minimiser les erreurs sans tenir compte des différences de niveaux de bruits, entraîne un gaspillage des ressources d'apprentissage (trop de paramètres vont être alloués à ajuster le modèle à la partie bruitée de la série) et de piètre performance en généralisation. A noter que ce phénomène de résistance au sur-apprentissage se retrouve pour des modèles modulaires fortement surparamétrés (tels que ceux dont les experts utilisent autant de paramètres que le perceptron cité plus haut).

7.3.2 Série de l'intensité d'un laser

La série présentée dans la suite est une série issue d'un phénomène physique lié aux fluctuations du flot de photons engendré par un laser. Cette série a été étudiée au sein d'un bon nombre d'articles [13], ce qui permet une analyse et des comparaisons entre les méthodes. Cette série présente un comportement chaotique relativement difficile à modéliser.

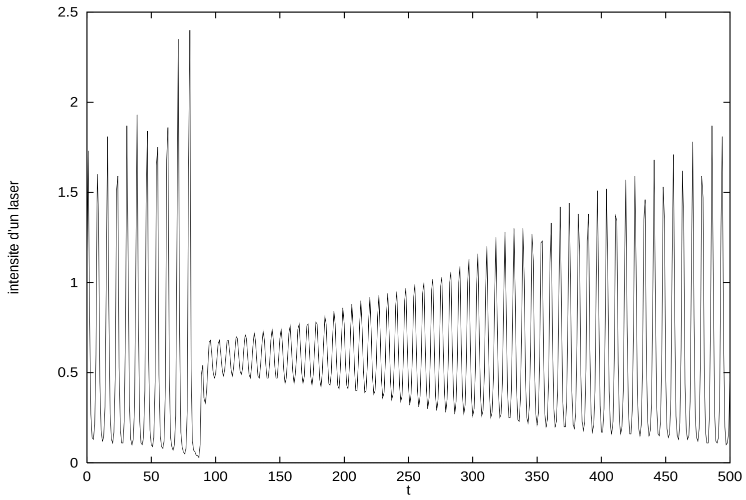


FIG. 7.11 – *Les 500 premiers points de la série laser.*

Ces données permettent, comme nous le verrons par la suite, de valider les résultats obtenus sur la série simulée artificiellement.

7.3.2.1 Description des données

Comme on peut le voir sur la fig.7.11, le processus passe par trois phases distinctes. La première est une période de lente et constante augmentation des oscillations, la seconde correspond à une augmentation moins régulière des oscillations, la troisième, enfin, est constituée d'un effondrement subit. Cet effondrement est caractéristique d'un comportement chaotique, la série évolue vers un attracteur puis change brutalement de régime. Le comportement de la série autour de cet attracteur et l'effondrement en découlant sont très difficilement prévisibles.

13. Plus de 30 articles traitent de l'étude de cette série, devenue un *benchmark*, et de sa modélisation. On pourra les retrouver au site WWW suivant: `http://www.cs.colorado.edu/~andreas/TSWelcome.html`

7.3.2.2 Architecture et apprentissage du modèle

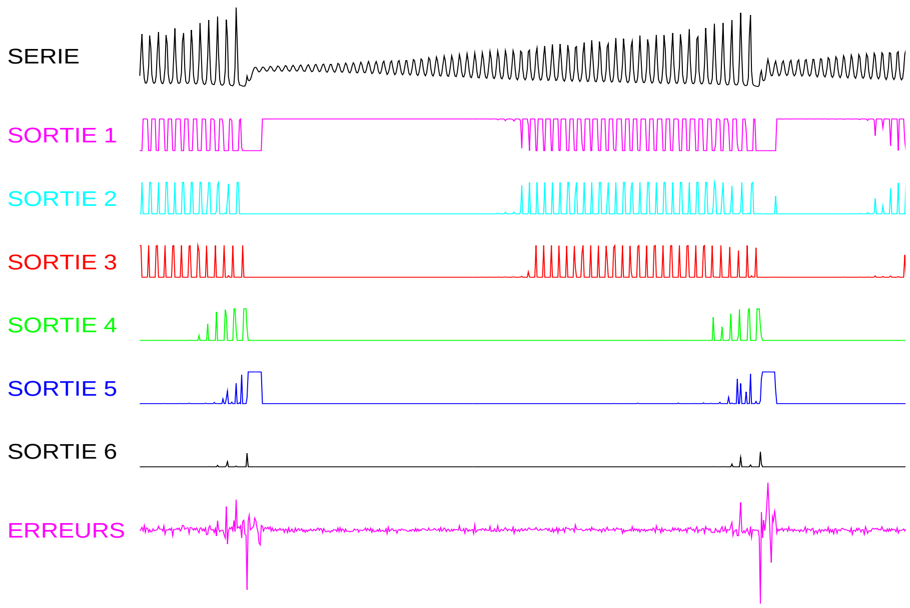


FIG. 7.12 – *Sorties du contrôleur (partie de la série appartenant à la base de test). La somme des sorties est égale à 1. Notez leur aspect binaire (juste après l'effondrement, la première sortie est sensiblement égale à 1, jusqu'à ce que l'amplitude des oscillations n'excède un certain niveau).*

On utilise, pour cet exemple, 8 experts neuronaux comportant 1 couche cachée munie de 5 tanh-neurones et d'un neurone de sortie linéaire; et un réseau contrôleur comportant une couche cachée munie de 10 tanh-neurones. La prévision d'un point se fera à l'aide des 10 retards précédents. Experts et contrôleurs seront alimentés par ces 10 mêmes entrées. On dispose pour l'apprentissage de 10000 points, 1250 autres points serviront de base de test. La distribution de ces points étant très stable, ces bases contiennent un ensemble de points consécutifs, et non pas des points tirés aléatoirement. Le meilleur modèle, en terme de performances, utilise 6 experts actifs et 2 experts inutilisés (les sorties associées du contrôleur sont nulles ou quasi nulle pour l'ensemble des exemples des bases). Pour 5 jeux de poids initiaux différents, les performances sont similaires (variantes de moins de 5%) et le nombre d'experts actifs varie entre 5 (3 essais) et 6 (2 essais).

7.3.2.3 Analyse de la modélisation

Figure 7.12 indique que le contrôleur associe les experts 4, 5 et 6 à la prévision de l'effondrement. Les autres experts actifs (1, 2 et 3) se partagent la prévision du reste de la série. Parmi ces trois experts, l'un d'eux (expert 1) s'occupe du post-effondrement (les plus petites oscillations) et des « vallées » (parties inférieures de la série) lors de plus grandes oscillations. Les deux autres gèrent les « pics » (parties supérieures de la série).

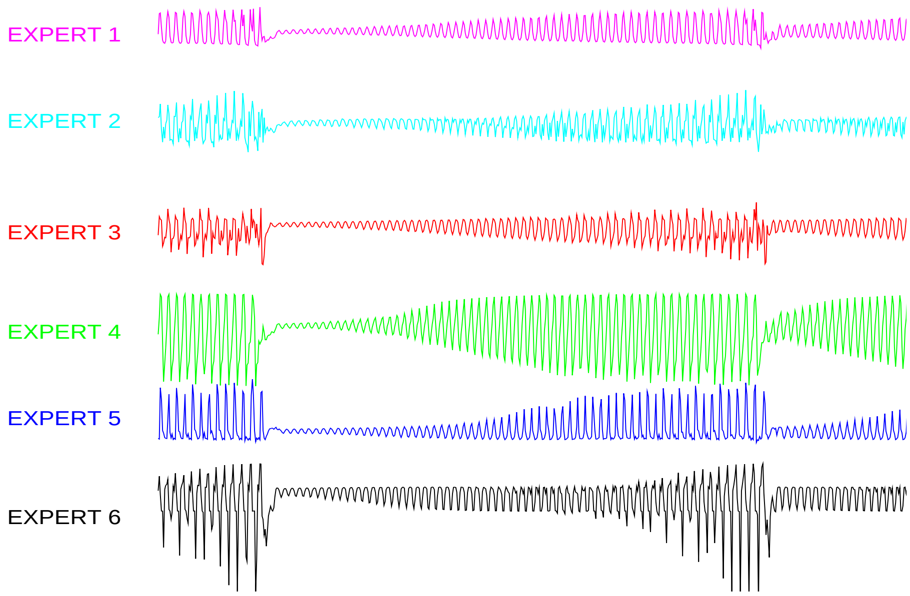


FIG. 7.13 – *Sorties des experts (prévision à un pas). Ces sorties ne sont pas pondédérées par celles du contrôleur. On peut observer les différences de prévision entre les experts 4 à 6 qui tentent de modéliser l'effondrement et les experts 1 à 3 qui modélisent le reste de la série.*

En fait, la segmentation est particulièrement intéréssante, puisque leur nature est de deux types :

- segmentation de la forme (pics, vallées)
- segmentation des régimes (pre-effondrement, effondrement, post-effondrement)

Figure 7.13, on remarque aussi que l'expert 5 produit de bonnes prévisions pour l'effondrement. L'expert 1, lui, produit de bonnes prévisions pour le post-effondrement mais des prévisions de trop faibles amplitudes pour le pre-effondrement.

On peut aussi noter qu'il n'y a pratiquement aucun sur-apprentissage (figure 7.14) et que les valeurs des variances à la fin de la convergence sont très disparates (l'échelle en y de la figure 7.15 est logarithmique). La plus petite variance est associée à l'expert 1, expert qui gère les prévisions de la partie stable constituée par le post-effondrement, la plus grande à celui qui tente de prévoir l'effondrement. On peut aussi noter que les transitions de phases durant l'apprentissage correspondent avec de larges variations de certaines variances (en particulier la $6^{ème}$), montrant le lien étroit entre apprentissage et réglage des variances.

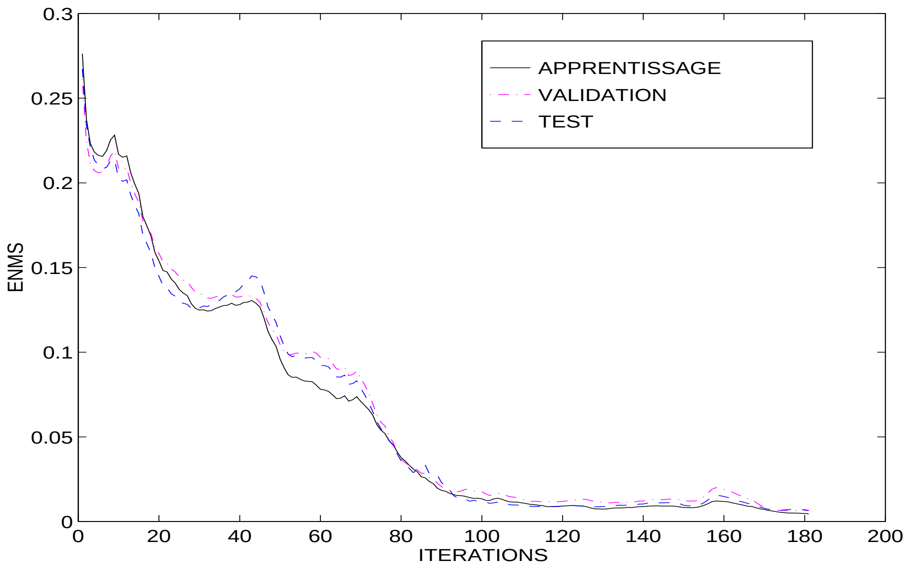


FIG. 7.14 – *Courbes d'apprentissage. Notez l'absence de sur-apprentissage, et les phases de transition dues au réglage des variances.*

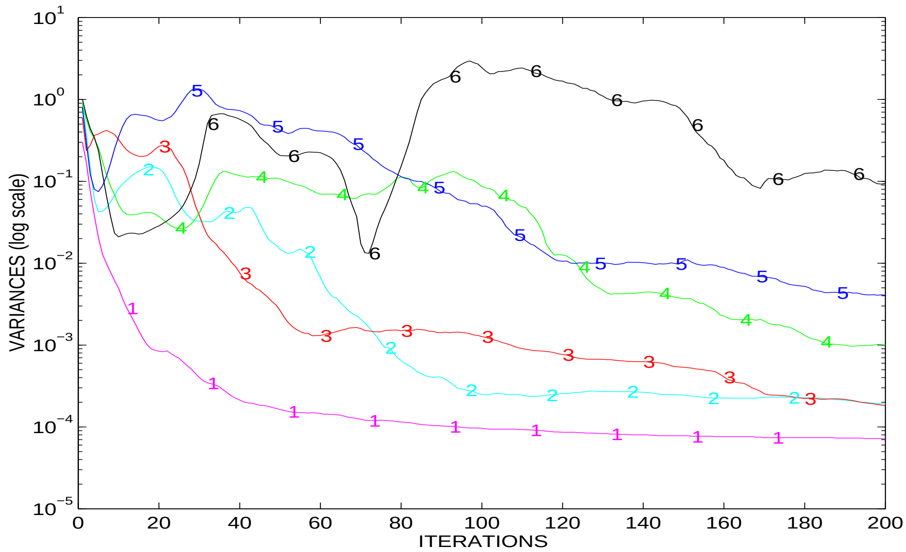


FIG. 7.15 – *Évolution des variances. Les chiffres de 1 à 6 indicent les experts. Ces indices correspondent à ceux utilisés pour les figure 7.12 and 7.13.*

Chapitre 8

Recherche de structures de modèle par algorithmes génétiques

8.1 Introduction

On a vu dans les chapitre 5 et 6 des méthodes qui sont utiles dans la résolution des problèmes d'estimation des paramètres et d'identification de modèle neuronaux sous réserve que l'hypothèse sous-jacente de débuter l'étape d'identification avec un modèle dominant est satisfaite. S'il est facile en théorie, en connectant fortement un modèle neuronal, d'obtenir un modèle dominant, on s'aperçoit dans la pratique que le choix des connexions est important, et peut faciliter ou au contraire rendre difficile la convergence vers le vrai modèle susceptible de rendre compte de la dynamique globale de la série. On propose dans ce chapitre une méthode empirique de détermination de modèle dominant basée sur une recherche stochastique de structure de modèle. Du fait de la forme particulière de la fonction à optimiser (les performances d'un modèle en fonction de sa structure) on choisit d'utiliser les algorithmes génétiques pour effectuer cette recherche.

Puisque la structure du modèle est déterminée de manière automatique, on choisit aussi d'élargir la classe des modèles à propagation directe à couches (modèle $\mathrm{NAR}_n(p)$ de la définition 4.8). On a vu précédemment (section 4.2.1) qu'une seule couche suffit pour modéliser une fonction continue sur une compact. Pourtant, il semble que pour modéliser de fortes ruptures, l'ajout d'une deuxième couche cachée apporte de meilleurs résultats (cf. (Mangeas et al., 1993)). D'autres architectures sans structure de couches peuvent être envisagées, tout en gardant le principe des réseaux sans boucles munis de fonctions d'activation non-linéaires : les « réseaux à propagation directe sans notion de couches ». Une des caractéristiques de ces modèles neuronaux est qu'une partie de la transformation de l'entrée vers la sortie peut être linéaire, et que le nombre de compositions successives des fonctions d'activation peut être de l'ordre du nombre des neurones cachés. Un exemple de tel réseau est donné figure 8.1.

Définition 8.1 :

On appelle modèle neuronal à propagation directe sans notion de couches *un modèle de la forme :*

$$X_t = \sum_{i=1}^{p} \theta_i^{n+1} X_{t-i} + \sum_{j=1}^{n} \theta_{p+j}^{n+1} h_j \tag{8.1}$$

où :

- *p est le nombre de retards nécéssaires à la modélisation, $p > 0$.*
- *n est le nombre de neurones cachés.*
- *$h_j = \psi(\sum_{i=1}^{p} \theta_i^j X_{t-i} + \sum_{k=1}^{j-1} \theta_{p+k}^j h_k)$, ψ fonction d'activation de $\mathbb{R}$ dans $\mathbb{R}$, non polynômiale.*
- *$(\theta_{i+1}^j)_{1 \leq j \leq n+1,\ 1 \leq i < p+j}$ est le vecteur des paramètres.*

Ces modèles sont peu utilisés à notre connaissance car leur structure très générale est difficile à manipuler. Cette famille de modèles inclut en particulier celle des perceptrons multicouches.

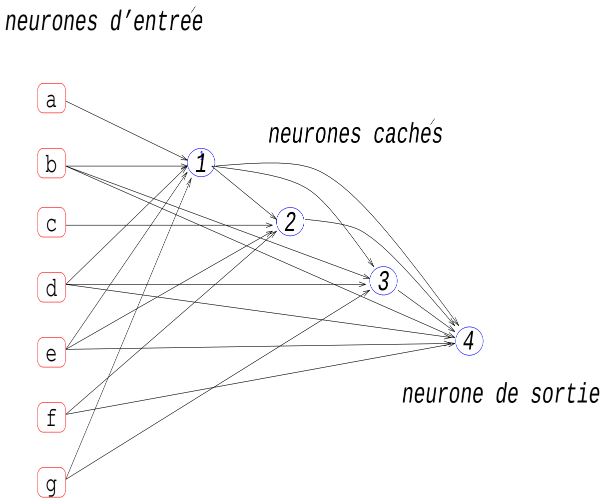


FIG. 8.1 – *Exemple de réseau à propagation directe sans notion de couches. Celui-ci à 7 entrées, 3 unités cachées et une sortie.*

Dans la section 8.2, on détaille l'algorithme de recherche stochastique qui permet de trouver l'architecture de tels réseaux. Il est de la famille des algorithmes génétiques. Finalement, on évalue les performances des modèles trouvés dans la section 8.3.

8.2 Introduction aux algorithmes génétiques

Les algorithmes génétiques sont des méthodes stochastiques d'optimisation de fonctions. Comme les réseaux de neurones, ils sont inspirés de phénomènes biologiques. Leur principe repose sur la sélection naturelle. Initialement, on dispose d'un ensemble d'individus - la population - et d'une fonction qui évalue leurs efficacités pour un problème donné (fonction à optimiser). On fait alors évoluer cette population en combinant les individus les plus performants. Nous donnons dans la suite une introduction succinte aux algorithmes génétiques. Pour plus de détails, on peut se reporter aux ouvrages de (Goldberg, 1989) et (Davis, 1991).

Concrètement, on représente un individu par un chromosome défini comme une suite de 1 et de 0. Le moteur de l'évolution est la reproduction qui fait intervenir divers opérateurs comme le *crossing-over* ou la *mutation* (figure 8.2). Les individus sont sélectionnés avec une probabilité proportionnelle à leur efficacité pour prendre part au processus de reproduction. Ainsi, ce sont les caractéristiques les plus intéressantes de la population qui survivent et qui sont combinées entre elles pour produire des individus encore plus efficaces.

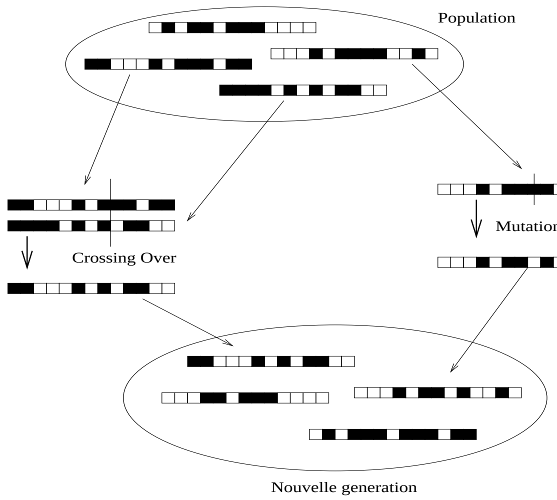


FIG. 8.2 – *Opérateurs* mutation *et* crossing-over.

Les algorithmes génétiques présentent certains avantages par rapport aux méthodes traditionnelles. Il s'agit tout d'abord d'une méthode générale et facilement transposable, puisque l'on travaille sur une représentation des objets, et non sur les objets eux-mêmes. L'évolution se fait sur les chromosomes et non sur les individus. La sélection naturelle est aveugle : il n'y a aucune mémorisation des propriétés de l'individu autres que celles inscrites dans ses gènes. La population peut ainsi évoluer vers une solution intéressante en dehors de toute considération sur la nature des objets manipulés. Il est également intéressant de travailler sur un ensemble d'individus : on peut ainsi trouver plusieurs voisins d'une solution optimale, ou plusieurs minima locaux. Il est même possible de travailler indépendamment sur plusieurs minima locaux. On peut aussi travailler indépendamment sur plusieurs populations isolées, les faire converger séparément, puis les mélanger pour obtenir de nouveaux minima plus intéressants.

Remarques :

- Les algorithmes génétiques sont particulièrement bien adaptés pour déterminer les minima de fonctions ne possédant pas les propriétés de dérivabilité nécessaires à l'emploi de méthodes plus standards. Pour ces fonctions, il n'existe que peu d'outils de recherche automatique dans l'espace complet des solutions.
- Dans le cas des fonctions dérivables, les algorithmes génétiques peuvent être utilisés comme une première phase de minimisation pour trouver les bassins d'attraction d'une fonction, les descentes aux minima de chacun des bassins peuvent être réalisés par toute autre méthode (gradient, méthodes du second ordre ...).
- Il est bien clair que rechercher le minimum d'une fonction convexe avec des algorithmes génétiques n'est pas approprié, bien que des exemples aussi simplistes soient relativement fréquents dans la littérature.

8.2.1 L'algorithme de base

Les différents algorithmes génétiques découlent tous d'un algorithme de base auquel on apporte des modifications en fonction du problème traité, pour en améliorer les performances. Commençons par décrire cet algorithme de base (voir figure 8.3).

Cet algorithme commence par l'initialisation de la population de départ. Nous travaillons ici avec des chromosomes binaires de taille fixe. La population initiale comprend un certain nombre de chromosomes (une vingtaine par exemple) pour lesquels chaque bit est pris aléatoirement égal à 0 ou 1 avec une probabilité de 1/2. On évalue ensuite chaque individu de la population, en mesurant la valeur de la fonction de coût pour chacun.

Une fois cette phase d'initialisation terminée, on produit de nouvelles générations jusqu'à atteindre un nombre limite d'individus. Pour créer un individu, on sélectionne un opérateur, *crossing-over* ou *mutation* (voir figure 8.2), avec une probabilité 1/2, puis un ou deux parents selon l'opérateur, et on effectue la reproduction. Nous aboutissons ainsi à la nouvelle génération qui servira de base à la suivante. On y rajoute les meilleurs individus de la génération précédente (l'élite), ce qui permet d'avoir à tout instant dans la population les meilleurs individus générés. Les tirages aléatoires sont tous effectués suivant une loi uniforme.

Le crossing-over : Deux parents de la population initiale ont été sélectionnés par tirage aléatoire. On tire alors aléatoirement un bit suivant une loi uniforme. Si le bit tiré est le $k^{ème}$, alors les k premiers bits du nouvel individu seront ceux du premier parent, les suivants ceux du deuxième parent.

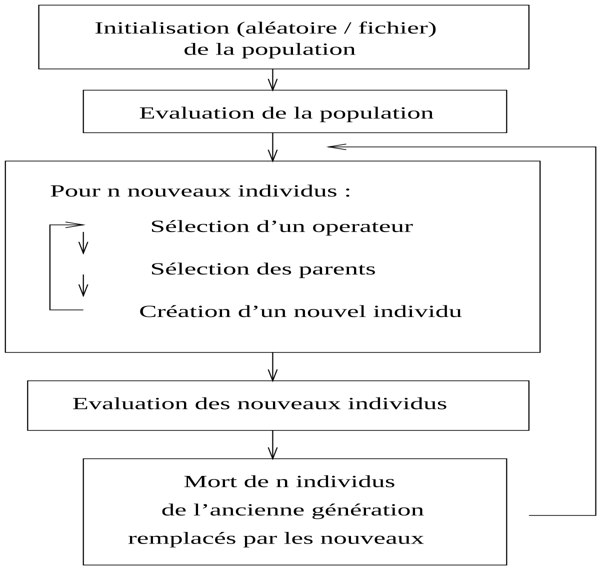


FIG. 8.3 – *Déroulement d'un algorithme génétique de base.*

La mutation : Un seul parent est tiré aléatoirement. On choisit au hasard un bit, alors le nouvel individu est identique à son père excepté pour le bit choisi dont la valeur est inversée.

Cet algorithme de base n'est malheureusement, dans bien des cas, guère plus satisfaisant qu'un simple tirage aléatoire. Il est donc capital de le modifier en fonction du problème traité pour augmenter sa capacité de convergence. C'est ce que nous verrons ultérieurement.

On peut dès à présent remarquer que cet algorithme, comme ceux qui en découlent, est très facilement parallélisable. Étant donné que la plus grande partie du temps de calcul est consacrée à l'évaluation des nouveaux individus, et que ces évaluations sont indépendantes, il suffit de donner à différents processeurs une partie de la nouvelle génération à évaluer.

8.2.2 Codage des architectures sous forme de chromosomes

Pour rechercher une architecture neuronale optimale, nous nous plaçons dans l'ensemble des réseaux à propagation directe dont le nombre maximal de neurones cachés est fixé a priori. On note alors $RPD(m)$ l'ensemble des réseaux à propagation directe munis de $m-1$ neurones cachés et d'un neurone de sortie. Ce qui permet de travailler sur des réseaux dont le nombre de connexions est majoré.

Nous avons choisi de coder sous forme de chromosome l'existence (1) ou la non existence (0) de chacune des connexions possibles d'un réseau quelconque de $RPD(m)$. Par conséquent, pour un réseau quelconque de cet ensemble, une connexion est codée sur un bit. Ce réseau étant sans cycle, on peut définir une relation d'ordre sur les neurones cachés du réseau telle que s'il existe une connexion du neurone i vers le neurone j alors $i < j$ (l'indice m correspondant toujours au neurone de sortie). Une fois les neurones cachés ordonnés, on les note $\{N_1, N_2, \ldots, N_m\}$. Un neurone N_i ne peut recevoir de signal que d'un neurone N_j avec $j < i$ et chacun de ces neurones peut être connecté aux entrées. Les connexions incidentes au neurone N_i peuvent donc être codées sur $p + (i-1)$ bits, où p est le nombre de variables d'entrées. Le nombre de connexions de ce réseau est donc finalement :

$$\sum_{i=1}^{m}(p + (i-1)) = pm + \frac{m(m-1)}{2} \quad .$$

Reprenons, à titre d'exemple, le réseau présenté dans la figure 8.1 et numérotons chacun des neurones pour obtenir le codage sous la forme d'un chromosome (voir la figure 8.4).

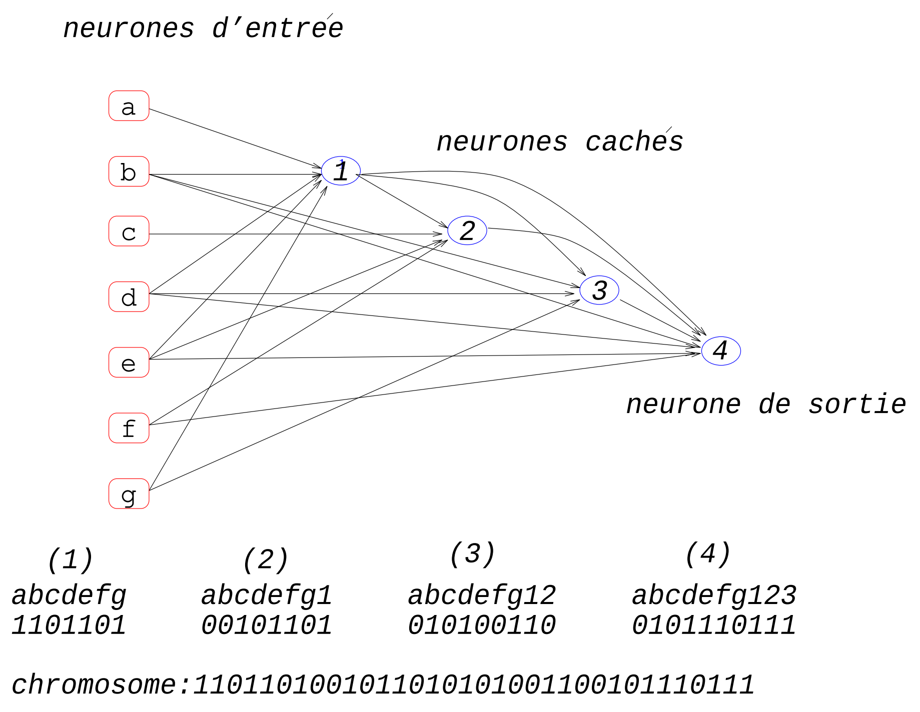


FIG. 8.4 – *Exemple de réseau à propagation directe, dont l'architecture est codée sous la forme d'un chromosome. Au bas de la figure, on peut voir le code associé aux neurones cachés (notés* (1), (2) *et* (3)*) et au neurone de sortie (noté* (4)*). Ainsi, si on détaille le code de* (1)*, le* 1 *sous le* a *signifie que le neurone d'entrée* a *est connecté au neurone caché* (1)*, alors que le* 0 *sous le* c *signifie que le neurone* c *n'est pas connecté au neurone* (1).

Remarque : dans le cas où pour au moins deux neurones cachés quelconque A et B, il n'existe de chemin ni de A vers B, ni de B vers A, alors l'ordre sur les neurones cachés est partiel et au moins deux indexations de ces neurones sont possibles. Dans ce cas, plusieurs chromosomes codent la même architecture (si n est le nombre de neurones cachés non reliés, on a $n!$ codages possibles). C'est notamment le cas pour les réseaux à couches. Compte tenu du grand nombre d'architectures générées, il semble que dans la pratique ce phénomène n'ait pas été source de difficulté de convergence.

Pour ce codage, l'opérateur de mutation correspond physiquement à l'ajout ou à la suppression d'une connexion (voir figure 8.5). Suivant que cette modification est faite en début de graphe ou en fin, le modèle neuronal résultant est plus ou moins modifié. Par exemple la suppression de la liaison entre la variable *a* et le neurone *1* aura moins de conséquences que la suppression de la connexion entre les neurones *3* et *4*. Pourtant dans ce qui a été fait ici, la probabilité de mutation est la même dans les deux cas. Le *crossing-over* correspond à la création d'une nouvelle architecture issue de la fusion de deux architectures différentes. Cet opérateur permet de combiner des caractéristiques différentes, en terme d'architecture, des deux parents.

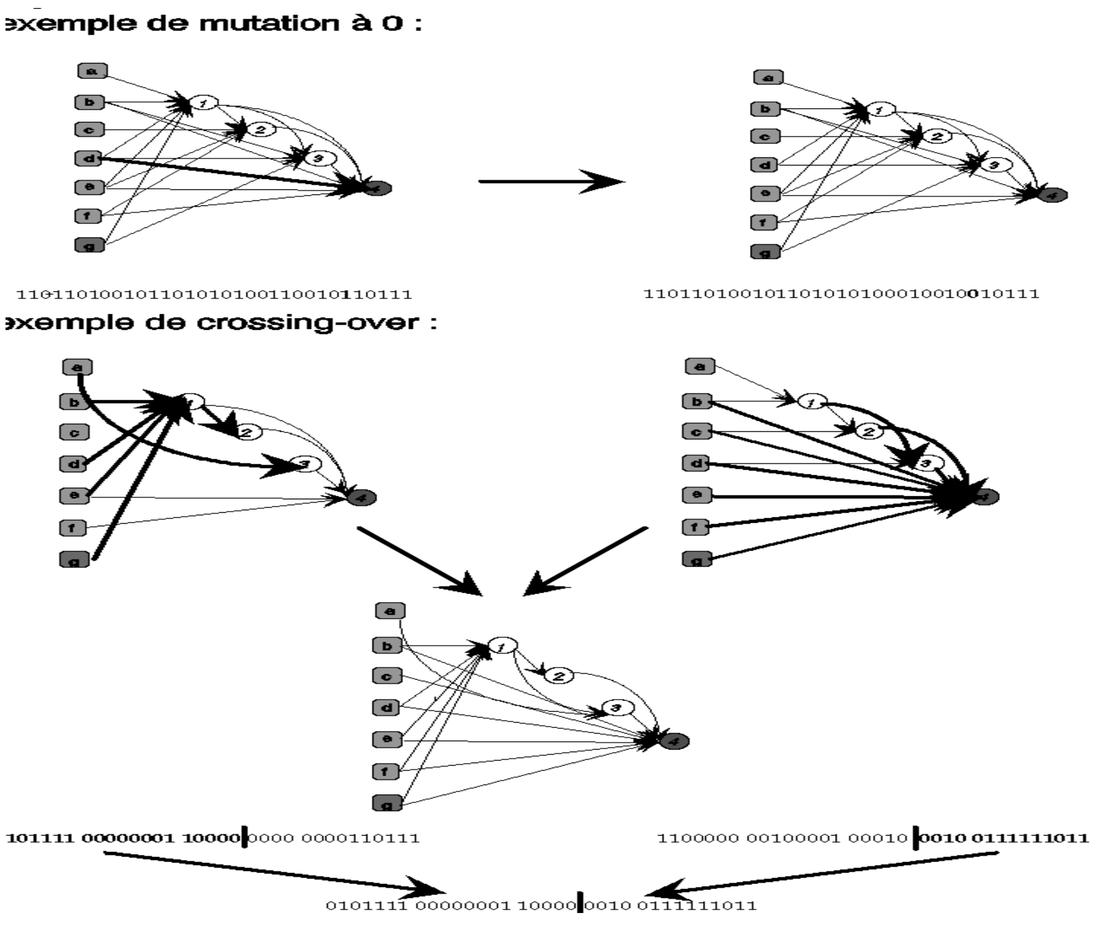


FIG. 8.5 – *Exemple de mutation et de crossing-over spécifique aux réseaux de neurones à propagation directe.*

La fonction d'évaluation de chaque architecture est indépendante de l'algorithme génétique lui-même. Elle mesure la qualité de généralisation sur des données n'ayant pas servi à calculer les poids. Pour avoir une fonction croissante, elle est de la forme $\hat{\theta} \mapsto 1 - NMSE(\hat{\theta})$, où $\hat{\theta}$ est le vecteur des paramètres estimé sur la base d'apprentissage et où $NMSE(\theta)$ est défini chapitre 5, définition 5.2.

8.2.3 Choix des opérateurs

Comme nous l'avons déjà souligné dans le chapitre 2.1, l'algorithme de base n'est pas très performant. Étant donnée la durée de l'évaluation d'une architecture, il est nécessaire d'améliorer l'algorithme pour limiter le nombre d'architectures à générer. Des améliorations de diverses natures ont été faites expérimentalement, elles sont exposées ci-dessous.

8.2.3.1 La pression de la sélection naturelle

Le graphique 8.6 qui représente l'évaluation [1] du meilleur individu au cours des générations pour l'algorithme aléatoire et pour l'algorithme génétique, ne montre qu'une faible différence entre les courbes. On observe surtout que les deux courbes restent très rapidement stationnaires, montrant l'incapacité des deux algorithmes à « regrouper » les caractéristiques intéressantes des individus pour progresser régulièrement.

Ceci s'explique par une pression insuffisante de la sélection naturelle. Si on prend le critère d'évaluation brute définie section 8.2.2, on a en effet des évaluations comprises entre 0.96 et 1. La probabilité de choisir un chromosome comme parent d'un nouveau chromosome étant proportionnelle à son évaluation, on aura ici presque une équiprobabilité de choix. On choisira donc aussi bien de bons parents que de mauvais.

Comment rétablir cette sélection naturelle qui fait tout l'interêt de l'algorithme génétique ? Il suffit en fait de réajuster les évaluations. Voici quelques méthodes :

1) *Translation* : On soustrait à chaque évaluation la valeur minimale des évaluations. On aura par exemple ici des évaluations comprises entre 0 et 0.04. Cette méthode qui peut paraître simple et efficace a un inconvénient majeur. Supposons que nous ayons une population de 5 individus avec les évaluations suivantes :

$$991, 991.5, 991.2, 990.8, 994.5$$

Après réajustement, nous obtenons les valeurs suivantes :

$$0.2, 0.7, 0.4, 0, 3.7$$

On a effectivement rétabli la sélection naturelle. Toutefois, on va perdre ici de la richesse de la diversité génétique, car il est très probable qu'à la génération suivante, nous n'ayons que des fils du dernier individu.

2) *Transformation linéaire du classement* : On trie les individus par ordre croissant

1. Evaluation d'un réseau de neurone = 1 - ($NMSE(\hat{\theta})$ sur la base de test, où $\hat{\theta}$ est le vecteur des paramètres associé au réseau, estimé sur la base d'apprentissage). Voir la définition du critère $NMSE(\theta)$ définition 5.2.

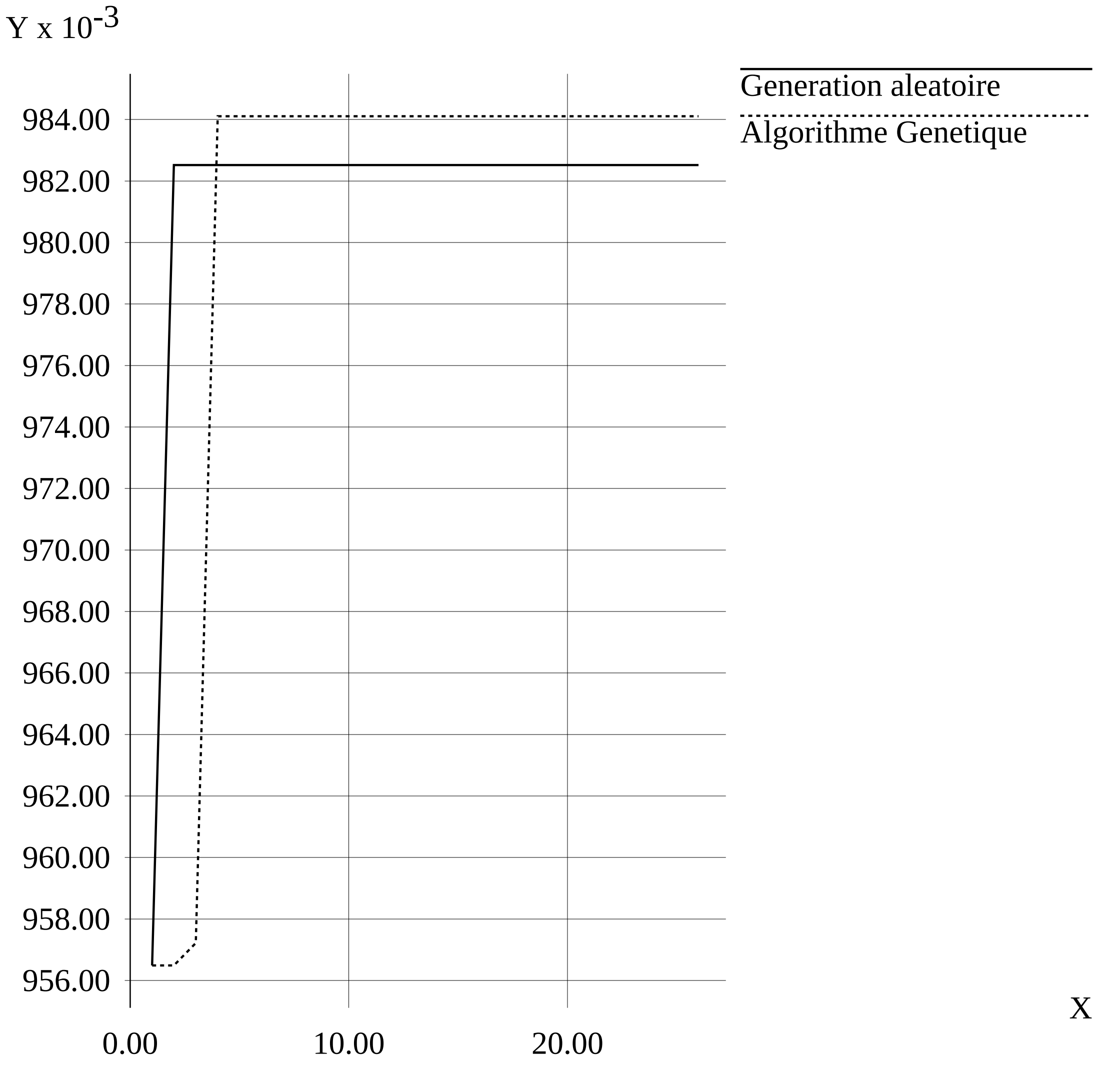


FIG. 8.6 – *Comparaison Algorithme Génétique - Génération Aléatoire. En abscisse : Nombre de générations. En ordonnée : Performance du meilleur réseau = 1 - (*NMSE$(\hat{\theta})$ *sur la base de test). Le vecteur $\hat{\theta}$ est le vecteur des paramètres associé au réseau, estimé sur la base d'apprentissage. Le critère* NMSE *est défini chapitre 5, définition 5.2.*

d'évaluation. Leur évaluation réajustée (on la note $Eval_{i,ajuste}$) est prise égale à leur classement, que l'on peut éventuellement transformer par une fonction linéaire, ce qui permet de moduler la pression de la sélection naturelle. Soit T_p la taille de la population. Un individu classé i^{ème}est ainsi noté :

$$Eval_{i,ajuste} = 1 + (\Delta - 1)\frac{i-1}{T_p - 1}$$

avec

$$\Delta = \frac{\text{Proba(Choisir le Meilleur)}}{\text{Proba(Choisir le Pire)}} \quad .$$

3) *Transformation logarithmique du classement* : On effectue un classement, que l'on modifie cette fois par une fonction logarithmique. La fonction logarithmique atténue les différences d'évaluation entre les individus les plus performants, bien que faisant une discrimination forte entre le premier et le dernier individu, ce qui permet de garder de la diversité génétique. Leur évaluation réajustée s'écrit alors :

$$Eval_{i,ajuste} = 1 + Log(1 + (e^{\Delta-1} - 1) * \frac{i-1}{T_p - 1})$$

avec

$$\Delta = \frac{\text{Proba(Choisir le Meilleur)}}{\text{Proba(Choisir le Pire)}}$$

Le graphique (8.7) donne l'avantage à la deuxième méthode, avec d'assez mauvais résultats pour les deux autres. La première sera définitivement abandonnée, car elle ne donne pas de bons résultats. La transformation logarithmique, quant à elle, donne des résultats intéressants une fois que nous aurons appliqué d'autres techniques. Nous la réutiliserons donc par la suite. Il y a en effet une diversité génétique beaucoup trop faible pour les méthodes logarithmique et translation (il ne reste respectivement plus que 5 et 3 classes d'individus), ce qui empêche toute progression.

Intéressons nous au rapport Δ. Les graphiques suivants comparent les performances pour différentes valeurs de Δ. Les différences n'apparaissent pas fondamentales au bout de 500 individus. Toutefois, lorsque l'on regarde la moyenne des évaluations des individus au cours des générations (fig. 8.8) on constate que pour un Δ trop élevé, on a une progression assez rapide, puis une chute ou au moins une stabilisation de la moyenne du niveau des individus à chaque génération. Pour des rapports trop faibles (2 ou 5), on constate d'importantes faiblesses en début de parcours avec pour certaines générations, des chutes brutales de performance. Toutefois, à long terme, des rapports Δ assez faibles finissent par donner de bons résultats. Pour conclure, on peut dire que ce rapport doit être ajusté en fonction du nombre d'itérations, afin de conserver suffisamment longtemps la diversité génétique. On utilise ici un rapport $\Delta = 10$.

On peut envisager d'abandonner la technique de sélection par roulette, c'est-à-dire avec une probabilité proportionnelle à l'évaluation de chaque individu. On remplacerait alors cette technique par celle de la sélection par tournoi : on sélectionne aléatoirement

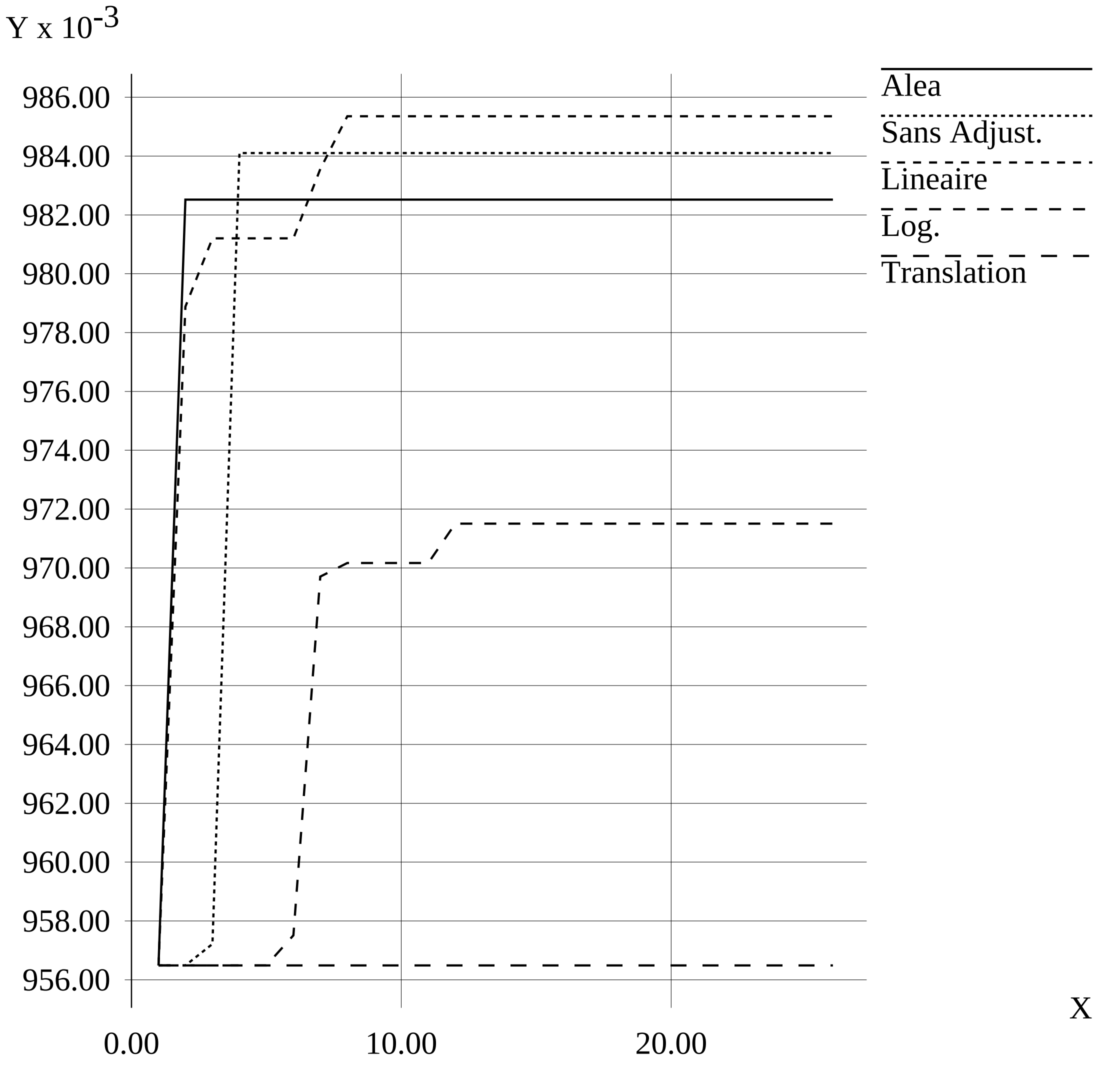


FIG. 8.7 – *Les différentes méthodes de réajustement de l'évaluation (Linéaire :* $\Delta = 10$*, Log :* $\Delta = 2$*). En abscisse : Nombre de générations. En ordonnée : Performance du meilleur réseau = 1 - (NMSE(*$\hat{\theta}$*) sur la base de test). Le vecteur* $\hat{\theta}$ *est le vecteur des paramètres associé au réseau et estimé sur la base d'apprentissage. Le critère* NMSE *est défini chapitre 5, définition 5.2.*

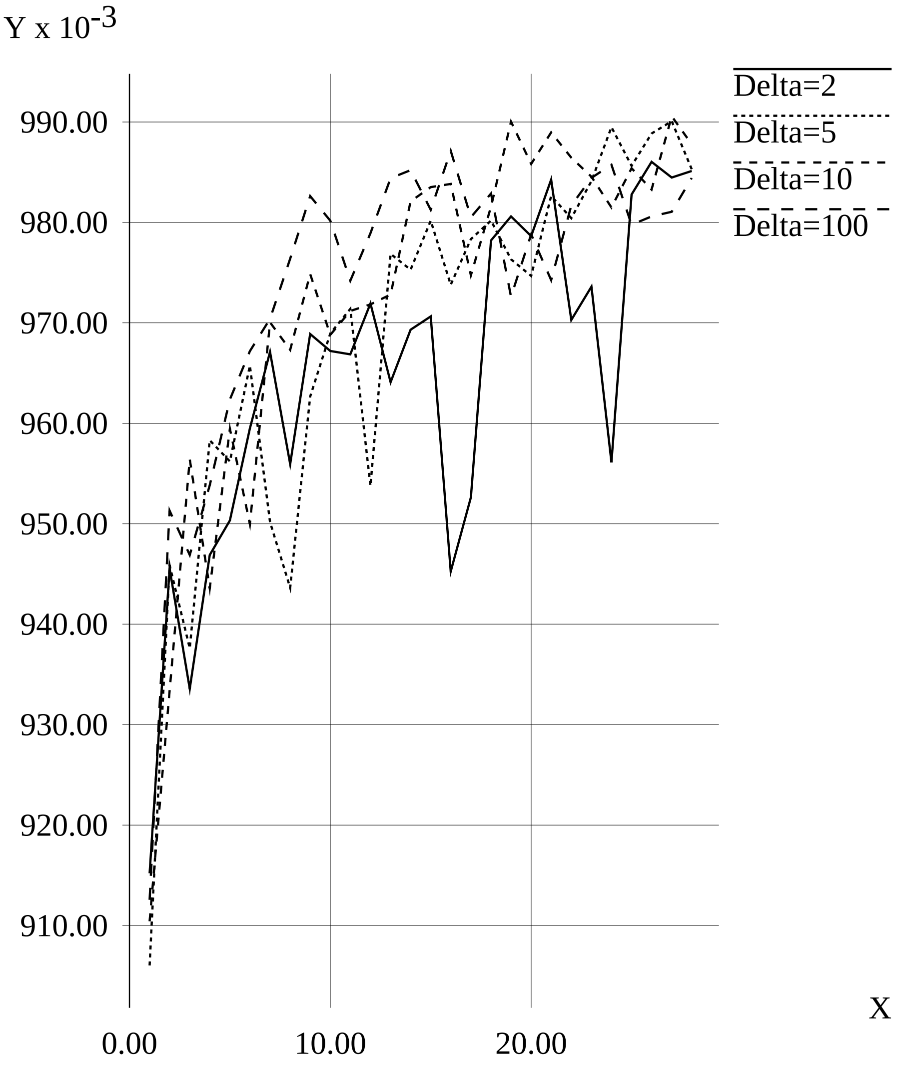


FIG. 8.8 – *Moyenne des évaluations des individus pour différentes valeurs de* Δ *(Ajustement linéaire). En abscisse : Nombre de générations. En ordonnée : Performance du meilleur réseau = 1 - (NMSE(*$\hat{\theta}$*) sur la base de test). Le vecteur* $\hat{\theta}$ *est le vecteur des paramètres associé au réseau et estimé sur la base d'apprentissage. Le critère* NMSE *est défini chapitre 5, définition 5.2.*

k individus, et on garde le meilleur comme parent de la nouvelle reproduction. Cette méthode n'apporte en fait pas d'idée radicalement nouvelle, puisqu'elle revient à modifier le classement des individus non pas par une fonction logarithmique ou linéaire, mais par une fonction puissance (d'exposant $k-1$). En effet, classons les individus par ordre croissant d'efficacité. Si S est la variable aléatoire qui désigne le classement de l'individu sélectionné alors :

$$\begin{aligned} P(S=i) &= \sum_{i_1,i_2,\ldots,i_k} \Big(P(S=i/i_1,i_2,..i_k)P(i_1,i_2,..i_k) \Big) \\ &= \sum_{\substack{i_1,i_2,\ldots,i_k \leq i \\ Max(i_1,i_2,\ldots,i_k)=i}} \frac{1}{N^k} \\ &= \sum_{i_1,i_2,\ldots,i_k \leq i} \frac{1}{N^k} - \sum_{i_1,i_2,\ldots,i_k < i} \frac{1}{N^k} \\ &= \frac{i^k-(i-1)^k}{N^k} \end{aligned} \tag{8.2}$$

Le graphique (8.9) montre les probabilités de sélection des différents individus en fonction de leur classement avant et après réajustement. On constate que sans réajustement, la courbe est plate. Les fonctions concaves ont l'avantage, pour un même rapport de probabilité de sélection entre le premier et dernier individu, d'être moins sélectives au niveau des meilleurs individus (faible pente). Comme il apparaît plus intéressant de modifier ce classement par une fonction concave que par une fonction convexe, la méthode de sélection par tournoi a été abandonnée.

8.2.3.2 Nouveaux opérateurs

Compte tenu du temps relativement long d'évaluation d'une architecture de réseaux de neurones, l'algorithme génétique doit être rapide, et la meilleure architecture doit être obtenue en un nombre minimal de créations de nouveaux individus. On présente ainsi deux nouveaux opérateurs plus complexes et plus performants, issus des deux opérateurs de base :

- *Le crossing-over universel* qui, comme pour le *crossing-over* simple, nécessite deux parents. Cependant, pour chaque élément du nouveau chromosome, on choisit aléatoirement s'il vient du chromosome père ou du chromosome mère (voir figure 8.10).
- *les opérateurs de mutation à 1 ou de mutation à 0*, qui correspondent à l'ajout ou à la suppression d'une connexion. On introduit aussi l'opérateur de mutation multiple, qui permet de modifier aléatoirement de 1 à 5 bits. On ajoute, de même, un opérateur qui génère aléatoirement un chromosome et qui permet de réintroduire de nouveaux gènes dans la population. Il faut remarquer que le choix de bons opérateurs est essentiellement lié au problème traité.

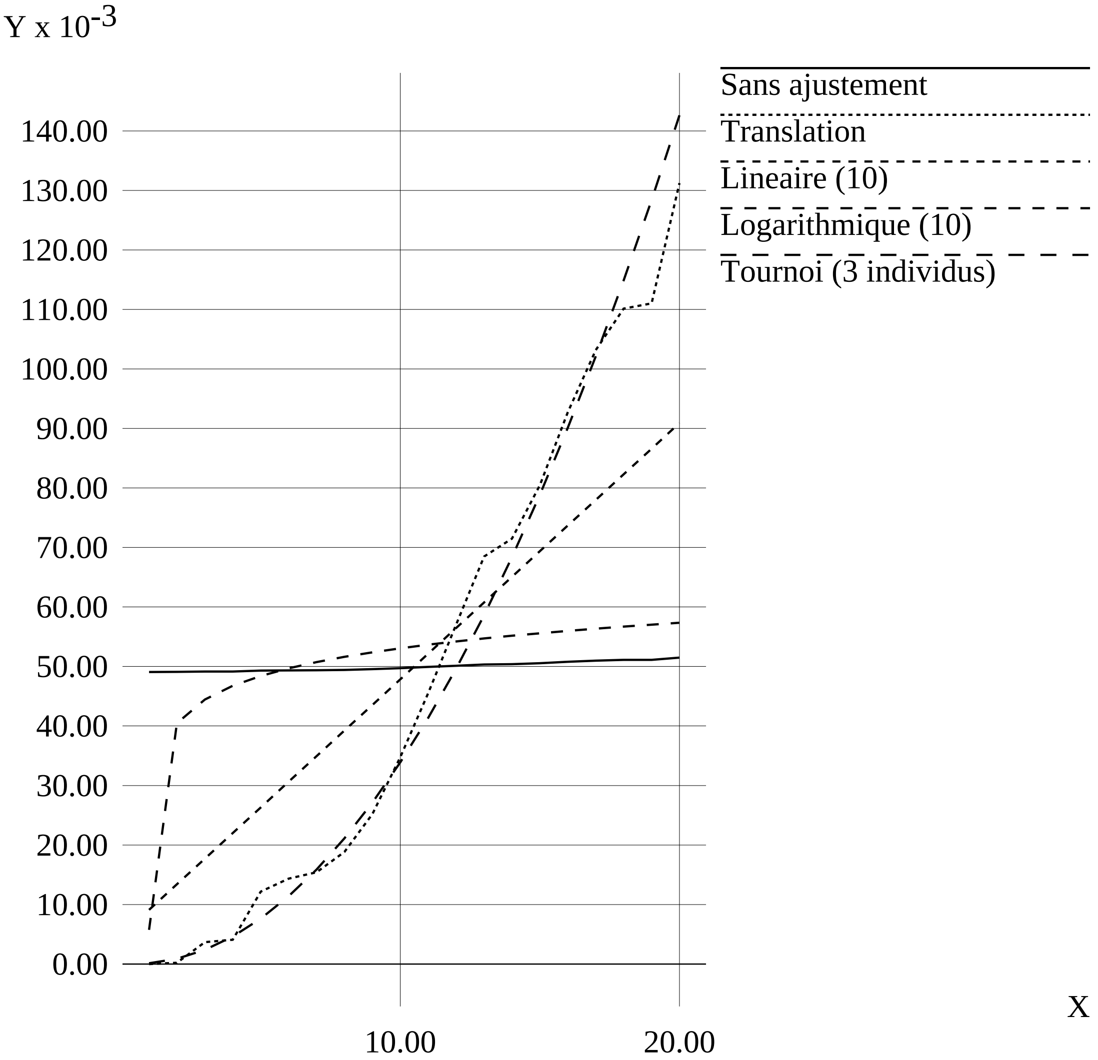


FIG. 8.9 – *Probabilités de sélection en fonction du classement des individus pour différentes méthodes de réajustement (population de 20 individus). En abscisse : Nombre de générations. En ordonnée : Performance du meilleur réseau = 1 - (NMSE($\hat{\theta}$) sur la base de test). Le vecteur $\hat{\theta}$ est le vecteur des paramètres associé au réseau et estimé sur la base d'apprentissage. Le critère* NMSE *est défini chapitre 5, définition 5.2.*

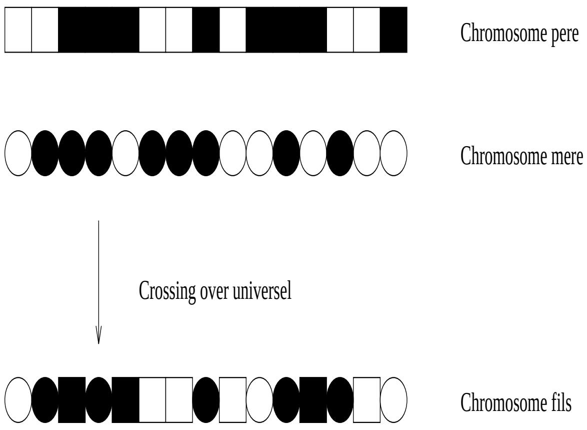


FIG. 8.10 – *Exemple de crossing-over universel.*

Choix des opérateurs pendant l'optimisation

Il est évident que l'opérateur de *crossing-over* universel est plus « utile » que l'opérateur aléatoire. Il est donc nécessaire de pouvoir choisir indépendamment les probabilités de sélection des différents opérateurs. On met pour cela à chaque opérateur une note, et la probabilité de sélection d'un opérateur sera proportionnelle à cette note.

Voici les notes mises arbitrairement et initialement aux différents opérateurs :

Crossing-over Universel	180
Mutation multiple (de 1 à 5)	100
Mutation à 1	90
Mutation à 0	60
Création aléatoire	15

L'opérateur « aléatoire » crée une nouvelle architecture en utilisant un générateur aléatoire d'architecture cohérente. Cette opérateur permet ainsi d'ajouter des individus susceptibles de posséder des caractéristiques nouvelles (des chemins de connexions n'appartenant à aucun individus de la population).

On voit ici plus clairement l'intérêt des opérateurs « Mutation à 1/0 » par rapport à l'opérateur « Mutation quelconque ». En effet, en augmentant la note du « Mutation à 1 » par rapport à « Mutation à 0 », on peut orienter l'algorithme génétique vers une densification du réseau, si l'on estime par exemple que les réseaux aléatoirement choisis au début sont trop peu denses.

Le rôle de l'opérateur de crossing-over est de combiner les caractéristiques intéressantes des individus tandis que le rôle de la mutation est d'introduire de la diversité génétique. Selon l'état de la population, un opérateur peut être plus adapté qu'un autre, et il peut donc être intéressant de modifier les notes de chaque opérateur en fonction de ses performances au cours de la génération précédente. Mais qu'entend-on par performance ? Pour calculer l'efficacité d'un opérateur, on s'intéresse à la moyenne des évaluations des reproductions qu'il a effectuées. On augmente les notes proportionnellement à cette moyenne. On peut à l'aide de coefficients modifier l'importance de ces réajustements. Ces modifications sont faites automatiquement durant la recherche du minimum par l'algorithme.

Dans la section suivante, nous reviendrons précisément sur l'algorithme final utilisé, ainsi que sur les modalités de comparaison des différentes architectures, avant de comparer les résultats sur différentes séries.

8.3 Comparaison des performances de différents modèles neuronaux

Pour chacune des trois séries que nous allons modéliser, nous appliquons le même protocole.

Les données

On sépare l'ensemble des données disponibles pour chacune des séries étudiées en trois parties qui sont appelées par la suite : base d'apprentissage, base de validation et base de test. Suivant les cas et notamment suivant le nombre de données disponibles, les trois bases représenteront respectivement soit 1/3, 1/3, 1/3 soit 1/2, 1/4 et 1/4 des données. La base de test sert exclusivement à quantifier les qualités de généralisation des modèles, à aucun moment les architectures ou les paramètres ne seront déterminés en fonction des éléments la composant.

Les réseaux de neurones comparés

On rappelle ici que le but est de trouver un réseau (voir chapitre 6 susceptible de rendre compte convenablement de la dynamique du phénomène étudié afin de faciliter les étapes d'estimation et d'identification. On compare en particulier la méthode de recherche par algorithmes génétiques, à celle exposée section 6.3 (méthode que l'on nomme ici « élagage » puisque l'on supprime des connexions), en soulignant que la recherche du modèle dominant s'est effectuée suivant la méthodologie de la section 6.2.3. La méthode d'élimination des poids SSM peut ainsi être appliquée aux modèles issus de la recherche stochastique, bien que nous ne l'ayons pas fait ici. On compare aussi les performances des modèles trouvés en utilisant les algorithmes génétiques à la méthode triviale du « early stopping » (arrêt prématuré en français). Cette méthode est décrite en détails dans (Weigend et al., 1990). L'architecture est fixée suivant la méthode de la section 6.2.3 ou choisie par l'utilisateur.. La détermination des paramètres se fait conjointement sur les bases d'apprentissage et de validation, de la façon suivante. La minimisation de l'erreur quadratique normalisée ($NMSE(\theta)$ voir définition 5.2) est faite par rapport à la base d'apprentissage. À chaque itération, on calcule aussi la valeur de cette fonction sur la base de validation. Tant que l'erreur quadratique normalisée diminue sur la base de validation, on continue la minimisation par rapport à la base d'apprentissage. Dès que cette valeur croît, on stoppe l'apprentissage et on retient les valeurs des paramètres à ce point (voir figure 8.11).

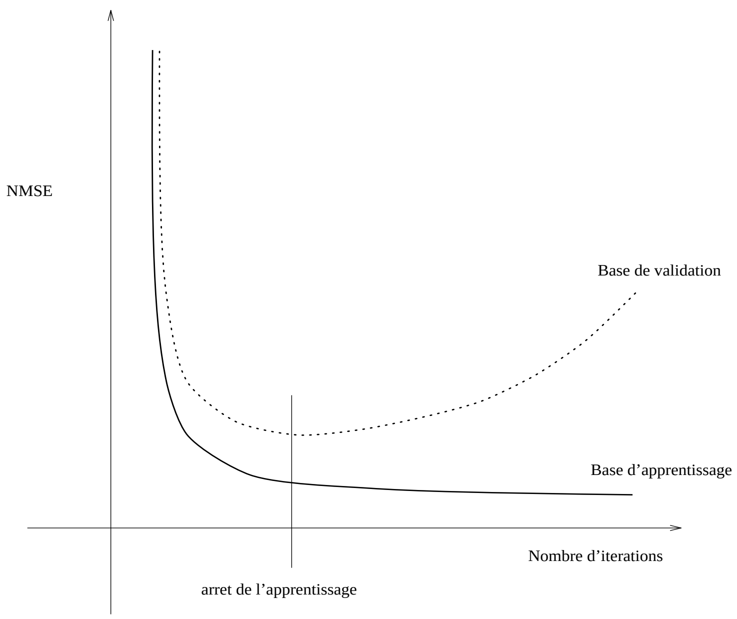


FIG. 8.11 – *Méthode de l'« arrêt prématuré ».*

Comparons ces méthodes :

- Pour les deux méthodologies, (*SSM* et l'arrêt prématuré), l'architecture initiale est à la charge de l'utilisateur (on prendra dans la pratique la même architecture pour ces deux méthodes.
- L'algorithme de l'arrêt prématuré nécessite obligatoirement une troisième base de données, la base de test, pour quantifier les performances en généralisation du réseau ainsi déterminé. A contrario, en ce qui concerne la méthode d'élagage, une seule base, la base d'apprentissage, est nécessaire. La base de validation peut être incluse dans la base d'apprentissage. Pour des raisons pratiques, on ne l'a pas fait ici, et cette méthode se trouve légèrement pénalisé.
- L'algorithme d'élagage converge plus lentement (en terme de temps CPU), puisque plusieurs minimisations successives sont effectuées alors qu'une seule est nécessaire pour la méthode de l'arrêt prématuré et qu'un nombre équivalent au nombre paramètres éliminés st nécessaire pour la méthode *SSM*.
- Dans les deux cas, le but est de réduire le surapprentissage et donc d'améliorer la généralisation.
- Pour l'élagage, l'architecture finale est incluse dans l'architecture initiale (avec un nombre de paramètre réduit, contrairement à la méthode de l'arrêt prématuré où elle reste invariante.

Dans les cas des deux méthodes choisies pour effectuer la comparaison, les réseaux obtenus sont des réseaux à couches de la définition 4.8 dont les étapes d'estimation et d'identification sont plus maîtrisées que dans le cas de réseaux de neurones sans notion de couches. Le nombre maximal de neurones est fixé arbitrairement, il sera pris égal au nombre de neurones du perceptron multicouches utilisé pour l'arrêt prématuré.

On détermine un schéma d'opération qui sera appliqué à chaque tentative de modélisation.

– *Évaluation d'un réseau* : la minimisation de l'erreur quadratique normalisée ($\mathrm{NMSE}(\theta)$) est faite sur la base d'apprentissage. L'évaluation du réseau en généralisation est donnée par la valeur du $\mathrm{NMSE}(\hat{\theta})$ sur la base de validation, où $\hat{\theta}$ est le vecteur des paramètres estimé. Le réseau finalement obtenu sera donc déterminé grâce aux données contenues dans les deux bases : la base d'apprentissage pour déterminer les valeurs des paramètres ; la base de validation, pour déterminer la meilleure architecture.

– *Population initiale*: la taille de la population initiale est de vingt individus. Ces individus peuvent être choisis aléatoirement, ou être des réseaux d'architectures établies par d'autres méthodes.

– *Sélection des parents* : ils sont choisis avec des probabilités proportionnellement à une transformation logarithmique de leurs évaluation.

– *Taille des futures populations* : vingt nouveaux individus sont créés à chaque étape. Une élite est constituée des meilleures architectures de la population à l'étape t et sera incluse dans la population à l'étape $t+1$.

– *Poids des opérateurs* : les opérateurs utilisés sont le crossing-over universel, la mutation multiple, la mutation à 1, la mutation à 0 et l'opérateur de création aléatoire. Des valeurs initiales des poids de chacun de ces opérateurs sont données, mais ils évoluent ensuite automatiquement en fonction de leurs résultats pendant l'optimisation.

– *Fin de l'algorithme*: on donne ici le nombre total d'architectures à générer. A titre indicatif, nous avons fixé arbitrairement ce nombre à 1000. Ce nombre provient de différents essais que nous avons effectués et de la contrainte de temps.

Remarques :

- Ici, on répond globalement aux deux problèmes du choix de l'architecture et de la qualité de la généralisation (biais et variance).

- Le prix à payer, en terme de temps de convergence, est bien évidemment beaucoup plus élevé que pour les deux autres méthodes. D'après nos essais et pour un problème donné, le temps CPU nécessaire augmente linéairement en fonction du nombre d'observations de la base d'apprentissage et exponentiellement en fonction du nombre de neurones. Pour réduire ce temps, il est très important de paralléliser l'algorithme ce qui réduit le temps CPU d'un rapport théorique de p, où p est le nombre de processeurs utilisés. En effet, le temps d'évaluation des architectures est, de très loin, supérieur au temps nécessaire à l'algorithme génétique lui-même.

Pour les trois modèles, nous donnons enfin les valeurs des variables suivantes :

– le nombre de paramètres
– le critère $NMSE$ de la base d'apprentissage
– le critère $NMSE$ de la base de test (qui donne une mesure de la qualité de généralisation)
– Le courbe du critère $NMSE$ de la base d'apprentissage (en abscisse) en fonction du critère $NMSE$ de la base de test (en ordonnée) au cours de l'apprentissage.

8.3.1 La série LASER

La série LASER est une une série issue d'une expérimentation en laboratoire où l'on mesure, à intervalle régulier, la variation de l'intensité d'un laser (voir figure 7.11 section7.3.2). L'évolution de cette série est marquée par des oscillations qui s'amplifient suivient d'effondrements subits. On reporte le lecteur à la section 7.3.2 pour plus d'information.

La série complète comprend 2 000 points, elle est divisée en trois parties égales pour constituer les différentes bases.

Les architectures initiales

Les variables d'entrée des différents réseaux sont composées des cinq premiers retards de la série, c'est-à-dire que pour prévoir la valeur X_t, on utilise $\{X_{t-1}, X_{t-2}, \ldots, X_{t-5}\}$.

Pour l'arrêt prématuré et l'élagage, on utilise un perceptron multicouches à une seule couche cachée de 10 neurones.

Méthodes	**Early stopping**	**Élagage**	**Algorithme Genétique**
Nombre de connexions	71	47	37
$NMSE$ base d'apprentissage	$26 \quad 10^{-3}$	$9.7 \quad 10^{-3}$	$9.2 \quad 10^{-3}$
$NMSE$ base de validation	$21 \quad 10^{-3}$	$16.6 \quad 10^{-3}$	$14.2 \quad 10^{-3}$
$NMSE$ base de test	$19 \quad 10^{-3}$	$10.1 \quad 10^{-3}$	$8.4 \quad 10^{-3}$

TAB. 8.1 – *Tableau des résultats. Série laser.*

Par rapport au réseau obtenu avec la méthode de de l'arrêt prématuré, l'élagage permet d'améliorer très sensiblement les résultats pour tous les critères comparés. Néanmoins, le réseau obtenu avec l'algorithme génétique est, dans ce cas, plus performant sur l'ensemble des trois bases et possède moins de paramètres. Le fait que le critère sur la base de test est le plus bas, vient du faible nombre d'effondrements de la série qui surviennent dans cette base.

Évolution a posteriori des capacités de généralisation

Une fois les différents réseaux déterminés, on s'intéresse à l'évolution de l'erreur quadratique normalisée sur la base d'apprentissage par rapport à l'évolution de cette même grandeur sur la base de test et ceci pour les 3 réseaux.

À l'itération k de l'algorithme d'optimisation utilisé, on note θ^k le vecteur des paramètres estimé et $NMSE(\theta^k)$ la valeur du critère d'évaluation. Le graphique 8.12 se lit de la droite vers la gauche. Au cours de l'apprentissage, la valeur de $NMSE(\theta^k)$ calculé sur la base d'apprentissage décroît. On s'intéresse alors simultanément à l'évolution de $NMSE(\theta^k)$ sur la base de test.

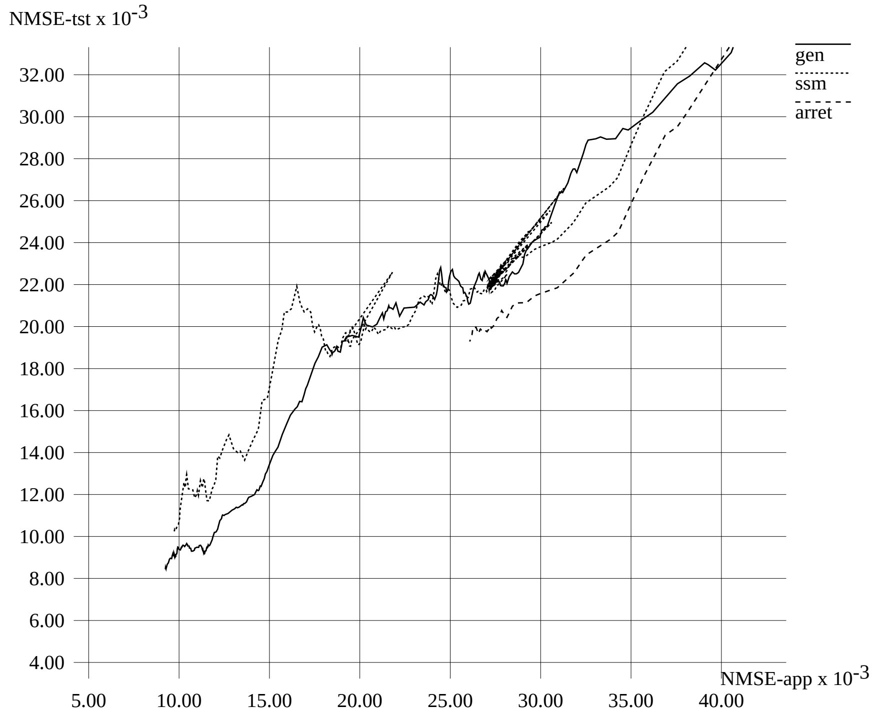


FIG. 8.12 – *Évolution des courbes $NMSE(\theta^k)$ calculé sur la base de test en fonction du $NMSE(\theta^k)$ calculé sur la base d'apprentissage (série LASER).*

Quand on utlise la méthode de l'arrêt prématuré, on voit clairement que le point d'arrêt de l'apprentissage n'est pas optimal par rapport à la base de test et donc que la qualité du résultat dépend très fortement du choix de la base de validation.

En ce qui concerne l'élagage (SSM), les résultats sont plus satisfaisants : les minima obtenus sur la base d'apprentissage et sur la base de test sont largement meilleurs à ceux obtenus pour l'arrêt prématuré. Les croissances soudaines de l'erreur quadratique normalisée sur la base d'apprentissage correspondent à l'élimination d'une connexion et donc à un début de nouvelle convergence. L'arrêt de l'apprentissage correspond à la fois au minimum sur la base d'apprentissage et sur la base de test.

L'algorithme génétique donne le meilleur résultat, qui est néanmoins assez proche de celui obtenu par l'élagage sur la base d'apprentissage. Par contre, les capacités de généralisation du réseau obtenu par l'algorithme génétique sont, dans tous les cas, plus intéressantes. Le réseau finalement obtenu est donné en figure 8.13. L'architecture obtenue est très différente de celle d'un perceptron multicouches, notamment certaines entrées ne sont utilisées qu'en phase finale du réseau (retards 1 et 3).

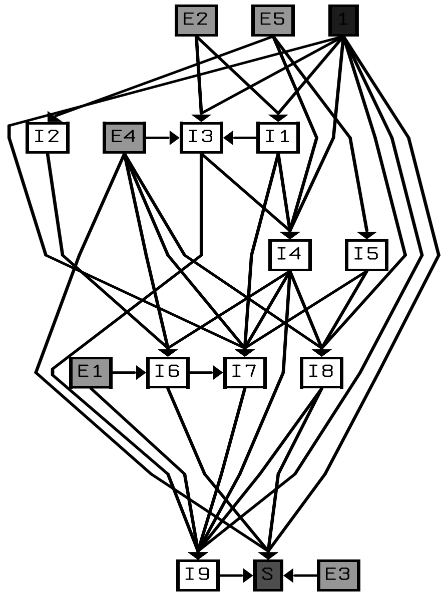


FIG. 8.13 – *Architecture du réseau final obtenu par algorithme génétique pour la série Laser.*

<u>*Conclusion*</u>

On voit, sur ce premier exemple, que le réseau obtenu par l'algorithme génétique est plus performant à la fois en matière de biais et de variance. Autrement dit le modèle proposé, décrit par l'architecture du réseau et la valeur des paramètres, est plus proche du modèle exact et inconnu régissant le processus.

8.3.2 La série SUNSPOTS

La série SUNSPOTS étudiée ici représente le nombre de tâches solaires observées annuellement. Plus de détails peuvent être trouvés section 5.3, et on peut voir son évolution depuis l'année 1700 figure 5.2. La série comporte peu de données : 228 au total (contre 2 000 pour la série LASER). Elles sont décomposées aléatoirement en trois parties comportant chacune respectivement 1/2, 1/4 et 1/4 du total.

Les architectures initiales

Les variables d'entrée des différents réseaux sont les 12 premiers retards de la série. Pour prévoir X_t, on utilise donc les variables $\{X_{t-1}, X_{t-2}, \ldots, X_{t-12}\}$.

Pour l'arrêt prématuré et l'élagage, on utilise un perceptron multicouches à une seule couche cachée de 3 neurones. Si le nombre de neurones cachés est supérieur, alors le nombre de connexions est de l'ordre de celui du nombre de données dans la base d'apprentissage et l'*overfitting* est trop important.

Tableau des résultats

Méthodes	**Early stopping**	**Élagage**	**Algorithme Génétique**
Nombre de connexions	43	15	11
NMSE base d'apprentissage	$33.9 \quad 10^{-2}$	$14.2 \quad 10^{-2}$	$19.1 \quad 10^{-2}$
NMSE base de validation	$34.1 \quad 10^{-2}$	$22.8 \quad 10^{-2}$	$18.2 \quad 10^{-2}$
NMSE base de test	$31 \quad 10^{-2}$	$21.8 \quad 10^{-2}$	$16.2 \quad 10^{-2}$

TAB. 8.2 – *Tableau des résultats. Série sunspots.*

Comme précédemment, pour la série LASER, les résultats obtenus par la méthode de l'arrêt prématuré sont moins bons sur les trois bases. En ce qui concerne l'élagage, on obtient le modèle ayant le moins de paramètres et dont les performances en apprentissage sont les meilleures. Par contre, sur les bases de validation et de test, c'est le réseau déterminé par l'algorithme génétique qui donne les meilleurs résultats. L'élagage ne supprime donc pas totalement l'*overfitting*.

Évolution a posteriori des capacités de généralisation

Les notations et le principe du graphique 8.14 sont les mêmes ceux utilisés pour le graphique 8.12. On remarque ici l'amélioration des performances du réseau issu de l'élagage et notamment le caractère itératif de recherche de la méthode (oscillations). Au contraire, la courbe relative à l'algorithme génétique est beaucoup moins heurtée, et se termine sur le minimum de l'erreur sur la base de test, ce qui n'est pas le cas pour l'élagage.

Conclusion

Les conclusions sont ici les mêmes que pour la série LASER. Le réseau obtenu par algorithme génétique est plus performant en terme de généralisation.

8.4 Fonctionnement de l'algorithme génétique

Pour mieux comprendre comment fonctionne l'algorithme génétique dans sa recherche d'une architecture performante, nous reprenons ici à titre d'exemple la série LASER.

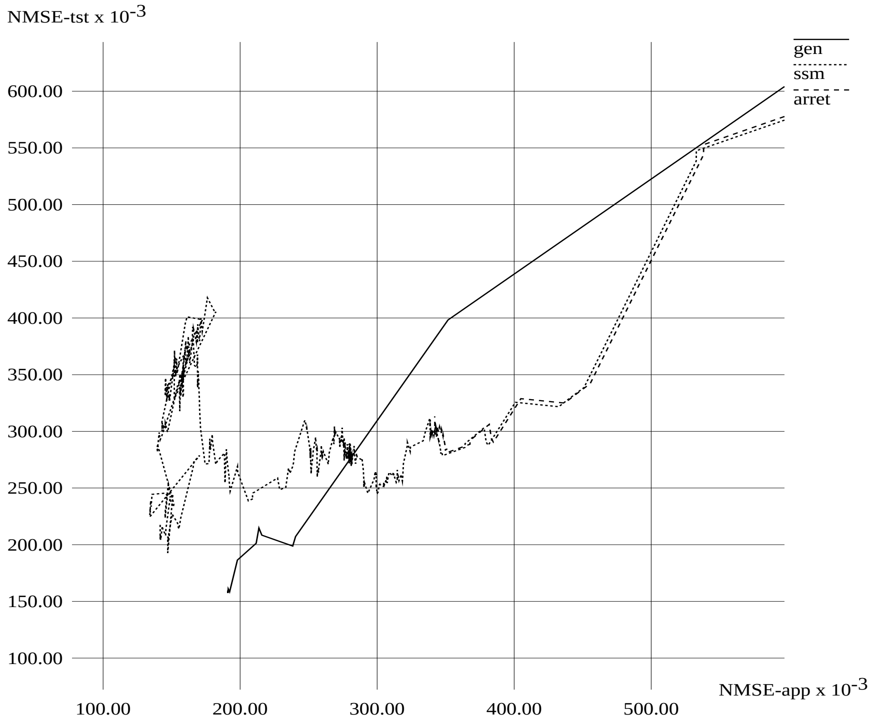


FIG. 8.14 – *Évolution des courbes $NMSE(\theta^k)$ calculé sur la base de test en fonction du $NMSE(\theta^k)$ calculé sur la base d'apprentissage (série SUNSPOTS).*

L'avantage essentiel de cet exemple est que la recherche s'effectue dans un espace relativement petit, dont les réseaux sont composés au plus de 10 neurones et de 106 connexions.

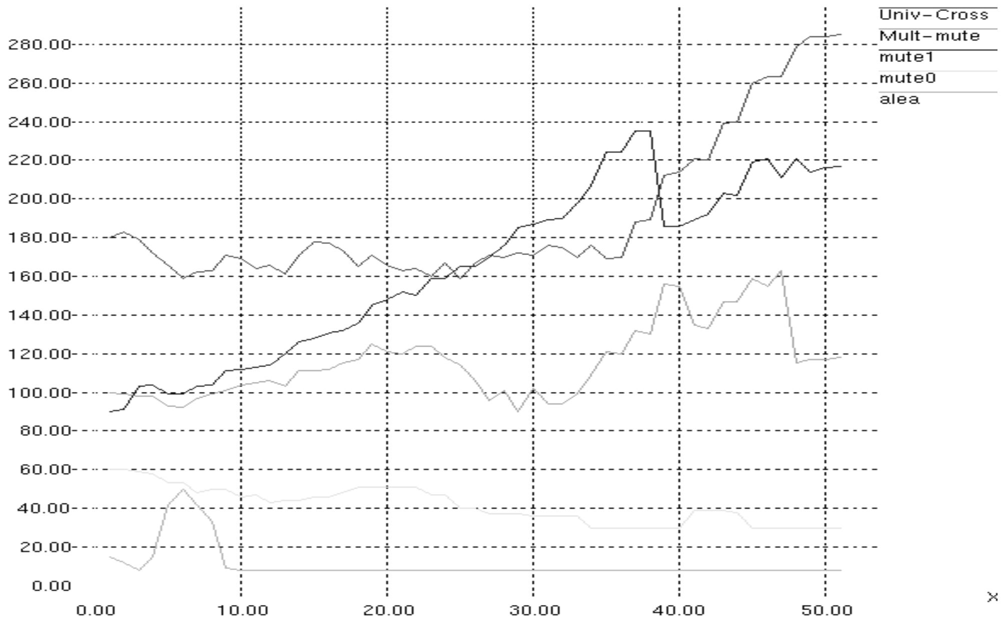


FIG. 8.15 – *Évolution des coefficients des opérateurs pendant l'estimation des paramètres.*

Dans un premier temps, étudions l'évolution des poids des différents opérateurs en analysant la figure 8.15. Les valeurs de départ ont été fixées arbitrairement en fonction de l'expérience acquise lors des précédentes simulations. Chaque itération (en abscisse) correspond à la génération de vingt nouveaux individus. Lors des premières générations, les poids varient peu, excepté le poids de l'opérateur *aléa* qui croit rapidement puis décroît. Ceci peut s'expliquer ainsi : la population initiale est peu performante, seuls les descendants générés aléatoirement ont de meilleures performances d'où l'augmentation du poids de cet opérateur. Par la suite, une fois un certain nombre d'individus intéressants aléatoirement générés, d'autres opérateurs apparaissent plus performants. D'abord la *mutation à 1*, c'est-à-dire l'ajout de connexions qui permet de densifier et de complexifier les réseaux déjà existants. Puis, le *crossing-over universel* qui permet des croisements entres des réseaux performants.

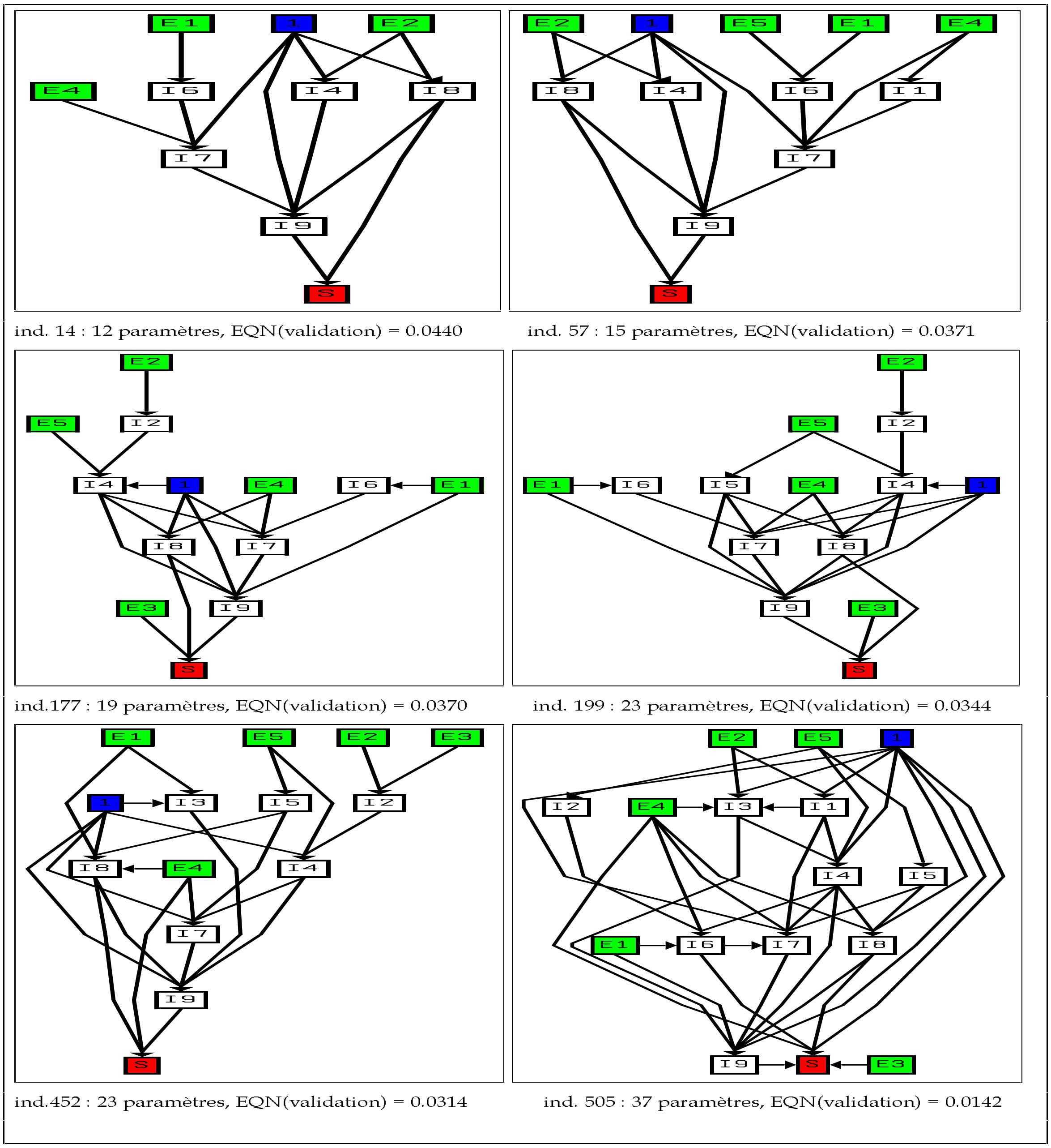
Étudions ensuite l'évolution du meilleur réseau à chaque itération. Rappelons que 1000 individus ont été générés. Nous donnons par la suite l'indice de création des réseaux. Les meilleurs d'entre eux ont été successivement les numéros : 14, 57, 177, 199, 452 et 505. Les architectures associées sont représentées dans la figure 8.16. Le premier réseau (individu 14) possède 12 paramètres et 5 neurones cachés. Son architecture est relativement simple bien que déjà éloignée de celles des perceptrons multicouches. Le second réseau (individu 57) a été trouvé après 39 générations, soit très rapidement. Il est très clairement issu du premier, trois connexions et un neurone caché ont été ajoutés. Les trois suivants ont des architectures qui vont en se complexifiant et qui sont visiblement parents les uns des autres. Le dernier est différent : sa complexité est beaucoup plus importante, comme son nombre de paramètres (passage de 23 à 37) et son nombre de neurones cachés (passage de 7 à 9). Il est issu d'un crossing-over, c'est-à-dire qu'il a été généré par le croisement de deux architectures très dissemblables mais toutes deux performantes. L'une d'entre elles était descendante de l'individu 452 (meilleur précédent), l'autre pas. Il a été trouvé à la 505 ième génération et n'a pu être amélioré pendant les 495 générations suivantes. Il est très intéressant de voir que la recherche est constructive pas à pas pour les 5 premières meilleures architectures et donc a fait beaucoup intervenir l'opérateur de mutation à 1. Par contre, la dernière architecture finalement retenue est issue d'une tout autre évolution - la fusion de deux architectures différentes.

Pour avoir une vision plus générale du fonctionnement de l'algorithme, il faudrait étudier précisément plusieurs cas de convergence. Cet exemple illustratif n'est peut-être qu'un cas particulier, il permet néanmoins de montrer que les réseaux se complexifient durant la recherche et que cette évolution peut être faite soit pas à pas grâce à la mutation à 1, soit beaucoup plus soudainement par le crossing-over.

8.5 Conclusion

Les réseaux de neurones les plus généralement utilisés pour résoudre des problèmes de régression ou de classement sont les perceptrons multicouches. Leurs architectures à couches sont simples à appréhender et leurs représentations graphiques aisées. Néanmoins, on peut nettement améliorer les performances en généralisation des perceptrons multicouches. Les expériences que nous avons menées montrent clairement que l'utilisation des réseaux à propagation directe mène à des modèles plus performants que les perceptrons multicouches.

Cependant, la recherche d'une architecture à propagation directe appropriée est complexe et ne peut être faite manuellement car l'ensemble des architectures possibles est beaucoup trop vaste. L'utilisation des algorithmes génétiques pour trouver automatiquement ces architectures performantes est tout à fait adaptée. Le codage d'une architecture en chromosome est aisé, mais les opérateurs doivent être sélectionnés judicieusement. Néanmoins la recherche par algorithme génétique permet de parcourir l'ensemble des architectures possibles et de finalement retenir une population d'architectures adaptées correspondant à différents minima locaux, parmi lesquelles peut se trouver le minimum global de la fonction. Pour autant, la dimension de l'espace de recherche ne doit pas être trop grande pour faciliter le travail de l'algorithme génétique. Le temps de convergence croit exponentiellement en fonction de la taille du chromosome et le nombre d'individus à générer doit alors augmenter en fonction de la dimension de l'espace. Ici aussi, le bon sens dicte que toutes les informations connues sur le problème



ind. 14 : 12 paramètres, EQN(validation) = 0.0440

ind. 57 : 15 paramètres, EQN(validation) = 0.0371

ind.177 : 19 paramètres, EQN(validation) = 0.0370

ind. 199 : 23 paramètres, EQN(validation) = 0.0344

ind.452 : 23 paramètres, EQN(validation) = 0.0314

ind. 505 : 37 paramètres, EQN(validation) = 0.0142

FIG. 8.16 – *Meilleurs réseaux générés par l'algorithme génétique durant la recherche (série LASER).*

à résoudre doivent permettre de réduire, autant que faire se peut l'espace de travail (par exemple la connaissance des variables corrélées au processus à modéliser).

Lorsque ces conditions sont satisfaites, on a observé sur nos simulations que le réseau le plus performant obtenu par algorithme génétique était systématiquement meilleur que ceux obtenus par des méthodes plus conventionnelles. Il est très satisfaisant de savoir qu'après une nuit de recherche automatique (à titre d'exemple), le modèle ainsi obtenu sera le plus performant.

Un des plus gros défauts de cette méthode est certainement le temps machine qu'elle demande. Ce temps machine peut aller jusqu'à une semaine de temps CPU sur une Station Sparc 20, dans le cas d'une architecture maximale comportant 200 paramètres. Néanmoins, la modélisation de processus dans le but d'effectuer de la prévision de séries temporelles n'est pas un problème à résoudre en temps réel. De plus, compte tenu de l'amélioration de la rapidité des calculateurs, le temps nécessaire à cette méthode est un inconvénient aujourd'hui, le sera-t-il encore demain ?

Un autre inconvénient de cette méthode est qu'il est évident que l'aspect boîte noire, souvent reproché aux réseaux de neurones, va ici en s'accentuant. Mais c'est peut-être à ce prix que les modèles existants peuvent être améliorés.

Dans cette étude, deux limitations ont été consciemment effectuées : le type de problème - la prévision de séries temporelles ; les modèles paramétriques utilisés - des réseaux de neurones. Elles peuvent être levées toutes les deux. Tous les problèmes généralement traités en modélisation peuvent être abordés : en régression et auto-régression (modèle explicatif, interpolation de processus, compression de données...) ; en classement et discrimination (reconnaissance de formes, séparation de sources...). Tout type de modèles paramétriques peut être utilisé comme l'ont été les réseaux de neurones, sous la condition de pouvoir effectuer un codage astucieux sous forme de chromosomes des dits modèles (ARIMA, TAR, RBF, etc, voir chapitre 2).

Les algorithmes génétiques sont des méthodes de recherche stochastiques appliquées aux problèmes d'optimisation tout comme le recuit simulé ou la recherche tabou. L'emploi de telles méthodes n'est pas immédiat, mais elles ouvrent des perspectives intéressantes pour des problèmes d'optimisation dont les fonctions de coût associés sont très complexes.

Chapitre 9

Application à la prévision de consommation électrique

Bien gérer les centrales électriques suppose d'anticiper convenablement les variations prévisibles de la demande et d'élaborer des solutions de repli en cas d'aléas défavorables. Prendre des marges de sécurité trop importantes coûte cher à l'exploitant. On cherche aujourd'hui à mettre les aléas en équations pour mieux anticiper l'avenir proche et pouvoir prendre les décisions à temps.

La consommation d'électricité suit les cycles de l'activité humaine. À l'échelle de l'année, on peut lire sur son évolution (figure 9.1) la baisse d'activité estivale et le besoin de chauffage hivernal. Si l'on observe le cycle hebdomadaire (figure 9.2, il est marqué par une demande industrielle plus faible en fin de semaine, ce cycle étant quelquefois rompu par des jours fériés ou des jours à tarification spéciale, La production électrique doit s'ajuster à la consommation à chaque instant. Pour atteindre cet objectif, l'exploitant du système électrique dispose d'un parc diversifié de centrales nucléaires, hydrauliques et à combustibles fossiles (charbon, fioul, gaz). L'énergie nucléaire, avec plus de cinquante unités de production de 900 ou 1300 MW, assure la majeure partie de la production francaise. Ces unités consomment de l'uranium enrichi. Il faut le renouveler tous les ans en moyenne, ce qui suppose l'arrêt de la centrale pendant six à huit semaines. Les quelques 600 centrales hydrauliques représentent la seconde source d'énergie électrique du pays. Certaines d'entre elles sont dites au fil de l'eau : directement installées sur le cours d'une rivière, leur production dépend de son débit et ne peut pas être différée. D'autres disposent de réservoirs de stockage (lacs de barrages et écluses) qui permettent, en fonction de leur taille, d'adapter la production électrique aux cycles de la consommation: journalier, hebdomadaire, voire annuel pour les plus grands barrages. Les centrales à combustibles fossiles assurent les besoins restants. Bien qu'elles soient assez coûteuses d'utilisation, leur présence est nécessaire pour faire face à des consommations exceptionnelles (journées très froides, etc.) ou pour préserver des ressources hydroélectriques en vue de situations ultérieures que l'on prévoit difficiles. Satisfaire la demande en électricité au meilleur coût de production : la tâche est rendue particulièrement difficile par deux facteurs. Le premier tient aux caractéristiques des centrales. Pour les grands barrages, comme pour les centrales nucléaires, la production dépend d'un stock d'énergie limité (le réservoir hydraulique ou le coeur du réacteur) qui ne peut pas être renouvelé rapidement. La seconde difficulté provient des nombreuses incertitudes auxquelles est confronté l'exploitant. Les conditions météorologiques, principalement, dans la mesure où elles influent à la fois sur le niveau de production des centrales hydrauliques et sur la consommation liée au chauffage électrique. Il faut aussi tenir compte des pannes de matériels, imprévues par définition.

La gestion des centrales est organisée selon trois niveaux, de manière à bien préserver la capacité d'anticipation et de réaction face aux incertitudes. On procède par effets de zoom, en distinguant différents horizons temporels, à savoir l'année, la semaine et la journée. On s'intéresse dans ce mémoire, à la prévision de journée. La principale motivation pour estimer la consommation électrique du lendemain concerne le choix des centrales de production à mettre en service. Elles seront mises en route à puissance maximale ou à faible puissance. Dans ce dernier cas, elles seront en réserve, prêtes à augmenter leur production en cas de hausse imprévue de la consommation ou de panne fortuite d'autres centrales. Cependant, pour faire face à d'éventuels aléas défavorables, l'exploitant peut être tenté de mettre en marche trop de moyens de production en réserve. Le coût d'une gestion excessivement prudente est celui de la mise en service de centrales superflues. L'enjeu est de taille : éviter pendant une journée d'avoir à démarrer une turbine à gaz, très rapide à mettre en service mais très chère d'utilisation, peut représenter une économie de l'ordre du million de francs. À l'horizon de la journée, il s'agit donc de minimiser le coût de satisfaction de la demande d'électricité, en tenant compte des incertitudes.

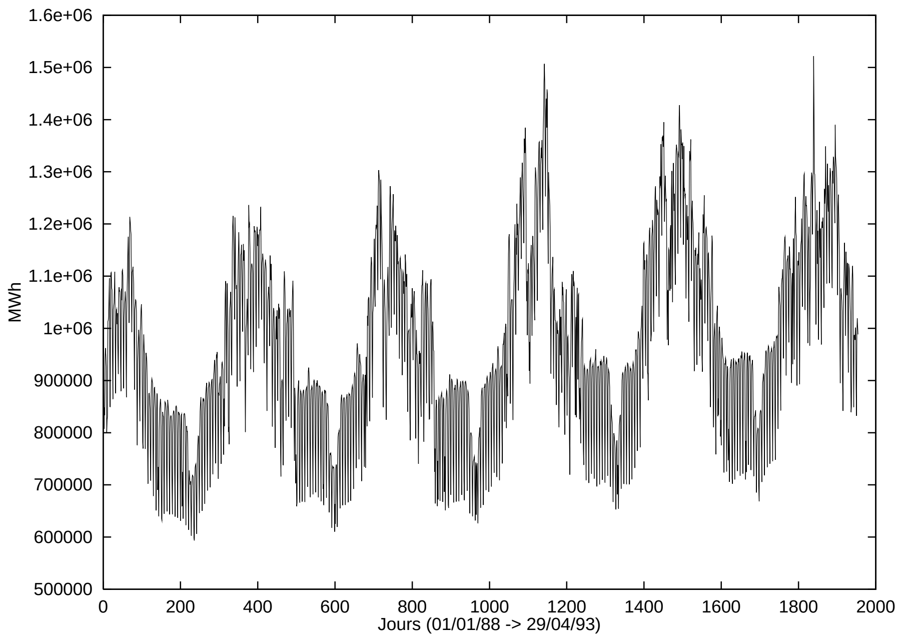


FIG. 9.1 – *Série des énergies journalières.*

La série des consommations journalières d'électricité, ou séries des énergies, présente deux périodicités d'ordre 7 et 365 (voir les courbes 9.1 et 9.2), et les valeurs de l'énergie consommée sont fortement corrélées aux valeurs de la température. D'autre part, certaines journées sont particulièrement difficiles à prévoir (voir la courbe 9.2): les jours fériés, leur veille et leur lendemain, ainsi que les jours *ejp* (effacement jour de pointe) qui correspondent à une tarification spéciale.

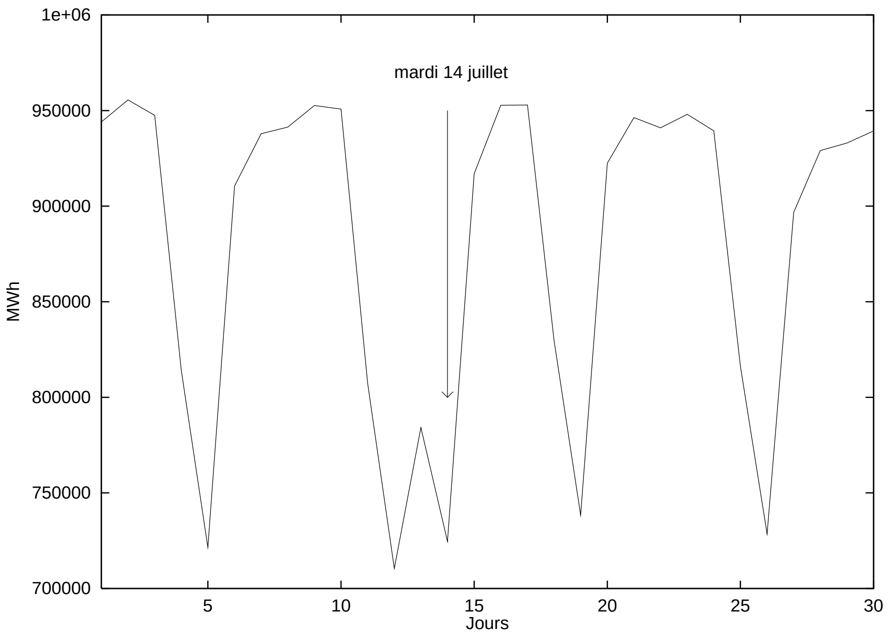


FIG. 9.2 – *Consommation d'énergie pour le mois de juillet 92. Le mardi 14 juillet brise la périodicité hebdomadaire.*

Pour bien se rendre compte de la corrélation entre consommation électrique et température, on peut se reporter à la figure 9.3. Cette courbe a un comportement inverse de la courbe 9.1, car les pics de l'une correspondent aux creux de l'autre, ce qui sous-entend une corrélation négative entre l'énergie consommée et la température moyenne de la journée considérée. Le graphe 9.4 représente l'énergie consommée en fonction de la température. La forme du nuage de points est linéaire par morceaux (l'une diagonale, l'autre commençant à peu près à 15^oC plus horizontale). Cela s'explique par le fait qu'à partir d'une certaine température le chauffage, qui constitue une grande part de la consommation électrique, n'a plus lieu d'être. Les courbes sont très différentes dans les pays où l'on utilise la climatisation l'été.

Les données dont on dispose, outre la consommation d'énergie et la température, sont la nébulosité, les dates des des jours *ejp*, et certaines caractéristiques du calendrier (jours fériés, ponts, saison, etc).

On reprend dans ce chapitre les notations utilisées dans les chapitre 5 et 6. Le critère de comparaison est la somme des erreurs quadratiques (voir équation 5.2) divisée par le nombre de jours. Pour l'ensemble des modélisations décrites dans ce chapitre, cette somme est exprimée en $(\text{MWh})^2$ et est relative à une base de test n'ayant pas servie à la

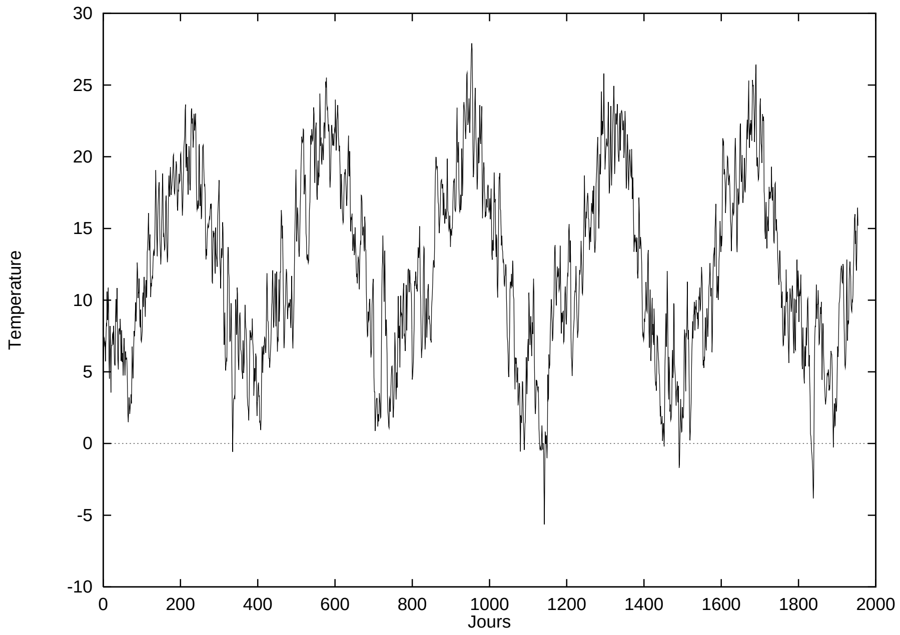


FIG. 9.3 – *Évolution de la température.*

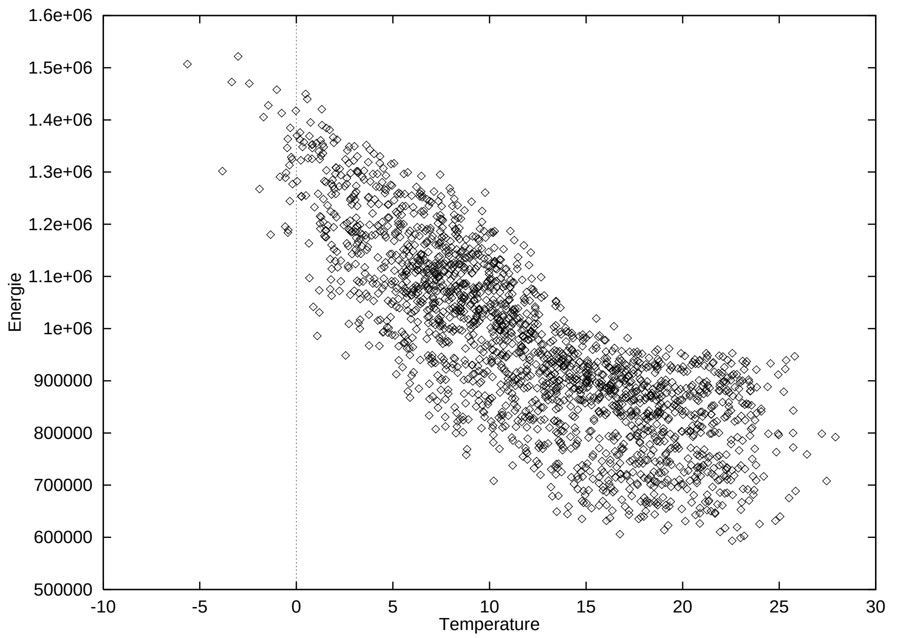


FIG. 9.4 – *Nuage de points représentant la consommation d'énergie en fonction de la température.*

modélisation. Dans la section suivante uniquement, on donne aussi les valeurs du critère BIC dont on donne la formulation équation 6.8, comme dans le cadre linéaire. Ce critère permet de comparer le modèle autorégressif avec variables explicatives (modèle ARX) trouvé avec le modèle neuronal correspondant (modèle NARX, voir équation 4.13 et figure 4.11) à la fois en terme d'erreur de prévision et en terme de nombre de paramètres utilisés.

9.1 Modélisation neuronale versus modélisation linéaire

Avant d'appliquer une méthode de modélisation neuronale, il est important, pour pouvoir effectuer des comparaisons fiables, de trouver un modèle linéaire valide. C'est pourquoi, dans un premier temps, a-t-il fallu dégager les variables pouvant améliorer l'ajustement de la série. Une modélisation ARX utilisant les variables de températures, de jours fériés et de ponts a ainsi été réalisée.

9.1.1 Modèle ARX de la série des consommations électriques

La série ayant une périodicité d'ordre 7 avec une légère tendance, il a été choisi d'opérer une différentiation d'ordre 7 et d'ordre 1.

Les 16 variables exogènes retenues sont :

- les douze moyennes journalières de températures de villes réparties sur la France
- les indicatrices des jours fériés
- les indicatrices des vendredi et des lundi de ponts
- les indicatrices des jours EJP à tarification particulière

Le meilleur modèle obtenu (avec l'aide du logiciel *MANDRAKE* (Azencott et al., 1991)) est un ARX(7), de la forme (B est le *Backshift* opérateur, $\mathrm{B}X_t = X_{t-1}$) :

$$\Delta^1\Delta^7(1-\beta_1\mathrm{B}-\beta_2\mathrm{B}^2-\cdots-\beta_7\mathrm{B}^7)(X_t-\alpha_1Y_t^1-\alpha_2Y_t^2-\cdots-\alpha_{16}Y_t^{16})=\mu+\varepsilon_t \quad (9.1)$$

où,

- Δ^1 est le filtre de différentiation d'ordre 1 ($\Delta^1 = 1-\mathrm{B}$)
- Δ^7 est le filtre de différentiation d'ordre 7 ($\Delta^7 = 1-\mathrm{B}^7$)
- $(Y_t^i)_{i=1,2,\ldots,16}$ sont les 16 variables explicatives au temps t, associées à la consommation X_t.

- $(\alpha_i, \beta_j, \mu)_{i=1,2,\ldots,16\ ;\ j=1,2,\ldots,7}$ sont 24 réels (les paramètres du modèle).
- $(\varepsilon_t)_{t\in\mathbb{Z}}$ est un bruit blanc indépendant de X_0.

La somme des erreurs quadratiques relative à ce modèle est égale à 587 10^6 $(\text{MWh})^2$ pour un nombre de paramètres égal à 29. Le BIC correspondant est égal à 20,30.

9.1.2 Modèle neuronal

Afin d'intégrer des variables exogènes en limitant le nombre de connexions synaptiques, on se ramène à un modèle neuronal calqué sur les modèles ARX en utilisant un perceptron multicouches à deux couches cachées munies de fonctions d'activations sigmoïdes. Ce modèle est noté dans la suite $\text{NARX}_n^*(p)$, lorsque la deuxième couche cachée comporte n neurones cachés et que l'on prend p retards de la série en entrée. Il est de la forme d'un modèle $\text{NARX}_n(p)$ (voir équation 4.13), à part que la partie consacrée à la « régression », qui correspond dans le modèle $\text{NARX}_n(p)$ à une combinaison linéaire des entrées et des retards, se traduit dans le modèle $\text{NARX}_n^*(p)$ par un traitement non-linéaire au niveau de la première couche cachée. Le réseau associé au modèle $\text{NARX}_n^*(p)$ comporte $p+1$ neurones sur la première couche cachée et n neurones sur la deuxième couche cachée. Ainsi si la sortie correspond au terme X_t d'une série, on connecte les variables exogènes au temps t sur le premier neurone de la première couche cachée (notons cette ensemble de connexions (a)) le terme X_{t-1} et les variables exogènes au temps $t-1$ sur le deuxième neurone de la première couche cachée (notons cette ensemble de connexions (b)) et ainsi de suite jusqu'au $p+1^{ème}$neurone. Afin de diminuer le nombre de paramètres et de rester fidèle au principe de l'ARX, on impose alors aux ensembles de connexions (a), (b),..., d'être égaux et on impose aux poids des connexions reliant le terme X_{t-i} au $i^{ème}$neurone de la première couche cachée (pour $i = 2, 3, \ldots, p+1$) d'être égaux à 1. On peut voir figure 9.5 le modèle $\text{NARX}_2^*(7)$ utilisé pour la modélisation de la série des consommations journalières d'électricité. Les variables exogènes employées sont les mêmes que celles du modèle ARX(7) décrit plus haut.

Après une initialisation aléatoire des poids synaptiques, et une fois terminé l'étape d'estimation du modèle $\text{NARX}_2^*(7)$, on obtient une variance résiduelle de 548 10^6 $(\text{MWh})^2$ pour un nombre de paramètres de 40. La valeur du BIC est de 20,29. Après un passage de l'algorithme d'élimination des poids superflus (SSM), on obtient un réseau ne comportant plus que 21 paramètres, et dont la variance résiduelle est de 565 10^6 $(\text{MWh})^2$ avec une valeur du BIC égal à 20,23.

Éléments d'analyse :

– Les retards pris en considération sont les retards d'ordre 1,2,6,7.

– Pour les connexions reliant les variables exogènes à la première couche cachée, qui sont celles reliées a sept des douze variables de température (caractérisant théo-

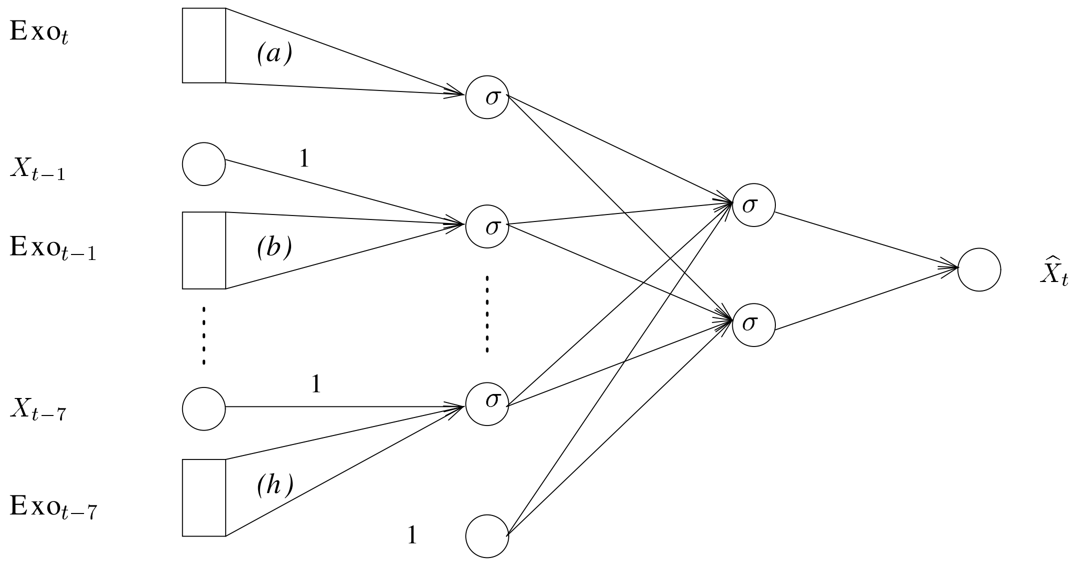


FIG. 9.5 – *Architecture du modèle NARX$_2^*$(7) (voir section 9.1.2), modèle de prévision de la série des consommations journalières d'électricité. La sortie du réseau au temps t est noté $\widehat{X}_t$, et Exo$_t$ caractérise l'ensemble des variables exogènes disponibles au temps t, et le terme (a) l'ensemble de connexions qui les relient au premier neurone de la couche cachée. On impose ici que (a)=(b)=$\cdots$=(h).*

riquement la France métropolitaine) sont éliminées. Cinq d'entre elles semblent donc suffire pour la prévision de la consommation électrique journalière.

– Toutes les connexions reliant les variables indicatrices de jours hors-normes (jour EJP, jours fériés ...) sont conservées.

– Les neurones possédant une fonctions de transfert sigmoïde (neurones des couches cachées), sont soumis à des signaux qui ne sont pas limités à un voisinage de zéro. Le réseau n'est donc pas équivalent à un modèle linéaire.

9.1.3 Comparaison

On remarque que le modèle ARX(7) obtient des performances inférieures à celles du modèle NARX$_2^*$(7) (variance résiduelle de 587 10^6 (MWh)^2 contre 565 10^6 (MWh)^2) pour un nombre de paramètres supérieur (29 paramètres contre 21), qui se traduit par une note de BIC supérieur (20,30 contre 20,23). On peut donc conclure que le modèle autorégressif neuronal, dans le cadre de la prévision de la série des énergies journalières d'électricité se comporte mieux, en terme d'ajustement et de parcimonie des paramètres, que le modèle linéaire correspondant.

Signalons que l'on arrive aux mêmes conclusions si on compare des modélisations ARMAX (Un modèle ARMA de la définition 2.3 intégrant des variables explicatives) et

NARMAX (voir équation 4.15). Notons néanmoins que la méthode de calcul des gradients (la rétropropagation du gradient (voir section 5.1.2) ne peut plus être utilisée lorsque l'on utilise en entrée du réseau des retards du bruit associé (dans la pratique on utilise les erreurs de prévision précédemment commises). On peut alors estimer les gradients par différences finies, méthode qui a l'inconvénient d'être lente et moins précise qu'un calcul exact.

9.2 Modèle modulaire neuronal

On décrit dans cette section une modélisation du type modulaire dont le principe est détaillé dans le chapitre 7. Pour effectuer la prévision de la consommation journalière d'electricité sur la France métropolitaine, on divise la tâche en deux parties distinctes. La première consiste à utiliser le modèle modulaire afin d'opérer la régression de la demande en énergie électrique sur les variables exogènes telles que la température et la nébulosité. La seconde consiste à prévoir les erreurs de cette première modélisation à l'aide d'un modèle neuronal simple. On analyse les segmentations effectuées lors de la première étape et on compare les performances de ce modèle à ceux déjà construits lors de précédentes tentatives de modélisation.

9.2.1 Architecture et apprentissage du modèle

Afin de prendre en compte directement les variables exogènes, et d'analyser quels impacts elles ont sur la consommation d'électricité, on décide de construire deux modèles distincts, appliqués de manière séquentielle Le premier (modèle (1)) utilise un modèle neuronal modulaire pour effectuer des régressions locales sur les variables exogènes et n'utilise aucune valeur passée de la série. Le deuxième (modèle (2)) tente de prévoir les résidus du premier modèle (résidus dépourvus de tendance et de périodicité qui ont déjà été pris en compte). Il apparaît qu'un modèle neuronal simple, avec une couche cachée suffit pour modéliser ces résidus. Afin de formaliser la structure de ce modèle, définissons quelques termes :

- $(X_t)_{1-p\leq t\leq T}$ représente les T valeurs de la consommation électrique,
- $(Y_t^1, Y_t^2, \ldots, Y_t^m)_{1-p\leq t\leq T}$ sont les m variables explicatives utilisées,
- f_1 est la fonction qui caractérise le modèle neuronal modulaire (modèle (1)),
- $(\varepsilon_t)_{1-p\leq t\leq T}$ sont les résidus de modèle (1),
- f_2 est la fonction qui caractérise le modèle neuronal simple (modèle (2)),
- p est le nombre de retards utilisés dans la prévision des résidus du modèle (2),
- $(\eta)_{1\leq t\leq T}$ sont les résidus du modèle (2) ; η est supposé être un bruit blanc.

Le modèle (1) s'écrit alors :

$$X_t = f_1(Y_t^1, Y_t^2, \ldots, Y_t^m) + \varepsilon_t \tag{9.2}$$

et le modèle (2) :

$$\varepsilon_t = f_2(\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-p}) + \eta_t, \quad \text{pour tout } t > 0 \tag{9.3}$$

La prédiction du modèle global s'obtient alors par :

$$X_t = f_1(Y_t^1, Y_t^2, \ldots, Y_t^m) + f_2(\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-p}) + \eta_t \tag{9.4}$$

Pour **la phase (1)** de la modélisation, chaque expert possède une couche cachée, munie de 5 neurones ; la tâche de segmentation étant ardue, la couche cachée du contrôleur est munie de 10 neurones. Experts et contrôleur reçoivent la totalité des variables à disposition.

Numéros des entrées	*Nature des entrées*
1-11	**Proximité d'un jour férié.** Indicatrices caractérisant la présence d'un jour férié. On utilise ici les informations des 5 jours précédents, du jour j et des 5 jours qui le suivent
12-27	**Classification de Kohonen.** On fournit au modèle une classification des profils de consommations journaliers issue d'un modèle de Kohonen.
28-34	**Type de jours.** 7 indicatrices caractérisent les 7 jours de la semaine.
35-41	**Proximité d'un jour EJP.** Indicatrices caractérisant la présence d'un jour EJP (Effacement Jour de Pointe, jour de « tarification spécial »). On utilise ici les informations des 3 jours précédents, du jour j et des 3 jours qui le suivent.
42-43	**Cycle annuel.** 2 variables déterminent la position du jour j dans l'année (un sinus, un cosinus, de période $365/\pi$)
44-51	**Variables climatiques.** 8 variables caractérisent la température et le niveau de nébulosité pour les jours j, $j-1$, $j-2$ et $j-5$

TAB. 9.1 – *Variables utilisées pour la modélisation neuronale modulaire de la série des consommations journalières d'électricité sur la France métropolitaine.*

Le rôle des ces entrées dans la segmentation est analysé dans la prochaine section figure 9.8.

On partage les données en deux bases :

– **une base d'apprentissage** : du 1er Janvier 1988 jusqu'au 31 Décembre 1992 (1826 jours).

– **une base de test** : du 1^{er} Janvier 1993 jusqu'au 1^{er} Mars 1994 (424 jours).

On effectue 10 apprentissages pour différents poids initiaux avec une structure comportant 8 experts. Le nombre final d'experts actifs est de 2 (3 convergences), 3 (5 convergences) et 4 (2 convergences). Ce relativement faible nombre d'experts peut être dû à la faible différence de niveau de bruit par rapport aux deux exemples détaillés précédemment. On analyse dans la section suivante le modèle comportant 4 experts actifs et donnant les meilleures performances.

Pour la **phase 2** de la modélisation, le second modèle reçoit pour entrées les résidus du premier modèle. Ceci permet de modéliser l'inertie existant dans l'évolution de la série, tâche que ne pouvait accomplir le précédent modèle dépourvu des informations sur le passé de la série. Cette modélisation en deux phases évite l'implication directe des retards de la consommation électrique, mais aussi de toutes les variables exogènes associées, implication qui entraîne une explosion du nombre des variables d'entrées. De cette manière, en traitant d'abord localement en fonction des variables exogènes, puis dans le temps, en fonction des erreurs commises, on évite cette explosion, souvent source de sur-apprentissage et de parasitage des données. On utilise dans cette phase un perceptron multicouches comportant 10 neurones d'entrée (10 retards de la série des résidus), et muni d'une couche cachée comportant 8 neurones. Pour éviter le sur-apprentissage, on utilise l'algorithme d'élimination des poids SSM décrit section 6.3, qui écarte environ 35% des poids du réseau. Les performances du réseau final, en terme d'erreurs quadratiques varient de 4% entre la base d'apprentissage et la base de test, ce qui indique qu'il n'y a pas de sur-apprentissage significatif.

9.2.2 Analyse de la modélisation

Sur la figure 9.6, on peut analyser les segmentations trouvées par le modèle modulaire en plusieurs points :

- L'expert 1 est spécialisé dans la prévision de consommation des journées proches des jours fériés
- L'expert 2 gère la prévision de consommation des jours fériés
- Les sorties du contrôleur associées aux experts 3 et 4 sont complémentaires. L'expert 3 est spécialisé dans la saison chaude (à l'exception du mois d'août) et l'expert 4 dans la saison froide (en plus du mois d'août).
- La sortie 2 du contrôleur est binaire alors que les trois autres sont complémentaires et passent de manière graduelle de l'une à l'autre.

Les valeurs finales des variances (figure 9.7) correspondent aux connaissances que l'on peut avoir sur la série des consommations. Il est en effet connu que le comportement

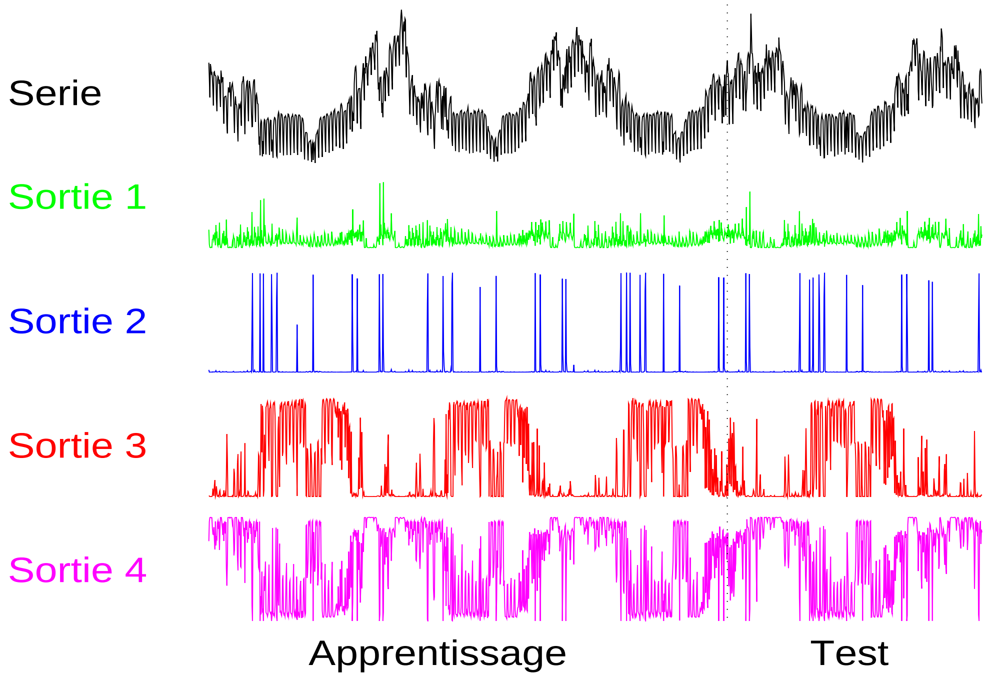


FIG. 9.6 – *Sorties du contrôleur. On remarque que la sortie associée à l'expert 2 est binaire, alors que les trois autres passent par des valeurs transitoires.*

de la clientèle est plus aléatoire les jours fériés. La prévision de la consommation n'en est alors que plus difficile. On sait aussi que certaines périodes sont plus difficiles à prévoir que d'autres. Ainsi, les périodes transitoires au sein desquelles les températures fluctuent fortement (le printemps et l'automne) sont plus difficiles à prévoir que les périodes de température stable, et les périodes de vacances sont plus difficiles à prévoir que les périodes où la plupart des gens travaillent.

On peut voir sur la figure 9.8 une analyse plus fine de ces segmentations en fonction des variables d'entrées.

On peut en particulier remarquer que le coefficient de corrélation le plus grand (0,98) correspond à celui entre l'indicatrice des jours fériés (entrée 6) et la sortie 2 du contrôleur. En fait dès que cette indicatrice est à 1, l'expert 2 s'active et s'implique totalement, alors qu'il est entièrement inhibé sinon. Les variables d'entrées 44 à 51, correspondant aux variables climatiques sont corrélées positivement avec l'expert 3 et négativement avec celui de l'expert 4, expliquant leur rôle dans la prévision des saisons chaudes/froides. Les variables d'entrées 28 à 34 (correspondant aux indicatrices des jours de la semaine) sont corrélées avec l'activation des experts 1, 3 et 4 mais très peu avec l'expert 2. Ceci est cohérent avec le fait que les types de jour n'ont que peu d'importance dans la prévision d'un jour férié (que ce soit un dimanche ou un mardi, l'activité sociale et économique est la même, et la consommation électrique varie peu entre ces deux cas).

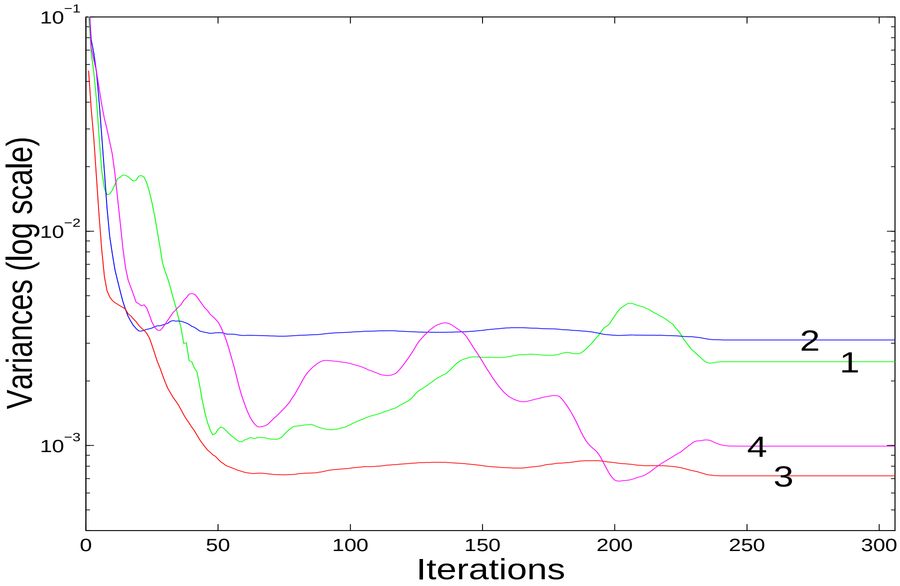


FIG. 9.7 – *Évolution des variances associées aux experts. A la fin de la convergence, celle associée à l'expert 2, gérant la prévision pour les jours fériés, est la plus grande.*

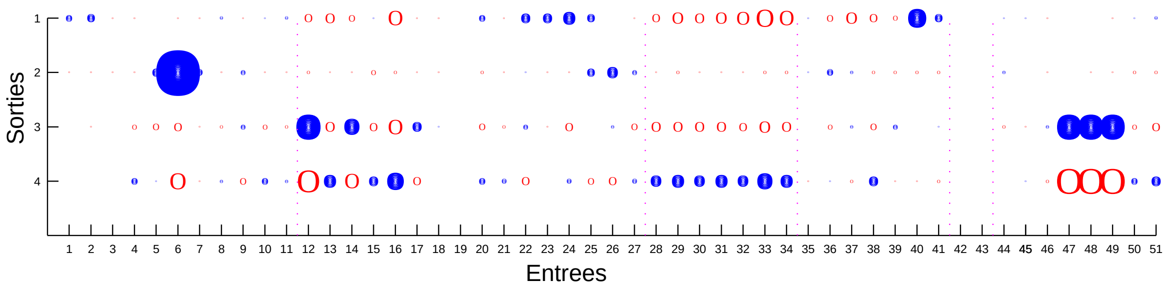


FIG. 9.8 – *Corrélations linéaires entre les variables d'entrées et les sorties du contrôleur. Les cercles pleins caractérisent des corrélations positives alors que les cercles simples caractérisent des corrélations negatives. La taille de ces cercles indique la valeur absolue de la corrélation. Ainsi, la corrélation entre l'entrée 6 et la 1ère sortie est de 0,98 alors que celle entre l'entrée 47 et la sortie 3 est de 0,54. Les indices des entrées correspondent à ceux indiqués dans le texte précédemment.*

9.3 Recherche stochastique de structures

On utilise dans cette section, la méthodologie exposée dans le chapitre 8. En particulier, on utilise des réseaux de neurones sans notion de couches de la définition 8.1 dont on peut voir une représentation graphique figure 8.1. Pour satisfaire à la méthodologie de recherche stochastique de structure par algorithmes génétiques, on divise la base en trois parties. La base d'apprentissage comprend la moitié des données tirées aléatoirement, les bases de validation et de test sont constituées de chacune un quart.

Les variables d'entrée sont à la fois composées de retards sur la série des énergies et sur d'autres variables exogènes :

- les températures ;
- les nébulosités ;
- les indicatrices des jours de la semaine ;
- les ponts des lundis et vendredis ;
- les indicatrices des mois de juillet et août ;
- les indicatrices des vacances de Noël ;
- les indicatrices des jours ejp et jours fériés ;

soit au total 60 variables d'entrée décrites de manière plus précise en annexe.

Le réseau à propagation directe sans notion de couches utilisé comme réseau dominant comprend 8 neurones cachés. L'algorithme génétique effectue donc sa recherche dans un espace dont les réseaux ont potentiellement 576 connexions.

Le réseau final comporte 114 connexions et la somme des erreurs quadratiques sur la base de test vaut 470 10^6 $(\text{MWh})^2$. On peut en voir figure 9.9 une représentation graphique. Quelques particularités peuvent être relevées comme la connexion directe (donc linéaire) entre la consommation du jour $j-1$ et la sortie, ou le regoupement des variables température du jour $j-1$, nébulosité du jour $j-1$ et température du jour j indiquant, comme on pouvait le supposer, de fortes corrélations entre ces variables. Il est cependant évident (voir figure 9.9) que l'aspect boîte noire souvent associé aux réseaux de neurones est encore plus accentué dans le réseau trouvé.

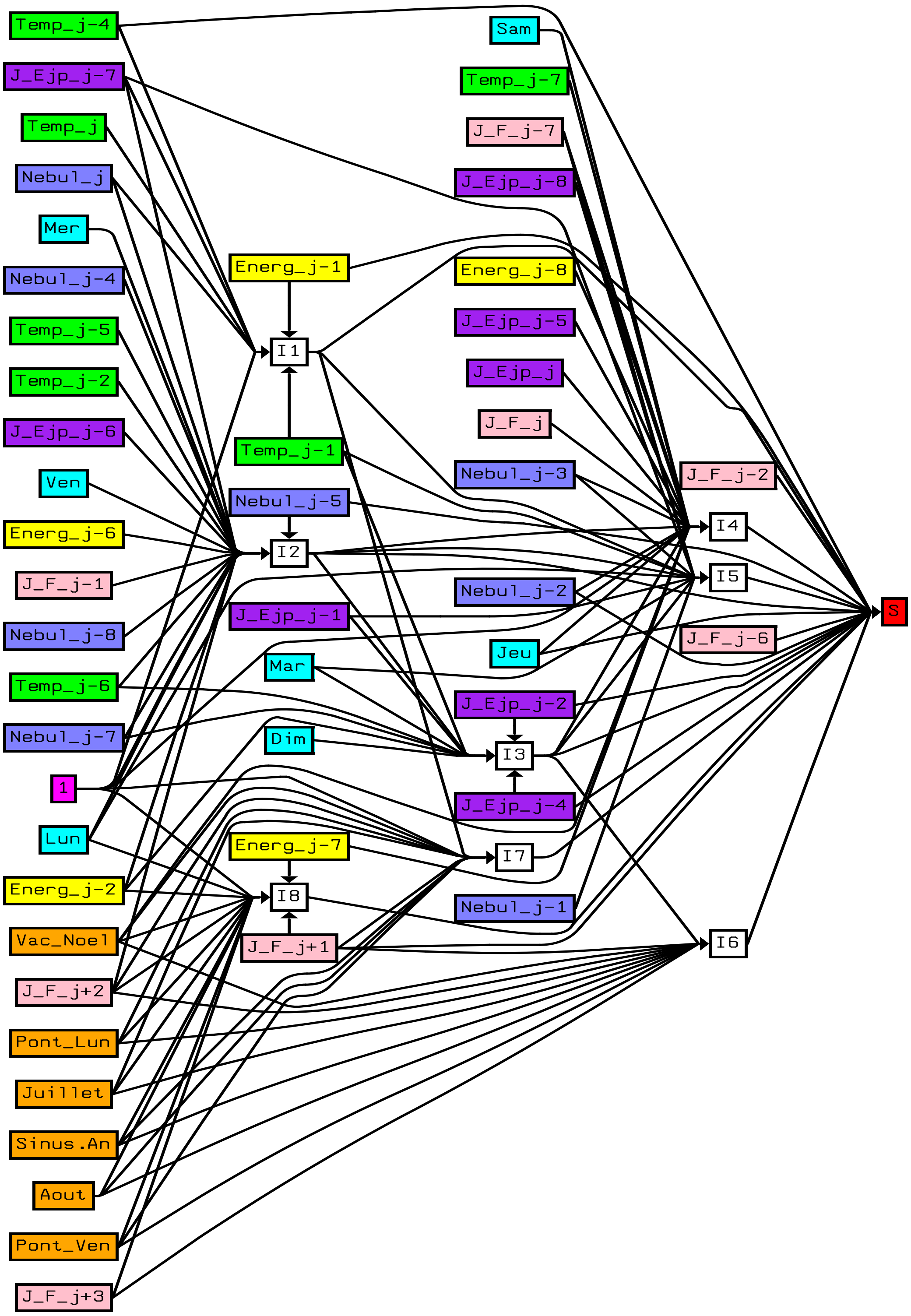


FIG. 9.9 – *Architecture résultante de la recherche par algorithmes génétiques d'un modèle neuronal de prévision de la série des consommations électriques journalière sur la France métropolitaine.*

9.4 Comparaison et Analyses

Plusieurs avantages sont apparus lors des tentatives de modélisation neuronale de la série des consommations journalières d'électricité :

- des performances pouvant être supérieures aux modèles linéaires classiques sur une base d'exemples n'ayant pas servi à la modélisation (en particulier les modèles issus de la recherche par algorithme génétique),
- des possibilités de fournir des intervalles de confiance précis pour les prévisions (modèles neuronaux modulaires),
- une interprétation des relations entre la consommation électrique et les variables exogènes (analyse des segmentations issues du modèle neuronale modulaire).

	Moyenne des erreurs quadratiques $(MWh)^2)$
ARX(7) (voir section 9.1.1)	$587\ 10^6$
$NARX_2^*(7)$ (voir section 9.1.2)	$548\ 10^6$
modélisation modulaire (voir section 9.2)	$524\ 10^6$
Recherche stochastique d'architecture (voir section 9.3)	$470\ 10^6$

TAB. 9.2 – *Comparaisons des performances des différentes modélisations de prévision de la série des consommations journalières d'électricité (calculs sur une base de test).*

Néanmoins, si la comparaison entre les modèles neuronaux et les modèles classiques linéaires est plutôt en faveur des réseaux de neurones en ce qui concerne la série des énergies journalières (voir tableau 9.2), certains problèmes se sont avérés difficiles à résoudre. En particulier, choisir la bonne structure parmi la grande variété de modéles possibles semble finalement être la tâche la plus ardue pour exploiter réellement le potentiel important des réseaux de neurones. Il apparaît ainsi indispensable d'adopter une démarche rigoureuse et circonspecte dans la mise au point de tels modèles.

Chapitre 10

Conclusion et perspectives

Ce mémoire est organisé autour d'une idée directrice, qui est la prévision de séries temporelles par des modèles paramétriques non-linéaires et en particulier par des modèles de réseaux de neurones à propagation directe. On précise en particulier les conditions assurant certaines propriétés asymptotiques utiles dans les étapes d'estimation et d'identification des modèles autorégressifs paramétriques non-linéaires. En particulier, on rappelle les notions de stabilité, d'ergodicité et de mélangeance des modèles autorégressifs fonctionnels, et on met en évidence un certain nombre de conditions assurant la consistance et la normalité asymptotique de l'estimateur des moindres carrés des paramètres. Ces résultats permettent alors de construire une méthodologie d'identification de modèle, un *stepwise* descendant que l'on nomme *SSM*, basé sur des tests successifs de nullité des paramètres.

Dans ce document, on s'intéresse aux réseaux de neurones à propagation directe qui sont des modèles non-linéaires paramétriques particuliers. Ces modèles, en raison de leur capacité d'approximation universelle, s'avèrent performants dans de nombreuses applications, mais leur analyse mathématique recèle encore aujourd'hui de nombreux points obscurs. Les résultats théoriques cités précédemment permettent alors de répondre à un certain nombre de questions, et permettent d'utiliser des techniques aidant à construire des architectures adéquates pour un problème de prévision donné. Toujours afin de faciliter les étapes d'estimation et d'identification, on propose aussi une technique de recherche stochastique d'architecture pour des modèles neuronaux à propagation directe sans notion de couches. La recherche s'opère à l'aide des algorithmes génétiques au sein d'un espace de modèles déterminé, assez large pour contenir l'architecture recherchée.

Dans ce document, on décrit aussi un modèle neuronal modulaire, permettant de modéliser des séries temporelles stationnaires par morceaux. Dans l'approche que l'on a choisie, on utilise des réseaux du type perceptron multicouches pour représenter à la fois les sous-modèles et le modèle de partitionnement. Ceci permet de prendre en considération une segmentation non-linéaire de la série et des distributions non-linéaires des données au sein de chaque régime trouvé.

L'ensemble des méthodes exposées est évalué à travers des applications sur des séries de laboratoire ou simulées sur ordinateurs. On détaille et on compare aussi l'application de ces méthodes sur un problème de prévision concret, concernant la prévision de la consommation journalière d'électricité sur la France métropolitaine.

À travers ces travaux, plusieurs aspects inhérents à la prévision de séries temporelles par modèle neuronaux sont apparus :

1. Il est important de constamment maîtriser la complexité inhérente aux modèles neuronaux.
2. Il est aisé d'obtenir des performances proches des modèles linéaires. Lorsque les variables à prendre en considération possèdent des corrélations non-linéaires, on peut obtenir de meilleurs résultats, mais la structure de modèle utilisé permet de faciliter ou de complexifier les étapes d'estimation et d'identification.
3. Si la plupart des modèles neuronaux ne permettent pas une analyse a posteriori de la série (distribution des variables, calcul des corrélations, ...), les modèles neuronaux modulaires permettent au contraire de dégager certaines caractéristiques, en particulier lorsqu'elles sont liées à différents régimes de la série.

Il apparaît donc indispensable d'adopter une démarche circonspecte, dès lors que l'on tente d'opérer un modélisation non-linéaire.

Il est a noter que la plupart des idées et méthodes décrites dans ce mémoire, notamment la méthode d'élimination des paramètres non-significatifs, peuvent s'appliquer à de nombreux autres modèles non-linéaires. Ces méthodes peuvent aussi le plus souvent être adaptées à des domaines autres que la prévision de séries temporelles, comme la régression mais aussi la classification et la reconnaissance de forme. On peut aussi faire un certain nombre de remarques, sur le contenu de ce mémoire et sur des orientations possibles de recherches.

En ce qui concerne les techniques de modélisation de prévision en général :

- Des techniques d'optimisation de fonctions paramétrées qui convergent vers un minimum global commencent à être étudiées (voir section 5.2.1). Dans le cadre de la prévision utilisant des modèles non-linéaires (par exemple les réseaux de neurones), ces techniques pourraient s'avérer très utiles.
- Le problème de la prévision à un horizon supérieur à 1 reste un problème ouvert dans le cas d'une modélisation non-linéaire. Pourtant, la plupart des problèmes que l'on rencontre dans la vie courante nécessite une prévision à plusieurs pas. Maîtriser cette prévision permettrait d'éviter les multiples tentatives de modélisation, dûs à l'ignorance de l'évolution de la prévision.

En ce qui concerne les conditions théoriques assurant la consistance et la normalité de l'estimateur des moindres carrés des paramètres d'un modèle autorégressif non-linéaire (chapitre 3) :

- Dans ce document, on étudie essentiellement le cas des modèles autorégressifs. Obtenir des résultats de convergence similaires à ceux décrits chapitre 3, en prenant en considération des modèles non-linéaires autorégressifs avec moyennes mobiles (utilisation de l'erreur des précédentes prévisions) peut être un objectif à court terme.
- Les modèles paramétriques polynomiaux du type *bilinéaire* ne satisfont pas les conditions relatives aux résultats de convergence. Il serait alors intéressant de tenter d'utiliser certains résultats récents assurant la stabilité des modèles polynomiaux, et de trouver un ensemble de conditions assurant la consistance et la normalité asymptotique de l'estimateur des moindre carrés des paramètres.

En rapport avec la prévision de séries stationnaires par morceaux à l'aide des modèles neuronaux modulaires (chapitre 7) :

- Dans le cadre des processus non-linéaires, la non-stationnarité peut prendre de multiples formes : par exemple la stationnarité par morceaux, la non-stationnarité sur certains intervalles, etc. Il semble donc intéressant de redéfinir ou d'affiner le critère de stationnarité.

En ce qui concerne la recherche stochastique de structures de modèle (chapitre 8) :

- On a décrit dans ce mémoire une méthode basée sur les *algorithmes génétiques*. On peut alors se demander si d'autres algorithmes de recherche peuvent être plus efficaces ou plus rapides. En particulier, une comparaison est en cours entre la méthode utilisée dans ce mémoire et une méthode basée sur un *recuit simulé*.

Bibliographie

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Auto. Control*, 19:716–723.

Andel, J. et Barton, T. (1986). A note on Threshold AR(1) model with Cauchy innovations. *J.T.S.A.*, 7:1–5.

Attali, J. et Pagès, G. (1995a). Approximation of functions by perceptrons: a new approach. *Neural Processing Letters*, 22(5):19.

Attali, J. et 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., et Martin, M. (1991). *MANDRAKE: un progiciel expert en analyse de séries temporelles*. Harcourt Brace and World, Paris.

Barron, A. (1993). Universal Approximation Bounds for Superpositions of a Sigmoidal Function. *IEEE Transactions on Information Theory*, 39(3).

Bayomog, S., Guyon, X., Hardouin, C., et f. Yao, J. (1996). Test de différence de contrastes et somme pondérée de khi-deux. *Canadian J. Statist.*, 24.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 21:307–328.

Bollerslev, T., Chou, R. Y., Jayaraman, N., et Kroner, K. F. (1990). ARCH modeling in finance: A review of the theory and empirical evidence. *Journal of Econometrics*, 52(1):5–60.

Box, G. E. P. et Jenkins, F. M. (1976). *Time Series Analysis: Forecasting and Control*. Prentice Hall, Englewood Cliffs, NJ.

Breidt, F., Davis, R., Lii, K., et Rosenblatt, M. (1990). Maximum likelihood for non causal autoregressive processes. *Preprint*.

Breush, T. et Pagan, A. (1978). A simple test for heteroskedasticity and random coeffient variation. *Econometrica*, 46:1287–1294.

Brockwell, P. et Davis, R. (1991). *Time Series: Theory and Methods*. Springer-Verlag, Berlin.

Buntine, W. L. et Weigend, A. S. (1994). Computing second derivatives in feed-forward networks: A review. *IEEE Transactions on Neural Networks*, 5:480–488.

Casdagli, M. C. (1989). Nonlinear Prediction of Chaotic Time Series. *Physica*, D.35:335–356.

Casdagli, M. C. (1991). Chaos and deterministic versus stochastic nonlinear modeling. *Journal Roy. Stat. Soc. B*, 54:303–328.

Chan, K. et Tong, H. (1986). On estimating thresholds in Autoregressive models. *J.T.S.A.*, 7:179–190.

Chatfield, C. (1989). *The Analysis of Time Series*. Chapman and Hall, London.

Cottrell, M., Girard, B., Girard, Y., Mangeas, M., et Muller, C. (1995). Neural modeling for time series: a statistical stepwise method for weight elimination. *IEEE Transaction on Neural Networks*, 6:1355–1364.

Cullogh, W. M. et Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5.

Cybenko, G. (1989). Approximation by Superposition of Sigmoidal Functions. *Mathematics of Control, Signals, and Systems*, 2:303–314.

Dacunha-Castelle, D. et Duflo, M. (1993). *Probabilité et statistiques*. Springer-Verlag, 2nd edition.

Davis, L. (1991). *Hanbook of Genetic Algorithms*. Van Nostrand Reinhold.

Diebolt, J. et Guégan, D. (1991). Probabilistic properties of the general nonlinear markovian process of order one and applications to time series modelling. Technical report, Technique L.S.T.A. Paris VI.

Doukhan, P. (1994a). *Mixing: Properties and Examples*, volume 85 of *L.N.S.* Springer-Verlag.

Doukhan, P. (1994b). *Mixing: properties and examples*. Springer-Verlag L.N.S.

Doukhan, P. et Ghindes, M. (1992). *Étude des processus* $X_n = f(X_{n-1}) + \varepsilon_n$. PhD thesis, Thèse de 3^{e} cycle - Université Paris XI.

Doukhan, P. et Tsybakov, A. (1993). Non parametric recursive estimation in nonlinear ARX-models. *Problems of Information Transmission*, 29(4):318–327.

Doutriaux, A. et Zipser, D. (1990). Unsupervised discovery of speech segments using recurrent networks. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., et Hinton, G. E., editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 303–309, San Fransisco, CA. Morgan Kaufmann.

Draper, N. et Smith, H. (1981). *Applied Regression Analysis*. John Wiley & Sons, New York.

Duflo, M. (1990). *Méthodes récursives aléatoires*. Masson, Paris.

Duflo, M. (1996). *Algorithmes Stochastiques*. Mathématiques & Applications (SMAI). Springer-Verlag, Berlin.

Duflo, M., Senoussi, R., et Touati, A. (1990). Sur la loi des grands nombres pour les martingales vectorielles et l'estimateur des moindres carrés d'un modèle de regression. *Ann. I.H.P.*, 26:549–566.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Engle, R. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50:987–1007.

Fambon, O. et Jutten, C. (1994). A comparison of two weight pruning methods. In *ESANN'94*.

Farber, J. (1982). Chaotic attractors of an infinite dimensional dynamical system. *Pysica*, D.4:366–393.

Farber, J. (1987). Predicting chaos time series. *Pysical Review Letters*, 59:845–848.

Farmer, J. D. et Sidorowich, J. J. (1988). Exploiting chaos to predict the future and reduce noise. In Lee, Y. C., editor, *Evolution, Learning, and Cognition*, Singapore. World Scientific.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, 19:1–142.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192.

Geman, S., Bienenstock, E., et Doursat, R. (1992). Neural networks and the bias / variance dilemma. *Neural Computation*, 5:1–58.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley publishing company.

Gorodkin, J., Hanses, L., Krogh, A., Svarer, C., et Winther, O. (1993). A quantitative study of pruning by optimal brain damage. *Int. J. of Neural Systems*, 4(2):159–169.

Gourieroux, C. et Monfort, A. (1990). *Séries temporelles et modèles dynamiques*. Economica, Paris.

Granger, C. et Andersen, A. (1978). *An introduction to bilinear time series analysis*. Gottingen, Vandenhoeck and Ruprecht.

Granger, C. W. J. (1994). Forecasting in economics. In Weigend, A. S. et Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 529–538, Reading, MA. Addison-Wesley.

Guégan, D. (1994). *Séries chronologiques non-linéaire à temps discret*, volume Statistique mathématiques et Probabilité. Economica.

Guégan, D. et Diebolt, J. (1994). Probabilistic properties of β-ARCH model. *Statisca Sinica*.

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.

Hall, P. et Heyde, C. (1980). *Martingale Limit Theory and Its Applications*. Academic Press, Inc., San Diego.

Hamilton, J. D. (1990). Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45:39–79.

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press, Princeton.

Hannan, E. et Kavalieris, L. (1986). Regression, autoregression models. *J. Time Series Analysis*, 7:27–49.

Hertz, J., Krogh, A., et Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, MA.

Hopfield, J. J. (1982). Neural Networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:3088–3092.

Hornik, K., Stinchcombe, M., et White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366.

Ito, Y. (1991). Representation of Function by Superpositions of a step or sigmoid Function and their Applications to Neural Networks Theory. *Neural Networks*, 4:385–394.

Ito, Y. (1992). Approximation of Continuous Function on $\mathbf{r}^d$ by linear Combinations of Shifted Rotations of a Sigmoid Function With and Without Scaling. *Neural Networks*, 5(1):105–115.

Ivanova, T. O., Mottle, V. V., et Muchnik, I. B. (1994). Estimation of the parameters of hidden markov models of noise-like signals with abruptly changing probabilistic properties (part i). *Automation and Remote Control*, 55:1299–1315.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., et Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

Jones, D. (1978). Non-linear autoregressive processes. *Roy.. Soc. London*, A.360:71–95.

Jordan, M. I. et Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214.

Jordan, M. I. et Xu, L. (1995). Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, (in press).

Jutten, C. et Chentouf, R. (1995). A new scheme for incremental learning. *Neural Processing Letters*, 2(1):1–4.

Kheradmania, M. et Tong, H. (1990). A bayesian approach to threshold autoregressive modelling. Technical report, University of Kent.

Klimbo, L. et Nelson, P. (1978). On conditional least squares estimation for stochastic processes. *Ann. Statist.*, 6:629–642.

Kohonen, T. (1982). Self-organazied formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

Lai, T. (1994). Asymptotic properties of nonlinear least squares estimates in stochastic regression models. *Ann. Statist.*, 22:1917–1930.

Lai, T. et 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.

Lapedes, A. et Farber, R. (1987). Nonlinear signal processing using neural networks. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau à seuil assymétrique. *Cognitiva 85*, pages 599–604.

LeCun, Y. (1989). Generalization and network design strategies. In Pfeifer, R., Schreter, Z., Fogelman, F., et Steels, L., editors, *Connectionism in Perspective*, Amsterdam. North Holland.

LeCun, Y., Denker, J. S., et Solla, S. A. (1990). Optimal brain damage. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2 (NIPS*89)*, pages 598–605, San Mateo, CA. Morgan Kaufmann.

Leshno, M., Lin, V. Y., Pinkus, A., et Schoken, S. (1993). Multilayer feedforward networks with a nonpolunomial activation function can approximate any function. *Neural Networks*, 6:861–867.

Levin, E. (1991). Modeling time varying systems using hidden control neural architecture. In Lippmann, R. P., Moody, J. E., et Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3 (NIPS*90)*, pages 147–154. Morgan Kaufmann.

Lewis, P. A. W., Ray, B. K., et Stevens, J. G. (1994). Modeling time series using multivariate adaptive regression splines (MARS). In Weigend, A. S. et Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 296–318, Reading, MA. Addison-Wesley.

Lii, K. et Rosenblatt, M. (1982). Deconvolution and estimation of transfer function phase and coefficients for nongaussien linear processes. *Annals of Statistics*, 10:1195–1208.

Liu, J. (1988). On the general bilinear time series models. *Journal of Applied Probability*, 25:553–564.

MacKay, D. (1992a). Bayesian interpolation. *Neural Computation*, 4:415–447.

MacKay, D. (1992b). The evidence framework applied to classification networks. *Neural Computation*, 4:720–736.

MacKay, D. (1992c). Information-based objective functions for active data selection. *Neural Computation*, 4:590–604.

MacKay, D. (1992d). A practical bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472.

MacKay, D. (1993). Bayesian non-linear modeling for the 1993 energy prediction competition. In Heidbreder, G., editor, *Maximum Entropy and Bayesian Methods*, Santa-Barbara.

Mangeas, M., Cottrell, M., Girard, Y., Girard, B., et Muller, C. (1993). Advantages of the multilayer perceptron for modeling and forecasting time series: application to the daily electrical consumption in france. In *Proceedings of Neuronîmes'93*, Nîmes, France.

Mangeas, M. et Yao, J. (1996). Sur l'estimateur des moindres carrés d'un modèle autorégressif non-linéaire. Technical Report 53, SAMOS, Université Paris I.

Marquardt, D. (1963). Optimization method. *J. Soc. Ind Appl. Math*, 11:431–441.

Meyn, S. et Tweedie, R. (1993). *Markov Chains and Stochastic Stability*. Springer-Verlag, Berlin.

Minoux, M. (1983). *Programmation Mathématiques, Théorie des Algorithme*, volume 2. Dunod, Paris.

Minsky, M. et Papert, S. (1969). *Perceptrons*. MIT Press, Cambridge (E.U.).

Mokkadem, A. (1987a). Sur un modèle autorégressif non linéaire: ergodicité et er godicité géométrique. *J. Time Series Analysis*, 8(2):195–204.

Mokkadem, A. (1987b). Sur un modèle autorégressif non linéaire: ergodicité et ergodicité géométrique. *Journal of time series analysis*, 8:195–204.

Moody, J. et Darken, C. (1988). Learning with Localized Receptive Fields. In Touretzsky, D., Hinton, G., et Sejnowski, T., editors, *Connectionist Models Summer School*, pages 133–143, San mateo: Morgan Kaufmann.

Moody, J. et Darken, C. (1989). Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, 1:281–294.

Mozer, M. et Smolensky, P. (1989). Using relevance to reduce network size automatically. *Connection Science*, 1:3–16.

Muller, C., Cottrell, M., Girard, Y., Girard, B., et Mangeas, M. (1994). A neural network tool for forecasting French electricity consumption. In *Proceedings of WCNN'94*, San Diego, California, USA.

Müller, K.-R., Kohlmorgen, J., et Pawelzik, K. (1994). Segmentation and identification of switching dynamics with competing neural networks. In *Proceedings of International Conference on Neural Information Processing (ICONIP'94)*, pages 213–218.

Nowlan, S. J. et Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493.

Perrone, M. P. (1994). General averaging results for complex optimization. In Mozer, M. C., Smolensky, P., Touretzky, D. S., Elman, J. L., et Weigend, A. S., editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 364–371, Hillsdale, NJ. Lawrence Erlbaum Associates.

Pham, T. (1986). The mixing property of bilinear and genralized random coefficient autoregressive model. *Stochastic Processes and their Applications*, 23:291–300.

Powell, M. J. D. (1987). Radial basis functions for multivariate interpolation: A review. In *IMA Conference on Algorithms for the apporximation of Functions and Data.*, volume Shrivenham: RMCS. J. c. Mason and M. G. Cox.

Press, W. H., Flannery, B. P., Teukolsky, S. A., et Vetterling, W. T. (1988). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge U.K.

Press, W. H., Flannery, B. P., Teukolsky, S. A., et Vetterling, W. T. (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Reed, R. (1993). Pruning algorithms - a survey. *IEEE Trans. on Neural Networks*, 4(5):740–747.

Robinson, P. (1977). The estimation of a non linear moving average models. *Proc. and their applications*, 1:81–90.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, New York.

Roynette, B. (1993). Vitesse d'approximation d'une fonction par un réseau de neurones. *Pré-publication de l'université Nancy 1.*

Rumelhart, D. E., Hinton, G. E., et Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. et McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume I: Foundations, pages 318–362. MIT Press/Bradford Books, Cambridge, MA.

Seber, G. A. F. et Wild, C. J. (1989). *Nonlinear Regression*. Wiley, New York.

Senoussi, R. (1990). Statistique asymptotique presque sûre de modèles convexes. *Ann. I.H.P. (Probabilités et Statistiques)*, 26:19–44.

Tong, H. (1990). *Non-linear Time Series: a Dynamical System Approach*. Oxford University Press.

Tong, H. et Lim, K. S. (1980). Threshold autoregression, limit cycles and cyclical data. *J. Roy. Stat. Soc. B*, 42:245–292.

Tsay, R. (1987). Conditional heteroskedasticity time series analysis. *J.A.S.A.*, 82:590–604.

Vapnik, V. (1982). *Estimation of dependences based on empirical data*. Springer-Verlag, Berlin.

Vapnik, V. (1992). Principles of risk minimization for learning theory. *Advances in Neural Information Processing Systems*, pages 831–838.

Waterhouse, S. R. et Robinson, A. J. (1995). Non-linear prediction of acoustic vectors using hierarchical mixture of epxerts. In Tesauro, G., Touretzky, D. S., et Leen, T. K., editors, *Advances in Neural Information Processing Systems 7 (NIPS*94)*. MIT Press, Cambridge, MA.

Weigend, A. S., Huberman, B. A., et Rumelhart, D. E. (1990). Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1:193–209.

Weigend, A. S. et Nix, D. A. (1994). Predictions with confidence intervals (local error bars). In *Proceedings of the International Conference on Neural Information Processing (ICONIP'94)*, pages 1207–1212, Seoul, Korea.

White, H. (1992). *Artificial Neural Networks, Approximation and Learning Theory*. Blackwell.

Xu, L. (1994). Signal segmentation by finite mixture model and EM algorithm. In *Proceedings of the 1994 International Symposium on Artificial Neural Networks (ISANN'94)*, pages 453–458, Tainan, Taiwan.

Table des matières détaillée

Table des figures

Liste des tableaux

Annexe A

Articles

Annexe B

Liste des communications

Cottrell, M., Girard, B., Girard, Y., Mangeas, M., et Muller, C. (1995). Neural modeling for time series: a statistical stepwise method for weight elimination. *IEEE Transaction on Neural Networks*, 6:1355–1364.

Mangeas, M. (1992). Rapports n^{o}1 du cerd soad/samos. Technical report, EDF/DER.

Mangeas, M. (1993a). Rapports n^{o}2 du cerd soad/samos. Technical report, EDF/DER.

Mangeas, M. (1993b). Rapports n^{o}3 du cerd soad/samos. Technical report, EDF/DER.

Mangeas, M., Cottrell, M., Girard, Y., Girard, B., et Muller, C. (1993). Advantages of the multilayer perceptron for modeling and forecasting time series: application to the daily electrical consumption in france. In *Proceedings of Neuronîmes'93*, Nîmes, France.

Mangeas, M. et Muller, C. (1996). How to find suitable parametric models using genetic algorithms, application to feedforward neural networks. In symposium on computaio-nal Statistics, X., editor, *COMPSTAT'96*, Barcelone, Espagne.

Mangeas, M., Muller, C., et Weigend, A. S. (1995). Forecasting electricity demand using a mixture of nonlinear experts. In *World Congress on Neural Networks (WCNN'95)*, pages II–48–53.

Mangeas, M. et Weigend, A. S. (1995). First experiments using a mixture of nonlinear experts for time se ries analysis. In *World Congress on Neural Networks (WCNN'95)*, pages II–104–109.

Muller, C., Cottrell, M., Girard, Y., Girard, B., et Mangeas, M. (1994). A neural network tool for forecasting French electricity consumption. In *Proceedings of WCNN'94*, San Diego, California, USA.

Weigend, A. S. et Mangeas, M. (1995a). Analysis and prediction of multi-stationary time series using nonlinear gated experts. Technical Report CU-CS-764-95, University of Colorado at Boulder, ftp://ftp.cs.colorado.edu/pub/Time-Series/MyPapers/experts.ps.

Weigend, A. S. et Mangeas, M. (1995b). Avoiding overfitting by locally matching the noise level of the data. In *World Congress on Neural Networks (WCNN'95)*, pages II–1–9.