République Algérienne Démocratique et Populaire

Ministère de L'enseignement Supérieur et de la Recherche Scientifique

Université Frères Mentouri, Constantine1

Faculté des Sciences Exactes

Département de Mathématiques

N° d'ordre : 49 / DS / 2022

N° de série : 04 / Math / 2022

Thèse

Présentée pour l'obtention du diplôme de doctorat en sciences

Spécialité : Mathématiques Appliquées

Option : Statistiques Appliquées

Régression non Paramétrique et Sélection de Modèles par le Moyen des Réseaux de Neurones Artificiels

Présenté par :

Mlle. Faiza Saadi

Devant le jury :

Président	M^{r} A. Hameida	Prof	Université des frères Mentouri, Constantine 1.
Encadreur	M^{me} S. Kharfouchi	Prof	Université Salah Boubnider, Constantine 3.
Examinateur	M^{r} M. Boukeloua	M.C.A	École polytechnique, Constantine.
Examinatrice	M^{me} S. Leulmi	M.C.A	Université des frères Mentouri, Constantine 1.

Soutenue le : 21 / 06 / 2022.

Dédicaces

À ma très chère mère Sassia et à mon très cher père Ahcene.

Je ne vous remercierai jamais et je ne pourrai jamais assez vous remercier pour votre amour, votre soutien, votre gentillesse et vos conseils.

***Merci** aussi à toute ma famille, Rima, Amina, fairouz, Ibrahim, Nouhene, Chahd, ...,*

, à tous mes amis, Chahrazed, Ilhem, Mouna, ...,

à tous mes collègues, et à tous ceux que j'aime et qui m'aiment.

À mon professeur A. Chibat

Je suis très fier de vous offrir la primeur de mes efforts et de mes sentiments pour exprimer mon respect envers vous. Vous avez constitué toujours un point de repère et un guide pour moi. Je suis très honoré de vous offrir ce travail. Merci pour vos encouragements et vos soutiens. Un grand merci pour l'aide et le soutient que vous m'avez apporté. C'est l'occasion aussi de vous exprimer ma plus grande estime et ma gratitude. Que Dieu vous fasse miséricorde et vous fasse entrer au Paradis, Inshaalah.

Remerciement

Merci et grand crédit à Dieu pour son aide et son succès pour que nous achevions ce travail au maximum. Nous le remercions beaucoup et nous le louons pour toutes les bénédictions et la miséricorde qu'il nous a confiées sans pouvoir ni force.

J'aimerais tout d'abord remercier mon directeur de thèse Mme S. Kahfouchi pour son aide, ces conseils et sa disponibilité. Soyez assuré de ma profonde gratitude.

Je remercie également Mr A. Hameida d'être le président du jury de ma soutenance. Merci pour ça compréhension et son soutien. Veuillez croire à l'expression de ma profonde reconnaissance et de mon grand respect.

Je tien à remercier aussi Mr M. Boukeloua et Mme S. Leulmi d'être les examinateurs de cette thèse et pour l'honneur qu'ils m'a fait en acceptant d'être membres de mon jury de thèse. Je tiens à l'assurer de ma profonde reconnaissance pour l'intérêt qu'il porte à ce travail.

TABLE DES MATIÈRES

INTRODUCTION

Les réseaux de neurones artificiels (RNA) sont des modèles statistiques qui ont été développés dans la perspective algorithmique de l'apprentissage automatique, ou anciennement appelé intelligence artificielle (IA). Les différentes techniques d'IA ont montré leur puissance dans le calcul non linéaire et ont été largement appliquées dans la plupart des domaines scientifiques.

L'objectif principal de l'IA est de simuler des comportements du cerveau humain. Les premières tentatives de modélisation du cerveau sont anciennes et précèdent même l'ère informatique, (McCulloch et Pitts 1943). Cette approche dite connexionniste a atteint ses limites technologiques, compte tenu de la puissance de calcul de l'époque, mais aussi théoriques au début des années 70, [34].

Parmi diverses techniques d'IA, les réseau de neurones artificiels peuvent apprendre et représenter directement n'importe quel phénomène complexe sous-adjacent aux données mesurées, et stocke les connaissances dans des neurones computationnels et poids de connexion. A cause de leur capacité d'approximation universelle et de leur mécanisme de traitement de l'information parallèle, les RNA ont été mis en œuvre avec succès dans la prédiction, [2].

Les réels développements des réseaux de neurones sont de nature purement mathématique et statistique. Leurs applications se situent dans des domaines qui n'ont généralement aucun rapport avec la neurobiologie, [11]. Un réseau de neurones est souvent considéré comme une boîte noire magique, malgré qu'il existe un modèle de probabilité défini derrière lui. Les RNA sont en effet un modèle statistique pour effectuer une

régression ou une classification non-paramétrique, [34]. Les procédures statistiques non-paramétriques sont largement utilisées en raison de leur simplicité, de leur applicabilité sous des hypothèses assez générales et de leur robustesse aux valeurs aberrantes dans les données. Ce sont donc des outils statistiques populaires dans de diverses disciplines.

Les réseaux de neurones constituent actuellement le sujet de beaucoup de recherches, en raison de leurs propriétés intéressantes d'apprentissage de modèles non-paramétriques et leurs possibilités d'application à des problèmes de classification et de prédiction. Les réseaux de neurones ont été généralement utilisés comme outil de modélisation à cause de leurs potentiel à apporter de nouvelles approches et à améliorer la précision de la prédiction. Les réseaux de neurones représentent une approche non-paramétrique qui ne nécessite pas une connaissance a priori de la distribution statistique des données.

Le choix des poids initiaux des couches cachées d'un réseau est un point fondamental de la phase d'apprentissage pour avoir un réseau convergent dans un délai raisonnable. Au début, les chercheurs ont convenu que les valeurs initiales peuvent être générées par une loi uniforme sur un petit intervalle centré sur zéro, [1], [46]. La proximité de cette valeur à zéro semble se traduire par de meilleurs taux de convergence, [46], [68]. Sans information préalable, il n'y a aucune raison de privilégier certaines valeurs par rapport à d'autres. Cependant, ce mode de sélection ne conduit pas nécessairement à des situations propices à une bonne entraînement du réseau. Les valeurs finales ciblées des paramètres peuvent être très éloignées des valeurs initiales aléatoires, [33], [46].

Le choix des réseaux de neurones comme outil d'une régression non-paramétrique est notre intérêt dans cette thèse dont nous essayons, dans le premier Chapitre, de ressortir le maximum d'informations sur les RNA. Une description plus détaillée sur les différents méthodes d'approximation non-paramétrique est exposée dans le Chapitre 2, en mettant le point sur le fait que les RNA sont des méthodes statistiques non-paramétriques. Les concepts théoriques et pratiques de la méthodologie de la construction de modèles statistiques et de la régression multi-variées par le moyen des RNA sont abordés dans le Chapitre 3. Le quatrième Chapitre est consacré aux différentes méthodes d'initialisation des poids, en se basant sur la méthode de Nguyen-Widrow. Le Chapitre cinq constitue le corps de la méthode proposé dans ce travail de thèse et qui représente une amélioration de la méthode d'initialisation de Nguyen-Widrow.

CHAPITRE 1

RÉSEAUX DE NEURONES ARTIFICIELS

1.1 Préambule

Les réseaux de neurones artificiels sont des inventions pratiques apparus de façon massive depuis des années 1980. Ils peuvent être perçus comme une classe de modèles mathématiques inspirés du fonctionnement des neurones biologiques du cerveau humain, et qui par la suite sont approché par des méthodes statistiques. Ils connaissent depuis quelques années un succès croissant dans de divers domaines de la science, grâce à leur traitement informatique et à leur mécanisme. Ils ont aujourd'hui un impact considérable et leur importance s'affirme avec le temps.

Dans ce chapitre, nous nous intéressons au fonctionnement des réseaux de neurones comme outil d'analyse d'une régression non paramétrique. Dans un premier temps, nous donnons un historique concernant l'évolution des réseaux de neurones. Nous rappellerons, ultérieurement, les définitions et les fondements relatifs aux réseaux de neurones. En suite nous exposerons les différentes architectures des réseaux de neurones et les types et la méthodologie d'apprentissage.

1.2 Historique des réseaux de neurones

Le champ des réseaux de neurones a démarré en 1943 par la proposition du neurone formel de Warren McCulloch et Walter Pitts, qui est une abstraction du neurone physiolo-

gique. Il s'agit d'un neurone binaire. McCulloch et Pitts montrèrent théoriquement que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes,[61].

En 1949, Donald Hebb présente, dans son ouvrage "The Organization of Behavior", les premières règles d'apprentissage par les réseaux de neurones, dont de nombreux modèles de réseaux aujourd'hui s'en inspirent.

En 1958, Frank Rosenblatt développe le modèle du Perceptron. C'est le premier système artificiel capable d'apprendre par expérience. Il possède deux couches de neurones : une couche de Perception qui sert à recueillir les entrées, et une couche de décision. C'est le premier modèle pour lequel un processus d'apprentissage a pu être défini.

Dans la même période, le modèle de l'Adaline (Adaptive Linear Element) a été présenté par Bernard Widrow et Ted Hoff. Ce modèle sera par la suite le modèle de base des réseaux de neurones multicouches.

En 1969, Marvin Minsky et Seymour Papert publient une critique des propriétés du Perceptron. Ils démontrent les limites théoriques du Perceptron. Notamment, l'impossibilité de traiter les problèmes non linéaires, ce qui va avoir une grande incidence sur la recherche dans le domaine des sciences cognitives et de l'intelligence artificielle. Cette recherche a été stoppé dans son élan jusqu'en 1972, où Teuvo Kohonen présente ses travaux sur les mémoires associatives et propose des applications a la reconnaissance de formes.

En 1982, John Hopfield a remis au monde le connexionnisme en donnant une architecture solide inspirée de la physique. Il développe un modèle basé sur la règle de Hebb pour définir les notions d'attracteurs et de mémoire associative.

En 1984, Kohonen désigne le concept des cartes auto-adaptatives fondés sur un algorithme non supervisé. Elles sont utilisées pour cartographier un espace de données réelles à grande dimension.

Une année plus tard (1985), la machine de Boltzmann a été inventé par Geoffrey Hinton et Terry Sejnowski. C'est un processus d'apprentissage stochastiques ayant une structure récurrente et est à la base des premières techniques d'optimisation utilisé dans les RNA.

En 1986, David E. Rumelhart, Geoffrey E. Hintont et Ronald J. Williams publient l'algorithme de la rétro-propagation du gradient de l'erreur qui permet d'optimiser les

paramètres d'un Perceptron multicouches. Il permet la possibilité de réaliser une fonction non linéaire d'entrée/sortie sur un réseau en décomposant cette fonction en une suite d'étapes linéairement séparables.

Aujourd'hui, l'utilisation des réseaux de neurones dans de divers domaines ne cesse pas de croître, à cause de leurs propriétés en particulier, leurs capacités d'apprentissage, et leur habilité à être des systèmes dynamiques.

1.3 Fondements des Réseaux de Neurones

Un réseau de neurones est un ensemble de neurones formels ou nœuds interconnectés permettant la résolution de problèmes complexes grâce à l'ajustement des coefficients de pondération dans une phase d'apprentissage. Les nœuds sont analogue aux neurones biologiques et sont issus d'une tentative de conception d'un modèle mathématique très simplifié du cerveau humain.

1.3.1 Neurone biologique

Un neurone biologique est une cellule vivante spécialisée dans le traitement des signaux électriques. C'est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites, où l'information est acheminée de l'extérieur vers le soma, qui est une cellule d'activité nerveuse au centre du neurone, [3].

L'information traitée par le neurone est cheminée ensuite le long de l'axone, attaché au somma, pour être transmise aux autres neurones indirectement. En fait, il existe un espace intercellulaire de quelques dizaines d'Angströms, (unité de longueur utilisée pour mesurer les distances très petites, 1Å$= 10^{-10}$ mètres), entre l'axone du neurone affèrant et les dendrites du neurone efférant, [61].

La jonction entre deux neurones est appelée synapse. Les neurones font une sommation des signaux reçus en entrée, et en fonction du résultat obtenu vont fournir un courant en sortie.

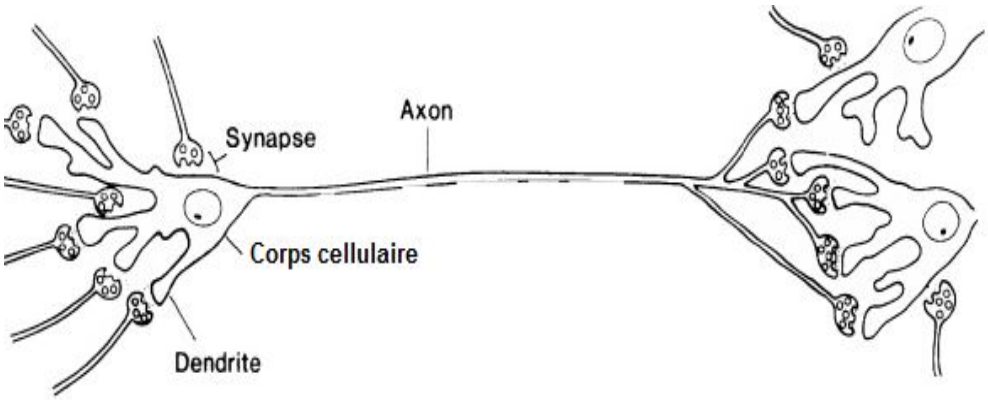


FIGURE 1.1 – Neurone biologique

1.3.2 Neurone artificiel

Lors de l'émergence d'une nouvelle technique dans le domaine d'informatique, les ingénieurs commencent à construire un neurone appelé neurone artificiel qui est une fonction algébrique non linéaire, à valeurs bornées. Chaque élément d'un neurone artificiel a une correspondance parmi les éléments de composition d'un neurone biologique. Un neurone artificiel reçoit un nombre de variables d'entrées en provenance de neurones amont. À chacune de ces entrées est associé un poids. Chaque processeur élémentaire est doté d'une sortie unique.

La Figure 1.2 donne une représentation schématique d'un neurone artificiel, désigné par les formules de l'équation 1.1.

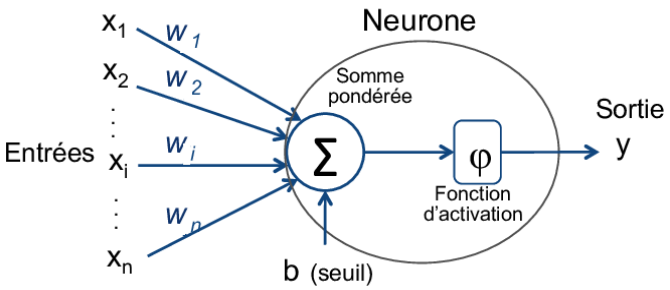


FIGURE 1.2 – Neurone artificiel

$$\begin{cases} u = \sum_{i=1}^{n} w_i \, x_i \\ \text{et} \\ y = \varphi(u - b) \end{cases} \tag{1.1}$$

où : $x_1, x_2, \ldots, x_n$ sont les entrées ; $w_1, w_2, \ldots, w_n$ sont les poids synaptiques du neurone ; u est la sortie de l'unité de sommation ; b est le seuil ; $\varphi(.)$ est la fonction d'activation et y est le signal de sortie du neurone.

Toutes les entrées n'ont pas une importance égale pour le neurone. Certaines d'entre elles sont plus importantes que d'autres. Le poids w affecté à chaque connexion mesure cette importance relative. La somme pondérée u est appelée activation. Le neurone effectue ensuite une opération qui dépend de u. Cela revient à dire qu'il applique une fonction φ à la valeur u. Cette fonction est appelée fonction d'activation ou fonction de transfert.

1.3.3 Fonctions d'activations

La fonction d'activation joue un rôle très important dans le comportement du neurone. Elle peut être une fonction à seuil, une fonction linéaire ou une fonction non linéaire.

La fonction seuil est la fonction de transfert dans le neurone de Mcculloch et Pitts (1943). Elle transforme les signaux d'activation positifs en signaux unité et les signaux d'activation négatifs en signaux nuls. La discontinuité se produit à la valeur d'activation nulle, qui est égal au "seuil" de la fonction signal.

La sortie d'un neurone fournie par une fonction d'activation linéaire est une somme linéaire des entrées pondérées, et peut prendre un nombre infini de valeurs.

Plusieurs types de fonctions non linéaires sont couramment utilisées comme fonction d'activation. Ce sont essentiellement les fonctions sigmoïdes et les fonctions gaussiennes.

Les fonctions sigmoïdes sont une famille de fonctions en forme de S dont la plus utilisée est la fonction logistique ou la fonction tangente hyperbolique. Elles ont l'avantage d'être régulières, monotones, continûment dérivables, et bornées, [10].

La fonction gaussienne est une fonction en exponentielle de l'opposé du carré de l'abscisse. Elle a une forme caractéristique de courbe en cloche.

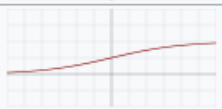
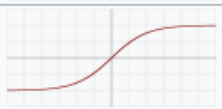
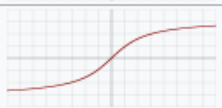
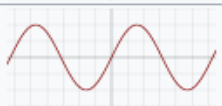
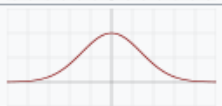
Nom	Graphe	Équation
Identité/Rampe		$f(x) = x$
Marche/Heaviside		$f(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x \geq 0 \end{cases}$
Logistique (ou marche douce, ou sigmoïde)		$f(x) = \frac{1}{1 + \mathrm{e}^{-x}}$
Tangente hyperbolique		$f(x) = \tanh(x) = \frac{2}{1 + \mathrm{e}^{-2x}} - 1$
Arc tangente		$f(x) = \tan^{-1}(x)$
Unité de rectification linéaire douce (SoftPlus)		$f(x) = \ln(1 + \mathrm{e}^{x})$
Sinusoïde		$f(x) = \sin(x)$
Fonction gaussienne		$f(x) = \mathrm{e}^{-x^2}$

FIGURE 1.3 – Fonctions d'activations usuelles

1.4 Architecture des réseaux de neurones

Un réseau de neurones est un mélange de plusieurs neurones, réalisant des traitements élémentaires, généralement organisé en couches. Les neurones d'une même couche sont tous connectés à l'ensemble des entrées, qui reçoivent les données du problème et chacune de ces connexions a une association d'un poids w. Cela donne une couche totalement connectée. Les couches qui suivent la première ont comme entrée la sortie de la couche précédente. La dernière couche est nommée couche de sortie. Les couches qui précédent la couche de sortie sont nommées couches cachées et définissent l'activité interne du réseau. En général, les fonctions d'activation sont non linaires sur ces couches, [50].

Pour spécifier la structure du réseau, il faut également choisir le nombre de couches, le nombre de neurones sur chaque couche et le nombre d'entrées du réseau. De même, le nombre de neurones sur la couche de sortie et les fonctions de transfert sont fixés d'après la nature du problème que l'on veut résoudre avec ce réseau. Par exemple, si l'on désire produire des sorties binaires (0 ou 1), alors on choisira éventuellement une fonction seuil pour la couche de sortie. Il reste ensuite a choisir le nombre de couches cachées ainsi que le nombre de neurones sur ces couches et leur fonction de transfert. Il faudra aussi fixer les différents paramètres de l'algorithme d'apprentissage.

Les réseaux multicouches sont beaucoup plus puissants que les réseaux a une seule couche (réseaux simples). Un réseau a une couche cachée, avec un nombre optimal de neurones, et une couche de sortie, en tenant une fonction d'activation sigmoïde sur la couche cachée, peut produire une approximation de la plupart des fonctions, avec une précision arbitraire. Les réseaux de neurones ont la capacité de représenter n'importe quelle fonction, linéaire ou pas, simple ou complexe. Ils permettent l'optimisation en temps réel des prévisions.

Il y a deux types de réseaux de neurones, en fonction du graphe de leurs connexions : réseaux bouclés (dynamiques) et réseaux non-bouclés (statiques où à propagation avant).

1.4.1 Réseaux de neurones à propagation avant

Un réseau de neurones non-bouclé réalise une (ou plusieurs) fonction algébrique de ses entrées, par composition des fonctions réalisées par chacun des neurones. Dans un tel

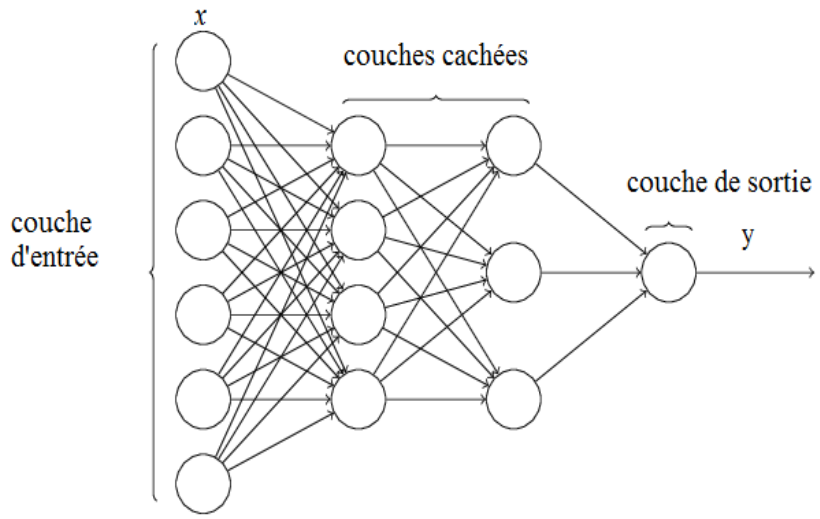


FIGURE 1.4 – Réseau de neurones à propagation avant

réseau, le flux d'information propage des entrées vers les sorties, (vers l'avant), sans retour en arrière. Si l'on représente le réseau comme un graphe dont les nœuds sont les neurones et les connexions entre ceux-ci, le graphe d'un réseau non bouclé est acyclique. Les neurones qui effectuent le dernier calcul de la composition de fonctions sont les neurones de sortie. Ceux qui effectuent des calculs intermédiaires sont les neurones cachés.

Dans un réseau de neurones à propagation avant (Feeedforward), le temps ne joue aucun rôle fonctionnel : si les entrées sont constantes, les sorties le sont également. Le temps nécessaire pour un fonctionnement de chaque neurone est négligeable et on peut le considérer comme instantané. Pour cette raison, ce type de réseaux sont souvent appelés réseaux statiques, par opposition aux réseaux bouclés (dynamiques). Les réseaux à propagation avant sont utilisés en classification, reconnaissance des formes (caractères, parole, ...), et en prédiction.

Un réseau de neurone à propagation avant est appelé aussi Perceptron (Frank Rosenblatt en 1957) qui est le premier système artificiel capable d'apprendre par expérience. On peut distinguer dans cette catégorie de réseaux, deux structures qui se diffèrent par le nombre de couches dans le réseau :

Le réseau monocouche (Perceptron simple) qui ne dispose que de deux couches : la couche d'entrée et la couche de sortie. Les neurones organisés en entrée sont entièrement connectés aux neurones de sortie par une seule matrice de poids, ce qui limite le réseau à un classificateur linéaire permettant de diviser l'ensemble d'informations obtenues en

deux catégories distinctes.

Il y a, également, le réseau multicouche, [48] dont les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche, mais chaque neurone d'une couche est connecté à tous les neurones de la couche suivante. Un Perceptron multicouche est, de ce fait, mieux adapté pour traiter les types de fonctions non-linéaires.

1.4.2 Réseaux de neurones bouclés

L'architecture la plus générale, pour un réseau de neurones, est celle des réseaux bouclés, dont le graphe des connexions est cyclique. Lorsque l'on se déplace dans un réseau bouclé en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ (cycle). Les chemins cycliques ramènent l'information en arrière par rapport au sens de la propagation du réseau.

La sortie d'un neurone dans un réseau bouclé peut donc être fonction d'elle même. Cela n'est certainement possible que si la notion de temps est explicitement prise en considération. En conséquence, à chaque connexion cyclique d'un réseau de neurones bouclé est introduit un multiple entier (éventuellement nul) de l'unité de temps choisie, appelé retard. Une grandeur, à un instant donné, ne peut pas être fonction de sa propre valeur au même instant, tout cycle du graphe du réseau doit avoir un retard non nul, [43]. Ces réseaux sont décrits par le système d'équations aux différences.

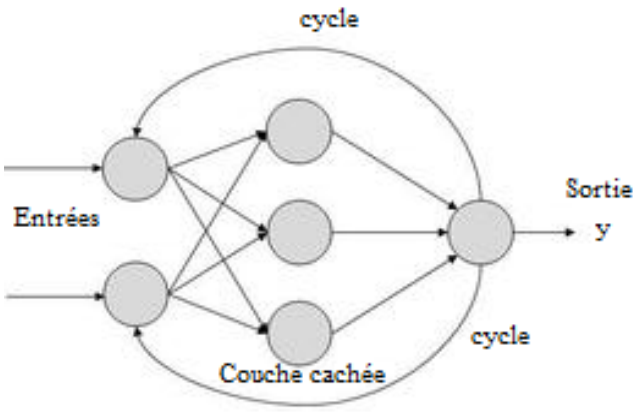


FIGURE 1.5 – Réseau de neurones bouclé

1.5 Apprentissage des réseaux de neurones

La propriété qui est d'une importance primaire pour un réseau de neurones est sa capacité d'apprendre de son environnement, et d'améliorer ses performances par l'apprentissage. C'est un problème numérique d'optimisation qui calcule les pondérations optimales des poids au fur et à mesure des itérations pendant lesquelles on présente la totalité des exemples, afin de minimiser l'écart entre les sorties calculées et les sorties expérimentales, [43].

L'apprentissage d'un réseau de neurones, sur un même échantillon, peut devenir inapte de reconnaître la vrai relation entre les données d'expérience. Le réseau ne cherche plus l'allure générale de la relation mais cherche à reproduire les allures de l'échantillon. Alors en parle ici du sur-apprentissage, dont le réseau est devenu trop spécialisé et ne généralise plus correctement. Ce phénomène apparaît aussi lorsqu'on utilise trop de connexions (poids), d'où la phase d'apprentissage devient alors trop longue et les performances du réseau en généralisation deviennent médiocres, [2].

Pour éviter les problèmes de sur-apprentissage, l'ensemble des données d'expérience est divisé en deux parties : ensemble d'apprentissage et ensemble de test. L'optimisation des poids se fait sur l'ensemble d'apprentissage, mais les poids retenus sont ceux pour lesquels l'erreur obtenue sur l'ensemble de test est la plus faible. En effet, si les poids sont optimisés sur tous les exemples de l'apprentissage, on obtient une précision très satisfaisante sur ces exemples mais on risque de ne pas pouvoir généraliser le modèle à des données nouvelles. A partir d'un certain nombre d'itérations, le réseau ne cherche plus l'allure générale de la relation entre les entrées et les sorties du système, mais s'approche trop prés des points et apprend le bruit, [14].

L'apprentissage dans les réseaux de neurones nécessite le passage par trois étapes : le chois du paradigme d'apprentissage, le chois de la règle d'apprentissage, et enfin la matérialisation du concept d'apprentissage qui est le choix de l'algorithme d'apprentissage.

1.5.1 Paradigmes d'apprentissage

Avec l'avènement des ordinateurs et l'ère de l'information, les problèmes statistiques ont explosé à la fois en taille et en complexité. De grandes quantités de données sont

générées dans de nombreux domaines, et le travail du statisticien consiste a extraire et comprendre les tendances importantes des données.

On peut distinguer plusieurs modes d'apprentissages, mais l'objectif fondamental de l'apprentissage reste le même ; soit la classification, l'approximation de fonction ou encore la prévision. Nous allons citer les deux pratiques les plus reconnus : le mode supervisé et le mode non-supervisé.

1.5.1.1 Apprentissage supervisé

L'apprentissage supervisé est utilisé quand nous avons une base d'exemples qui contient, en même temps, les énoncés du problème et les réponses. En apprentissage supervisé, le réseau est alimenté avec les résultats corrects pour les exemples d'entrées donnés. Il a alors comme but d'approximer ces exemples aussi bien que possible et de développer à la fois la bonne représentation mathématique qui lui permet de généraliser ces exemples pour ensuite traiter de nouvelles situations.

La technique consiste à évaluer l'erreur pour chaque entrée x_i, et modifier les paramètres libres des neurones afin de minimiser l'erreur dans les prochaines itérations. D'une manière conceptuelle, la sortie désirée d_i est supposée optimale et comparée (par soustraction) avec la sortie du réseau pour produire un signal d'erreur e_i qui est réinjecté dans le réseau pour modifier son comportement via une procédure itérative qui, éventuellement, lui permet de simuler la réponse désirée. Autrement dit, la connaissance de l'environnement est graduellement transférée vers le réseau jusqu'à l'atteinte d'un certain critère d'arrêt (une valeur d'erreur minimum), [50].

1.5.1.2 Apprentissage non-supervisé

L'apprentissage est qualifié de non-supervisé lorsque seules les valeurs d'entrée sont disponibles. Dans ce cas, les exemples présentés à l'entrée provoquent une auto-adaptation du réseau afin de produire des valeurs de sortie qui soient proches, en réponse, à des valeurs d'entrée de même nature. Elle ne dispose ni d'un signal d'erreur (comme dans le cas supervisé), ni d'un indice de satisfaction (comme dans le cas par renforcement), [11]. L'apprentissage repose alors sur un critère interne de conformité du comportement du réseau par rapport à des spécifications générales et non sur des observations. L'apprentissage

non-supervisé est surtout utilisé pour le traitement du signal et l'analyse factorielle.

1.5.2 Règle d'apprentissage par correction d'erreur

Une règle d'apprentissage est une méthode qui améliore les performances et le temps d'entraînement d'un réseau. Habituellement, cette règle est appliquée à plusieurs reprises sur le réseau, en comparant les résultats attendus et les résultats réels du réseau pour donner des valeurs optimales.

La règle d'apprentissage est l'un des facteurs qui influe sur la vitesse et la précision du développement du réseau artificiel. Il existe plusieurs règles d'apprentissage : apprentissage par correction d'erreur, apprentissage basé sur la mémoire, apprentissage de Hebb, apprentissage compétitif et apprentissage de Boltzmann. Nous nous intéressons dans cette thèse au premier type d'apprentissage

L'apprentissage par correction d'erreur consiste à minimiser un indice de performance ε dans le but d'appliquer une séquence d'ajustement correctif au poids w_k du neurone k afin de minimiser l'erreur $e(n)$, entre la sortie désirée $t(n)$ et la sortie du neurone $y(n)$. Celle-ci conduit à une règle d'apprentissage communément appelée règle Delta ou règle de Widrow-Hoff, [14].

$$e(n) = t(n) - y(n) \tag{1.2}$$

où $e(n) = [e_1(n), e_2(n), \cdots, e_k(n), \cdots, e_p(n)]$ désigne le vecteur des erreurs observées sur les p neurones de sortie du réseau.

Soit $w_{k(n)}$ le vecteur des valeurs des poids synaptiques du neurone k liée par le vecteur signal $x(n)$ à l'étape de temps n. Conformément à la règle Delta, l'ajustement $\Delta w_{k(n)}$ appliqué au poids synaptiques w_k au temps n est défini par

$$\Delta w_k(n) = \eta \cdot e_k(n) \cdot x(n) \tag{1.3}$$

où η est une constante positive qui détermine le taux d'apprentissage (la vitesse d'apprentissage) lorsque nous passons d'une étape à une autre dans le processus d'apprentissage.

Il est important que η soit choisi avec précaution pour assurer la stabilité ou la convergence du processus itératif d'apprentissage. Le choix de η a aussi une profonde influence

sur l'exactitude et d'autre aspects du processus d'apprentissage, [50].

Le produit $e_k(n) \cdot x(n)$ désigne la direction dans la quelle nous allons chercher le minimum. La valeur mise à jour du poids synaptique w_k est basé sur l'algorithme de la descente du gradient et est déterminée par

$$w_k(n+1) = w_k(n) + \Delta w_k(n) \tag{1.4}$$

En effet, $w_k(n)$ et $w_k(n+1)$ peuvent être vues, respectivement, comme l'ancien et le nouvel vecteur de poids synaptique w_k.

1.5.3 Algorithmes d'apprentissage

L'algorithme d'apprentissage est la méthode mathématique de modification des poids de connexions afin de converger vers une solution qui permettra au réseau d'accomplir la tâche désirée. Les poids de connexions sont corrigés de manière à minimiser la différence entre les sorties calculées et les sorties désirées. Cette phase de minimisation est primordiale à l'efficacité du réseau. Ensuite, il est recommandée de procéder à une phase de vérification qui consiste à tester le réseau sur un ensemble des données qui n'ont pas servi à l'apprentissage, ce qui permet de vérifier le pouvoir de généralisation du réseau. Si l'étape de la vérification est satisfaisante, c'est-à-dire, si le réseau arrive à prédire correctement les couples entrées-sortie, n'ayant pas servi à l 'apprentissage, le réseau est efficace. Dans le cas contraire il faut recommencer l'apprentissage avec de nouveaux paramètres, [2], [14].

Plusieurs algorithmes itératifs peuvent être mis en œuvre, parmi lesquels on note l'algorithme de la rétro-propagation qui est le plus simple et le plus utilisé pour une tâche de modélisation.

1.5.3.1 Apprentissage basé sur la rétro-propagation

L'algorithme de la rétro-propagation est l'algorithme supervisé le plus utilisé actuellement dans le domaine des réseaux de neurones. C'est l'algorithme économique qui permet d'évaluer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première. C'est une technique de calcul des dérivées qui peut être appliquée à n'importe

quelle structure des fonctions dérivables. L'objectif de la méthode de la rétro-propagation est d'adapter les paramètres w_j de façon à minimiser la valeur moyenne de l'erreur sur l'ensemble d'entraînement.

La courbure de la surface d'erreur E en un point est exprimée par la deuxième dérivée d'erreur par rapport aux poids w, $(\partial^2 E/\partial w^2)$, lequel est obtenue en différenciant l'erreur dérivative $(\partial E/\partial w)$ en ce qui concerne un poids w. D'une manière générale, la distance entre les poids optimaux peut être estimée en divisant la première dérivée par la dérivée seconde, [50], [24]. Cela donne la distance nécessaire du changement du poids. Ainsi, le changement de poids peut être exprimé en tant que

$$\triangle\ w = -\frac{\partial E}{\partial w}/\frac{\partial^2 E}{\partial w^2} \tag{1.5}$$

Les poids sont alors initialisées à des valeurs aléatoire. Les données d'apprentissages sont présentés au réseau de manière répétée, et les poids sont ajustés de façon incrémentielle jusqu'à ce que la performance diminue progressivement.

La performance utilisé généralement est l'erreur moyenne quadratique, EMQ, représenté par $E(w)$ comme suit :

$$\begin{aligned} E(w) &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} e_i^2 \\ &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N}[t_i - y_i]^2 \\ &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N}[t_i - f(w; x_i)]^2 \end{aligned} \tag{1.6}$$

où e_i est l'erreur entre la sortie cible t_i et la sortie du réseau y_i qui est exprimée par la fonction d'ajustement f. Le nombre d'exemples d'apprentissage est N.

L'algorithme de la rétro-propagation souffre des limitations inhérentes à la technique du gradient à cause du risque d'être piégé dans un minimum local. Il suffit que les gradients ou leurs dérivées soient nuls pour que le réseau se retrouve bloqué dans un minimum local. Ajoutons à cela la lenteur de convergence lorsqu'il y a un grand nombre de poids de connexion à déterminer. Pour éviter l'obstacle des minima locaux, des recherches avancés sont effectués pour améliorer l'algorithme de la rétro-propagation et le rendre de plus en plus rapide et efficace, [50].

Il existe deux catégories d'algorithmes basés sur le même principe de la descente du

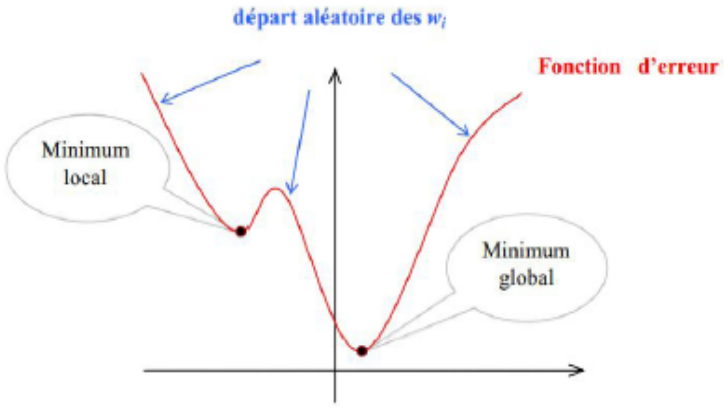


FIGURE 1.6 – Courbe explicative du phénomène du minimum local.

gradient : les algorithmes du premier ordre, qui sont basés sur l'algorithme de Newton et qui n'utilisent que le gradient, (qui indique combien l'erreur est sensible aux changements des poids pour atteindre l'optimum itérativement), et les algorithmes du second ordre qui sont plus coûteuses en calcul mais plus performantes puisqu'elles utilisent la dérivée seconde ou une valeur approchée, (qui mesure la courbure de la surface d'erreur), pour guider l'entraînement plus efficacement vers le bas de la surface d'erreur.

On peut citer plusieurs algorithmes du premier ordre : l'algorithme du Momentum, l'algorithme Delta-Bar-Delta (taux d'apprentissage adaptatif), et l'algorithme de la descente la plus raide.

Toutes les algorithmes de minimisation d'erreur se ressemblent en ce qu'ils sont itératifs. En commençant par des valeurs initiales des poids, ils mettent itérativement à jour les coefficients des poids dans le sens négatif du gradient par

$$w_m = w_{m-1} - \eta \cdot R \cdot d_m \tag{1.7}$$

où m est l'époque actuelle, η est une constante positive comprise entre 0 et 1 et qui détermine le taux d'apprentissage, et d_m est la somme des dérivées sur l'époque.

Le seul nouveau paramètre dans les algorithmes du second ordre, dont les diverses méthodes diffèrent, est R, qui a plusieurs variantes de dérivée seconde de la fonction d'erreur. Ceci est très utile pour changer la direction de recherche à partir du sens opposé du gradient vers une direction plus favorable, [50].

Le plus simple algorithme du second ordre est appelée Quick-Prop, qui a été proposée

pour accélérer le processus de la convergence de l'entraînement par la rétro-propagation. Il illustre les concepts de la minimisation de second ordre de l'erreur d'une manière simple en employant seulement la première dérivée. Il utilise une approximation de la dérivée seconde.

Il y a aussi, l'algorithme de Gauss-Newton, l'algorithme de la rétro-propagation de gradient conjugué, l'algorithme de Powell-Beale, l'algorithme BFGS (du nom de ses inventeurs : Broyden, Fletcher, Goldfarb et Shanno) quasi-Newton rétro-propagation, l'algorithme de Fletcher Reeves, l'algorithme de Polak Ribière, et à l'heure actuelle l'algorithme le plus performant et le plus largement utilisé est l'algorithme de Levenberg-Marquardt.

1.5.3.2 Algorithme de Levenberg-Marquardt

L'algorithme de Levenberg-Marquardt (LM) est développé par Kenneth Levenberg, puis publié par Donald Marquardt. Il permet d'obtenir une solution numérique au problème de minimisation d'une fonction, souvent non linéaire et dépendant de plusieurs variables. Il repose sur l'algorithme de Gauss Newton et l'algorithme du gradient.

L'algorithme de LM améliore la solution aux problèmes qui sont beaucoup plus durs à résoudre, même s'il démarre très loin d'un minimum, en ajustant seulement le taux d'apprentissage à plusieurs reprises. Au lieu d'ajuster η, l'algorithme de LM le place à l'unité et un nouveau terme e^{λ} est ajouté à la deuxième dérivée du terme qui est notée par $d^s = (\partial^2 E/\partial w^2)$, où e est la bijection réciproque de la fonction Logarithme népérien.

$$\triangle\ w_m = -\frac{d_m}{d_m^s + e^{\lambda}} \tag{1.8}$$

où λ est une valeur aléatoire choisie au début de l'entraînement.

La matrice Hessienne, H, qui représente l'ensemble des deuxièmes dérivées d'erreur par rapport à chaque poids, [53], est modifiée comme :

$$H' = H + e^{\lambda} I \tag{1.9}$$

Alors R qui est l'inverse de H devient :

$$R = \frac{1}{H'} = \frac{1}{H + e^{\lambda} I} \tag{1.10}$$

où I est une matrice d'identité.

La formule générale pour le changement de poids de tous les poids pour l'époque m peut être exprimée sous la forme de matrice comme :

$$\triangle\ w_m = -\frac{d_m}{H_m + e^{\lambda} I} \tag{1.11}$$

Dans l'algorithme de LM, λ est choisi automatiquement jusqu'à ce qu'une étape inclinée soit produit pour chaque époque. Commençant par une valeur initiale de λ, l'algorithme essaye de diminuer sa valeur par des incréments de $\triangle\ \lambda$ dans chaque époque, [50]. Si l'erreur quadratique moyenne (EQM) n'est pas réduit, λ est augmenté à plusieurs reprises jusqu'à un pas de colline plus bas soit produit.

Quand λ est petit, l'algorithme de LM est assimilée à l'algorithme de Gauss-Newton parce que le deuxième terme dans le dénominateur est petit et en conséquence, les deuxièmes dérivées jouent un rôle important dans l'équation d'ajustement de poids. En essayant de réduire λ au début, l'algorithme de LM tente d'employer les premiers et les deuxièmes dérivées de l'erreur afin d'utiliser leurs efficacité combiné, comme dans l'algorithme de Gauss-Newton.

Cependant, quand λ est grand, l'algorithme est assimilée à la descente la plus raide parce que le terme de traitement dans le dénominateur de l'ajustement du poids de l'équation devient grand. Ainsi, l'effet de la deuxième dérivée de l'erreur n'est pas significatif comparée à celui de la première dérivée, et l'erreur est réduite presque entièrement le long de la direction du gradient d'erreur négative, comme dans l'algorithme de la descente la plus raide. L'algorithme de LM recourt à cette approche et emploie un plus grand λ quand le changement de poids mène à une erreur accrue qui est provoquée en montant vers le haut, la surface d'erreur due aux problèmes lié au deuxième dérivé de l'erreur comme précédemment mentionné.

Ainsi, l'algorithme de LM est un algorithme qui combine les avantages de la descente la plus raide et des algorithmes de Gauss-Newton pour produire un algorithme plus efficace que l'un ou l'autre de ces deux algorithmes.

1.5.3.3 Algorithme de la Machine d'Apprentissage Extrême

Les algorithmes d'apprentissage traditionnels, (basés sur la descente de gradient), sont généralement beaucoup plus lents que nécessaire, en raison d'étapes d'apprentissage inappropriées ou peuvent facilement converger vers des minima locaux. De nombreuses étapes d'apprentissage itératives peuvent être exigées par ces algorithmes d'apprentissage afin d'obtenir de meilleures performances d'apprentissage.

Dans les applications réelles, Huang et Babri, montrent qu'un réseau de neurones à couche cachée (SLFN), [24] avec au plus N nœuds cachés et avec presque n'importe quelle fonction d'activation non-linéaire peut exactement apprendre N observations distinctes.

Contrairement à la croyance établie et aux mises en œuvre pratiques, que tous les paramètres des réseaux doivent être ajustés, les poids d'entrée et les biais de la couche cachée des SLFN peuvent être attribués au hasard si les fonctions d'activation dans la couche cachée sont infiniment différentiables, [20]. Sur la base de ce concept, les SLFN peuvent être simplement considérés comme un système linéaire et les poids de sortie, (reliant la couche cachée à la couche de sortie), des SLFN peuvent être déterminés analytiquement par une simple opération inverse généralisée, [47], de la matrice de sortie de la couche cachée. Cet algorithme d'apprentissage simple pour les SLFN appelé Machine d'Apprentissage Extrême (MAE) dont la vitesse d'apprentissage peut être extrêmement rapide que les algorithmes traditionnels tout en obtenant de meilleures performances de généralisation. IL a non seulement tendance à atteindre la plus petite erreur d'apprentissage, mais également la plus petite norme de poids.

L'inverse généralisé de Moore-Penrose et la solution des moindres carrés de la norme minimale d'un système linéaire général qui jouent un rôle important dans le développement de notre nouvel algorithme d'apprentissage, MAE, [56].

Étant donné un ensemble de N observations arbitraires distincte (x_i, t_i), où $x_i = [x_{i1}, x_{i2}, ..., x_{in}]^T \in \mathbb{R}^n$, $t_i = [t_{i1}, t_{i2}, ..., t_{im}]^T \in \mathbb{R}^m$, et $n, m \in \mathbb{N}$.

Les SLFN standard avec L neurones cachés et la fonction d'activation $g(x)$ sont modélisés mathématiquement par

$$y_j = \sum_{i=1}^{L} \beta_i \cdot g_i(x_j) = \sum_{i=1}^{L} \beta_i \cdot g(w_i \cdot x_j + b_i); \qquad j = 1, ..., N. \tag{1.12}$$

où $w_i = [w_{i1}, w_{i2}, ..., w_{in}]$ est le vecteur des poids reliant le $i^{\text{ème}}$ neurones caché et les neurones d'entrée, $\beta_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ est le vecteur des poids reliant le $i^{\text{ème}}$ neurone caché et les neurones de sortie, b_i est le seuil du $i^{\text{ème}}$ neurone caché, et $(w_i \cdot x_j)$ dénote le produit scalaire de w_i et x_j. Les neurones de sortie sont choisi linéaire ici.

Les SLFN standard avec L neurones cachés avec la fonction d'activation $g(x)$ puissent approcher ces N échantillons avec une erreur nulle

$$\sum_{j=1}^{L} \|y_j - t_j\| = 0. \tag{1.13}$$

Alors, il exist β_i, w_i et b_i tel que

$$\textstyle\sum_{i=1}^{L} \beta_i \cdot g(w_i \cdot x_j + b_i) = t_j \quad ; \qquad j = 1, ..., N. \tag{1.14}$$

Ce qui peut être écrit sous la forme matricielle suivante

$$H\beta = T, \tag{1.15}$$

où

$$H = \begin{pmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{pmatrix} = \begin{pmatrix} g(w_1 \cdot x_1 + b_1) & \dots & g(w_L \cdot x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_N + b_L) & \dots & g(w_L \cdot x_N + b_L) \end{pmatrix}_{N \times L} \quad ; \tag{1.16}$$

$$\beta = \begin{pmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{pmatrix}_{L \times m} \quad et \quad T = \begin{pmatrix} T_1^T \\ \vdots \\ T_N^T \end{pmatrix}_{N \times m} \tag{1.17}$$

H est appelée la matrice de sortie de la couche cachée du réseau de neurones.

La $i^{\text{ème}}$ colonne de H est la $i^{\text{ème}}$ sortie du neurone caché par rapport aux entrées $x_1, x_2, ..., x_N$.

Alors les poids de sortie sont donnés par

$$\beta = H^{-1}.T \tag{1.18}$$

où T est la matrice cible des données d'apprentissage et H^{-1} est l'inverse généralisé de

Moore-Penrose de la matrice H.

En général, L'MAE est une sorte de réseaux de neurones de régularisation, mais avec des mappages de couches cachées non ajustées, (formées par des neurones cachés aléatoires, des noyaux où d'autres implémentations). L'algorithme de l'MAE fonctionne pour toute fonction d'activation $g(x)$ infiniment différentiable. Cette fonction d'activation comprend les fonctions sigmoïdes, les fonctions à base radiale (FBR), le sinus, le cosinus, les fonctions exponentielles, et beaucoup d'autres fonctions. La limite supérieure du nombre requis de neurones cachés, L, est le nombre d'observations distinctes N ($L \leq N$).

Théorème 1. *Étant donné un SLFN standard avec N nœuds cachés et une fonction d'activation $g : \mathbb{R} \to \mathbb{R}$ qui est infiniment différentiable dans tout intervalle.*

Pour N échantillons distincts arbitraires (x_i, t_i), où $x_i \in \mathbb{R}^n$ et $t_i \in \mathbb{R}^m$, pour tout w_i et b_i choisis au hasard parmi tous les intervalles de $\mathbb{R}^n$ et $\mathbb{R}$, respectivement, selon toute distribution de probabilité continue, puis avec une probabilité un, la matrice de sortie de la couche cachée H du SLFN est inversible et $\|H\beta - T\| = 0$.

Démonstration. [20] □

Théorème 2. *Étant donné toute petite valeur positive $\varepsilon > 0$ et une fonction d'activation $g : \mathbb{R} \to \mathbb{R}$ qui est infiniment différentiable dans tout intervalle.*

Il existe $L \leq N$ tel que pour N échantillons distincts arbitraires (x_i, t_i), où $x_i \in \mathbb{R}^n$ et $t_i \in \mathbb{R}^m$, pour tout w_i et b_i choisis au hasard parmi tout intervalle de $\mathbb{R}^n$ et $\mathbb{R}$, respectivement, selon toute distribution de probabilité continue, puis avec une probabilité un, $\|H_{N\times L}\beta_{L\times m} - T_{N\times m}\| < \varepsilon$.

Démonstration. , [20] □

La Machine d'apprentissage Extrême (MAE) offre de meilleures performances de généralisation à une vitesse d'apprentissage beaucoup plus rapide et avec moins d'intervention humaine, [21].

CHAPITRE 2

MODÉLISATION STATISTIQUE

2.1 Préface

La plupart des techniques statistiques standard sont des méthodes paramétriques (e.g. la régression linéaire et la régression logistique), dont l'ajustement du modèle et accomplie en choisissant des valeurs optimales pour les paramètres (ou en trouvant leur distribution postérieure). Cela est réalisé en supposant que le choix de la famille de modèles est la bonne famille, (e.g. une relation linéaire avec une erreur gaussienne indépendante).

Au cours de cette dernière décennie, les statisticiens ont réalisé que la pensée paramétrique pure dans les estimations de courbes ne répond souvent pas au besoin de flexibilité dans l'analyse des données et le développement du matériel a créé la demande pour la théorie des estimations non-paramétriques désormais calculables, [25].

L'approche non-paramétrique a été utilisée comme un outil majeur d'analyse empirique. La flexibilité de la méthode est extrêmement utile dans une analyse statistique préliminaire et exploratoire d'un ensemble de données. Si aucune information de modèle a priori sur la courbe de régression n'est disponible, l'analyse non-paramétrique pourrait aider à suggérer des formulations paramétriques simples de la relation de régression.

Dans ce chapitre, nous montrerons notamment la conception et toutes les connaissances mathématiques disponibles concernant la modélisation statistique non-paramétrique. Nous allons, en premier, donner les différentes méthodes d'estimation non-paramétrique. En suite, nous allons détailler la méthodologie statistique de la modélisation par réseaux

de neurones artificiels, dont le but essentiel est de bien voir que ces derniers sont des modèles statistiques pures.

2.2 Modélisation non-paramétrique

L'objectif de la régression non-paramétrique est de construire la fonction continue qui soit l'approximation performante des données. Le but est de choisir la famille particulière qui contient le modèle de la régression le plus proche de la variation des données, en évitant le sur-ajustement (voir Section 1.5). Dans ce cas, le modèle n'est pas limité aux fonctions linéaires, ni même aux fonctions dérivables mais il est non-paramétrique dans le sens où le modèle estimé n'a pas un nombre fini de paramètres. Donc, choisir l'espace de dimension fini qui estime l'espace de dimension infini, [34].

Le modèle de régression non-paramétrique est de la forme

$$y_i = f(x_i) + \varepsilon_i \quad ; \qquad i = 1, \cdots, n; \tag{2.1}$$

où $f \in \mathcal{F}$ une classe de fonctions de régression, x_i est le vecteur de variables explicatives, y_i est le vecteur de variables expliqués, et ε est une erreur additive iid avec une moyenne nulle et une variance constante, généralement supposée avoir une distribution gaussienne.

Les méthodes non-paramétriques se diffèrent par la classe de fonctions de régression $\mathcal{F}$. Chaque fonction de régression f, parmi cette classe, devrait pouvoir rapprocher une très large gamme de fonctions, telle que l'ensemble de toutes les fonctions continues, ou l'ensemble de toutes les fonctions carrées intégrables, [53].

Plusieurs méthodes d'approximation ont était développé comme les méthodes d'approximation locale et les méthodes de la représentation de base, [34]. Ces deux méthodes montrent bien que les réseaux de neurones sont des méthodes statistiques non-paramétriques.

2.2.1 Méthodes d'approximation locale

Les méthodes d'approximation locale sont des méthodes plus simples, utilisés pour l'approximation de la fonction de régression non-paramétrique. Nous avons des méthodes

d'estimation de la fenêtre mobile et des méthodes d'estimation par partition d'espace.

2.2.1.1 Estimation de la fenêtre mobile

Dans une dimension, une moyenne mobile avec une taille de fenêtre fixe serait un exemple simple et local, où l'estimation de la moyenne mobile est une fonction étagée. L'idée est de choisir la fenêtre, le poids de la moyenne (le Kernel) ou la forme de l'ajustement au dessus de la fenêtre.

L'idée d'estimation par noyau est d'utiliser une moyenne pondérée mobile, où la fonction de poids est appelée noyau. Un noyau est une fonction continue, bornée, symétrique dont l'intégrale est un. Le noyau le plus simple serait la fonction indicatrice sur l'intervalle $[-\frac{1}{2}, \frac{1}{2}]$,

$$K(x) = I_{[-\frac{1}{2} \leq x \leq \frac{1}{2}]}.$$

Il y a aussi le noyau d'Epanechnikov ou le noyau gaussien qui est une fonction de densité a une distribution gaussienne standard, [34].

$$K(x) = \frac{3}{4}(1 - x^2) I_{[-1 \leq x \leq 1]}.$$

Quel que soit le choix du noyau, l'estimation de la fonction de régression résultante à une valeur x d'une seule variable explicative est

$$\hat{y} = \frac{\sum_{i=1}^{n} K(x - x_i)}{\sum_{i=1}^{n} K(x - x_i)}. \tag{2.2}$$

La régression par noyau est étroitement liée à l'estimation de la densité par noyau, [52].Les noyaux peuvent être généralisés à plusieurs dimensions, mais la rareté des données est un grand problème dans ce cas. Une généralisation pertinente d'une approche à noyau gaussien est la régression de processus gaussien, qui définit un processus sur tout l'espace avec la distribution de tout ensemble fini de points étant une gaussienne multivariée, [41].

Le lissage du nuage de points à pondération locale, LOWESS, (Locally Weighted Scatterplot Smoothing), Patrick Laurie Davies (2014), est une autre méthode d'ajustement qui utilise des polynômes à pondération locale. L'algorithme LOWESS d'origine prend également des mesures supplémentaires pour s'assurer que l'ajustement est robuste par

rapport aux valeurs aberrantes.

2.2.1.2 Estimation par partition d'espace

Une approche complètement différente consiste à abandonner la fenêtre mobile au profit d'une partition de l'espace en un ensemble exhaustif d'intervalles (ou régions) mutuellement exclusifs. L'utilisation d'une fonction constante sur la région donne lieu à une fonction étagée. Dans une dimension, il s'agit d'un estimateur d'histogramme, [18].

Dans les dimensions supérieures, un arbre est une méthode utile pour représenter la division de l'espace en régions. Avec un ajustement constant sur chaque région, cette méthode est connue sous le nom de régression de partitionnement récursif (RPR) ou d'arbres de classification et de régression (CART), [34].

CART est devenu populaire en raison de sa facilité d'utilisation, de l'interprétation claire des résultats et de sa capacité à s'adapter raisonnablement bien dans de nombreux cas.

Pour la régression, la valeur prédite est la valeur moyenne de la variable de réponse. Pour la classification, la probabilité ajustée d'appartenance à une classe est la probabilité empirique (le nombre d'observations dans cette classe divisé par le nombre total).

Les modèles arborescents peuvent être étendus pour permettre l'ajustement de la régression linéaire ou d'autres modèles plus complexes sur chacune des régions. Des versions bayésiennes de CART ont également été explorées. Les modèles de partition peuvent également être généralisés au-delà des arbres, comme avec une partition de pavage de Voronoi, [41].

En revenant à une partition sur une seule dimension, le lissage Spline utilise des polynômes d'ordre inférieur sur les éléments de la partition, avec l'ordre maximum spécifié dans le cadre du modèle. Les points limites des partitions sont appelés nœuds, et il est nécessaire que l'ajustement de Spline résultant soit continu au niveau des nœuds. En outre, la continuité des dérivées est également obligatoire, assurant ainsi la régularité de la fonction de régression ajustée, [53].

Les Splines se sont très utiles dans les problèmes unidimensionnels. La généralisation à plusieurs dimensions est plus problématique. L'idée canonique est d'utiliser des produits tensoriels de Splines unidimensionnelles, [18]. Bien que conceptuellement simple, il existe

de nombreuses complications pratiques. Chui (1988) fournit une discussion approfondie des Splines multivariées. Un cas particulier est celui des Splines de régression adaptative multivariée (MARS), qui utilise une fonction de base ; crosse de hockey, [34]

$$f_j(x_{ij}) = s_j(x_{ij} - t_j)_+$$

où s est une fonction signe (soit $+1$ ou -1) et t est un seuil. Il s'agit d'une Spline linéaire avec un seul nœud à t_j, et elle est mise à zéro d'un côté du nœud (en fonction de s). Ceci est adapté au cadre multivarié en prenant un produit tensoriel de ces Splines linéaires,

$$f(x_i) = \prod_j f_j(x_{ij})$$

Denison, Mallick et Smith décrivent une version bayésienne de MARS (1998b). Une alternative aux produits tensoriels consiste en des combinaisons linéaires simples, c'est-à-dire

$$f(x_i) = \sum_j \beta_j(\gamma^t x_i)_+ + \varepsilon_i$$

où γ sont les coefficients de la Spline hyperplane et $(z)_+ = max(0, z)$.

Une extension multivariée aux Splines plus simple à utiliser en pratique est le modèle additif généralisé (GAM), [34]. Un ajustement Spline distinct est utilisé pour chaque variable explicative afin que seules des Splines univariées soient nécessaires. Ils sont ajustés simultanément pour donner un modèle additif :

$$y_i = \sum_{j=1}^{r} f_j(x_{ij}) + \varepsilon_i$$

où chaque f est une Spline unidimensionnelle. De tels modèles sont assez flexibles et calculables.

Une autre généralisation de GAM est la régression de poursuite de projection (PPR), qui reconnaît que l'additivité sur les axes de coordonnées d'origine peut être insuffisante, mais qu'une certaine rotation des axes peut fournir un bien meilleur ajustement. Ainsi, il utilise des combinaisons linéaires des variables explicatives comme nouvelles entrées des

Splines additives :

$$y = \sum_{j=1}^{r} f_j(\beta^t x) + \varepsilon \tag{2.3}$$

où les $(f_j)_{j=1,\cdots,r}$ sont des Splines univariées. L'idée conceptuelle est que les (f_j) sont totalement flexibles, et donc PPR ajuste une combinaison linéaire de fonctions lisses arbitraires sur un nouveau système de coordonnées choisi pour décrire au mieux les données. En ce sens, nous verrons qu'un réseau de neurones peut être considéré comme un cas particulier de PPR, où la forme fonctionnelle de f_j est restreinte à des fonctions logistiques mises à l'échelle.

Remarque 3. *CART, MARS et PPR peuvent également être considérés comme des représentations de base, car la classe des fonctions à seuil peut rapprocher n'importe quelle fonction continue. Ainsi, l'ensemble de base de CART peut être vu pour rapprocher des fonctions lisses. En ce qui concerne MARS, la base comprise des fonctions de bâton d'hockey et de leurs produits tensoriels.*

2.2.2 Méthodes de représentation de base

Une grande classe de méthodes non-paramétriques sont celles qui utilisent un ensemble infini de fonctions de base pour couvrir l'espace d'intérêt, généralement soit l'espace des fonctions continues, soit l'espace des fonctions carrées intégrables, [53]. Les réseaux de neurones entrent également dans cette catégorie. Le concept clé est que le modèle de régression est de la forme

$$y_i = \sum_{j=1}^{k} \beta_j f_j(x_i) + \varepsilon_i \tag{2.4}$$

où $f_1, f_2, \cdots, f_k$ est un ensemble de k fonctions de base. Typiquement f_0 est défini comme étant 1 partout, fournissant le terme constant. Les ε_i sont les erreurs liés aux sorties y_i.

2.2.2.1 Fonctions Polynomiales

Un exemple simple d'un ensemble de base qui couvre l'espace des fonctions continues dans une dimension est l'ensemble des fonctions polynomiales de base, $1, x, x^2, x^3, \cdots, x^k$, c'est-à-dire :

$$f_j(x) = x^j \quad ; \quad j = 1, \cdots, k; \tag{2.5}$$

Toute fonction continue peut être approchée arbitrairement étroitement avec une combinaison linéaire de ces fonctions, à condition qu'un nombre suffisant d'entre elles soient utilisées. En pratique, il est souvent préférable que l'ensemble de base se compose des éléments orthogonaux, [25].

2.2.2.2 Séries de Fourier

Un autre exemple bien connu de représentation d'ensemble de base est la régression en série de Fourier. En prenant un nombre suffisamment grand de fonctions sinus et cosinus de fréquences et d'amplitudes différentes, on peut approcher n'importe quelle fonction carrée intégrable, [25], [18], aussi bien qu'on le souhaite. Cet ensemble de base présente également l'avantage de l'orthogonalité. Si la fonction sous-adjacente est périodique, les coefficients auront des interprétations significatives, mais si la fonction est apériodique, il est difficile d'avoir une intuition pour les coefficients, un problème partagé par de nombreuses méthodes de fonctions de base, y compris les réseaux de neurones.

2.2.2.3 Modèles de mélange

Les modèles de mélange peuvent être des représentations de base, selon la forme particulière du modèle. Une forme courante de mélanges, en particulier pour l'estimation de la densité mais aussi pour la régression, est un mélange de normales. Une collection de densités gaussiennes est utilisée comme ensemble de base.

Pour un problème univarié, le modèle est

$$y_i = \beta_0 + \sum_{j=1}^{k} \phi(\frac{x_i - \mu_j}{\sigma_j}) + \varepsilon_i \tag{2.6}$$

où $\phi(x) = \frac{1}{\sqrt{2\pi}} exp(-x^2/2)$ est la fonction de densité gaussienne standard. Pour x multivarié, une distance $\|x_i - \mu_j\|$, habituellement euclidienne, remplace la différence $(x_i - \mu_j)$.

Dans le contexte de l'apprentissage automatique, lorsque tous les σ_j sont égaux, ce modèle est appelé réseau de base radial, [34].

2.2.2.4 Ondelettes

Les Ondelettes ont gagné en popularité, en particulier pour le traitement du signal mais aussi pour la régression non-paramétrique en général. Ils fonctionnent également en créant un ensemble de fonctions de base. Ils prennent une fonction de départ, $\phi(x)$ (fonction générique), appelée Ondelette de mère, puis utilisent une fonction d'échelle pour obtenir l'ensemble de base complet. Les autres bases sont des versions à échelle de localisation entière, $\phi_{j,k} = 2^{j/2}\phi(2^j x - k)$, [34]. Cela produit un ensemble de base orthogonal (infini).

La figure 2.1 montre l'ondelette mère de Haar (ligne continue) et deux autres membres de la famille (avec $j = 1, 2$ et $k = 0$).

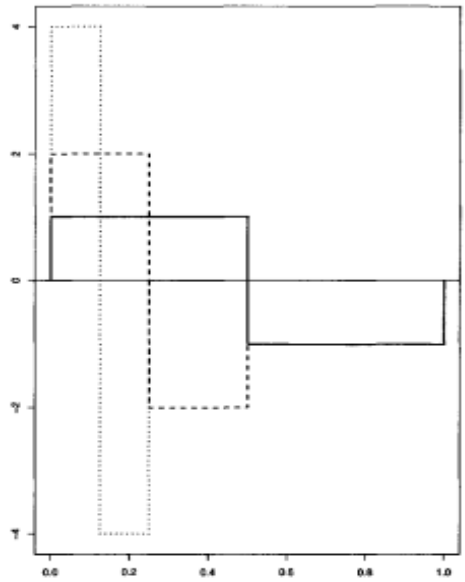


FIGURE 2.1 – Ondelettes mère de Haar

Il existe de nombreux autres choix largement utilisés d'Ondelettes mères et de fonctions d'échelle associées, dont certaines produisent des ensembles de base orthogonaux et d'autres qui ne le font pas. Les réseaux d'Ondelettes et les réseaux d'Ondelettes bayésiens peuvent également être appliqués à la régression non-paramétrique, [34].

Remarque 4. *CART et MARS sont généralement présentés comme faisant partie de la famille des méthodes d'approximation locale, ils peuvent également être considérés comme des représentations de base. L'un des outils de base utilisés dans l'analyse réelle est le fait que la classe des fonctions échelonnées peut approximer n'importe quelle fonction continue.*

Ainsi, CART peut être considéré comme une méthode de représentation de base en utilisant des fonctions pas à pas comme ensemble de base pour l'approximation des fonctions lisses.

Pour MARS, l'ensemble de base comprend les fonctions des bâton de hockey et leurs produits tensoriels. En effet, toute l'idée de l'ajustement de Spline est d'utiliser le fait que les polynômes s'étendent sur l'espace des fonctions continues. En partitionnant l'espace, on peut généralement se débrouiller avec des polynômes d'ordre très faible, au lieu d'exiger un grand nombre de termes.

Les splines, ainsi que la PPR, peuvent être considérées comme des méthodes de représentation de base approximative.

La figure 2.2 résume l'aperçu des techniques de régression non-paramétrique dans les classes relatives aux réseaux de neurones. Elle montre les méthodes locales à gauche et les méthodes de représentation de base à droite.

Les lignes relient les méthodes associées. Une ligne continue noir d'une méthode à l'autre montre que la seconde méthode est un cas particulier de la première. Par exemple, un estimateur d'histogramme est un cas particulier d'un estimateur d'arbre.

Les lignes continue bleu indiquent des approximations et des relations limites. Les PPR et les Splines sont des approximations de représentations de base. Un cas limite d'un réseau de neurones peut être un cas particulier d'un processus gaussien.

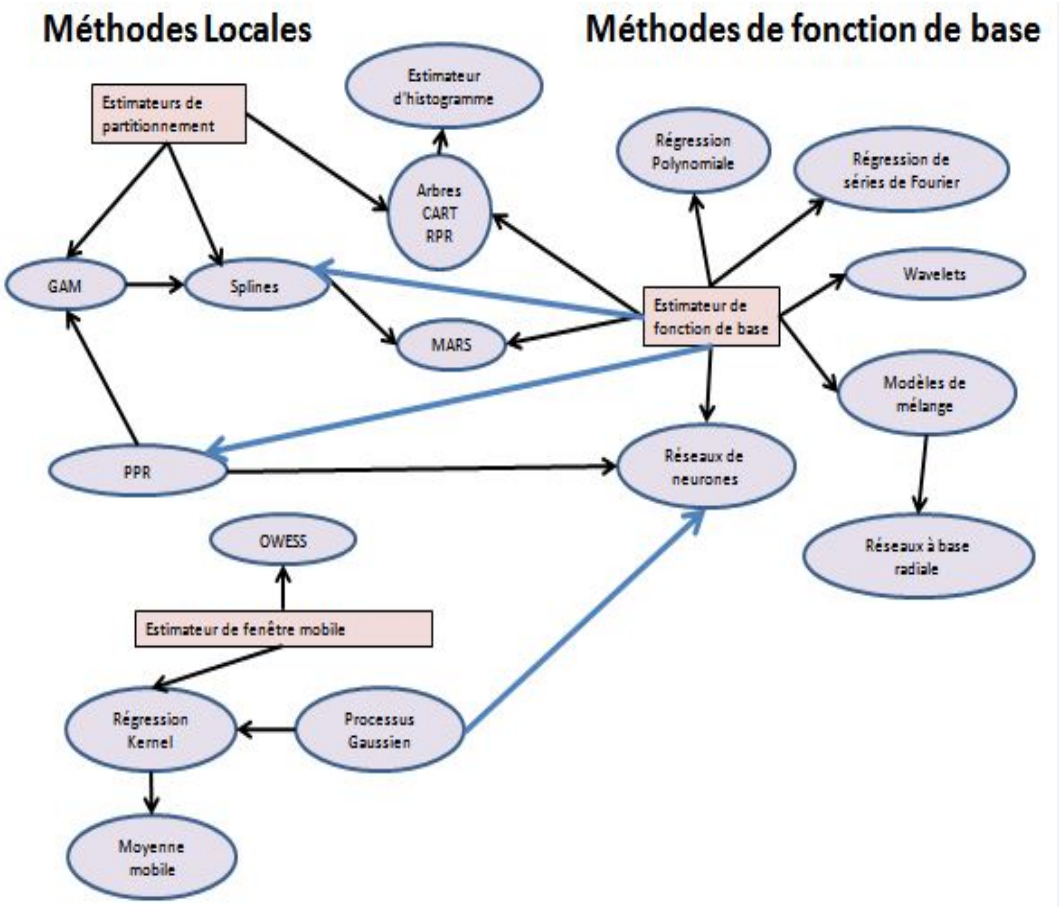


FIGURE 2.2 – Diagramme des méthodes non paramétriques : les méthodes locales sont à gauche et les méthodes de base sont à droite ; les cas particuliers sont représentés par un trait plein noire avec (a $\rightarrow$ b) signifiant que b est un cas particulier de a ; les trait plein bleu représentent des approximations et des cas limites.

2.3 Modélisation par réseaux de neurones

2.3.1 Régression statistique et réseaux de neurones

Les réseaux de neurones ont été généralement développés à partir de la perspective des machines d'apprentissage. Ils ont été créés comme une tentative de modéliser le fonctionnement d'un neurone biologique. Les premiers travaux, sur cette modélisation, sont faits par McCulloch et Pitts (1943), dont les nœuds sont des fonctions à seuil. Ils ont été explorés par Rosenblatt (1962) sous le nom Perceptrons, et Widrow et Hoff (1960) sous le nom d'Adalines, [61].

Mathématiquement, Hornik, Stinchcombe, et White (1989) ont prouvé que l'ensemble infini de fonctions logistiques de location scalaire est une base réglée (un recouvrement) pour beaucoup d'espaces communs, tels que l'espace des fonctions continues, ou des fonctions carrées intégrables, [34].

Les modèles de réseaux de neurones dans le contexte de la régression sont sous la forme suivante :

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j \varphi(W_j^t x) + \varepsilon \tag{2.7}$$

où k est le nombre de fonctions de base (nœuds cachés), les β_j sont les poids des fonctions de base, les W_j sont des vecteurs de paramètres de localisation et d'échelle (avec des éléments w_{jh}, où h indexé sur les covariables), $\varepsilon \sim N(0, \sigma^2)$ est le terme d'erreur, et φ est la fonction logistique :

$$\varphi(z) = \frac{1}{1 + exp(-z)} \tag{2.8}$$

Un réseau de neurones est généralement décrit en termes de nœuds cachés. Chaque nœud caché peut être considéré comme une fonction, prenant une combinaison linéaire des variables explicatives ($W_j^t x_i$) en entrée et renvoyant la transformation logistique en sortie. Le réseau de neurones prend ensuite une combinaison linéaire des sorties des nœuds cachés pour donner la valeur ajustée en un point.

Pour une taille de réseau fixe k, et pour un nombre d'entrées fixe r, le nombre de paramètres, d, est

$$d = (r + 1) \cdot k + (k + 1) = k \cdot (r + 2) + 1 \tag{2.9}$$

où le premier terme de la somme est le nombre de w_{jh} et le second est le nombre de β_j.

Il ressort clairement de l'équation 2.7 qu'un réseau de neurones est simplement une régression non-paramétrique utilisant une représentation de base (comparer à l'équation 2.3), avec les φ_j, fonctions logistiques à l'échelle de la localisation, comme bases et les β_j comme leurs coefficients. L'ensemble (infini) de fonctions logistiques à échelle de localisation couvre l'espace des fonctions continues, ainsi que l'espace des fonctions carrées intégrables, [18]. Bien que ces bases ne soient pas orthogonales, elles se sont avérées très utiles en pratique, avec un nombre relativement petit capable d'approcher un large éventail de fonctions. Ainsi, un réseau de neurones peut être interprété comme une combinaison linéaire de régressions logistiques.

D'après l'équation 2.7, il est évident qu'un réseau de neurones est un modèle paramétrique standard, avec une vraisemblance et des paramètres à ajuster. Lorsqu'on les compare aux modèles de représentation de base précédents, on peut voir comment les réseaux de neurones ne sont qu'un autre exemple d'une méthode de fonction de base pour la régression non-paramétrique.

La vision d'un réseau de neurones comme un modèle statistique facilite la compréhension, l'interprétation, l'amélioration, et l'emploi convenable de ces modèles. Ainsi, sa vision comme modèle de base rend la discussion des issues plus systématique telles que le choix préalable, la construction d'un modèle, la vérification des suppositions de la validité d'un modèle, et la compréhension incertaine dans nos prévisions. Également, un réseau de neurones peut être considéré comme un cas particulier de régression de poursuite de projection (équation 2.3) où les fonctions lisses arbitraires sont restreintes aux fonctions logistiques mises à l'échelle. De plus, avec un grand nombre de fonctions de base, un réseau de neurones peut être construit pour converger à un modèle de processus gaussien, [34].

2.3.2 Régression multivariée par réseaux de neurones

Un modèle de régression multivariée se traite par le raccordement de chaque nœud caché, après leurs traitements de façon séparé, et cela par la supposition que la variance d'erreur est la même a chaque dimension. Alors chaque dimension de la réponse est modélisée comme une combinaison linéaire différente, d'une même base de fonction.

Un réseau à propagation avant est un réseau de couches multiples de sorte que les entrées à chaque couche soient des combinaisons linéaires des sorties de la couche précédente ce qui donne un modèle de la forme

$$y = \beta_0 + \gamma^t x + \sum_{j=1}^{k} \beta_j \varphi(W_j^t x) + \varepsilon \tag{2.10}$$

qui est souvent appelé un modèle semi-paramétrique, car il contient à la fois une composante paramétrique (une régression linéaire standard, $\gamma^t x$,) et une composante non-paramétrique (la combinaison linéaire de bases logistiques à l'échelle de la localisation).

Les fonctions de base logistique sont les plus couramment utilisées. En théorie, toute fonction sigmoïde peut être utilisée, car l'ensemble infini comprend un ensemble de base et ne dépendent que du fait que les fonctions sont sigmoïdes. En pratique, l'une des deux fonctions sigmoïdes généralement choisie, soit la tangente logistique, soit la tangente hyperbolique :

$$\varphi'(z) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.11}$$

qui est une simple transformation de la fonction logistique, $\varphi'(z) = 2\varphi(2z) - 1$.

Un type complètement différent de fonction de base, parfois utilisé, est la fonction de base radiale qui est généralement appelée noyau en statistique. Ainsi, un réseau de base radiale est fondamentalement appelé un modèle de mélange. Les modèles de réseaux de neurones, en particulier les versions de fonction de base radiale, sont parfois utilisés pour des problèmes d'estimation de densité, [19], [52].

Dans notre modèle de base, nous utilisons une sortie linéaire dans la mesure où la prédiction finale est simplement une combinaison linéaire des sorties de la couche cachée. Dans d'autres implémentations, la transformation logistique peut également être appliquée à la prédiction finale, produisant des prédictions dans l'intervalle unitaire. Il ne semble pas y avoir d'avantage substantiel à le faire dans un problème de régression. Pour un problème de classification, une telle transformation supplémentaire peut être très utile, [41].

CHAPITRE 3

CONSTRUCTION DE MODÈLES

3.1 Introduction

Depuis l'aube des temps, l'homme pratique sa vie quotidienne en utilisant la technologie avancée quand il essaie de répondre aux problèmes et questions sur l'étude des modèles dans de divers domaines. C'est-à-dire, la construction des modèles pour tout problème soustrait grâce à la performance accrue des ordinateurs.

En statistique, la construction des modèles est un sujet essentiel dans les étapes d'apprentissage des réseau de neurones. En effet, avant de réaliser un modèle, il faut identifier la nature des données. Dans les domaines de traitement, des techniques d'acquisition de données ont été développées et avec ce développement les données mesurées disponibles sont plus en plus nombreuses. Il est maintenant possible d'analyser de grandes quantités de données de dimension élevée grâce à l'évolution technologique.

Ce qui peut poser en problème lors de l'analyse des données décrites par un très grand nombre de variables, est que les méthodes d'analyse, d'apprentissage, ou de feuille de données peuvent se révéler inefficaces ou peuvent conduire à des résultats inexacts. La résolution du problème devient parfois difficile, à cause de la dimensionnalité trop importante de ces données, et pour répondre à ce problème, il est souvent utile, et nécessaire de réduire la taille de données à une taille plus compatible, en utilisant des méthodes de résolution dédiées, [48].

L'objectif de ce chapitre est de déterminée les méthodes de la construction de modèles

statistiques et la régression multivariées par le moyen des réseaux de neurones artificiels pour obtenir un bon modèle à la fois performant (les plus petits résidus possibles) et économique (le moins possible de variables explicatives).

3.2 Organisation des données

La résolution de beaucoup de problèmes statistiques se base sur le traitement des données. Lorsque la taille des données de traitement est plus grande ainsi que leur dimension augmente, et lorsque l'exploitation de ces données se révèle difficile en pratique ; il est nécessaire de réduire la dimension à une taille compatible en sélectionnant les variables pertinentes pour caractériser le problème étudié.

Les deux grandes catégories de réduction d'espace sont : L'extraction des caractéristiques et La sélection des caractéristiques. Après cela, on définit la pertinence d'une caractéristique, et les différentes étapes du processus général de la sélection de variables. Nous présenterons quelques méthodes d'élimination des caractéristiques qui sont classées en deux méthodes : les méthodes supervisés (Filtre et Wrapper) et les méthodes non-supervisés (Embedded et Régularisation), [34].

3.2.1 Extraction de caractéristiques

Une réduction basée sur une transformation des données est appelée extraction des caractéristiques. Elle permet de créer de nouveaux ensembles de caractéristiques, en utilisant une combinaison des caractéristiques d'espace de départ, en exécutant une réduction du nombre de dimensions. Elle commence à partir d'un ensemble initial des données mesurées et construit des valeurs dérivées destinées à être informatives et non-redondantes, facilitant les étapes d'apprentissage.

La modélisation du problème est liée à la sélection de meilleures caractéristiques représentatives. Pour cela, on distingue usuellement les caractéristiques globales qui sont calculées sur toutes les données et les caractéristiques locales qui sont calculées autour du point d'intérêt. La façon du classement des caractéristiques est d'examiner s'ils sont extraits globalement ou localement de régions spécifiques de l'ensemble de données.

3.2.2 Sélection et réduction de variables

La taille des données peut être mesurée selon deux dimensions, le nombre de variables et le nombre d'exemples. L'augmentation de ces dimensions peut poser un problème lors de l'exploration et l'analyse de ces données. Il est fondamental de mettre en place des outils de traitement de données permettant une meilleure compréhension de la valeur des connaissances disponibles dans ces données.

La réduction des dimensions est un processus qui consiste à prendre des données dans un espace de grande dimension, et à les remplacer par des données dans un espace de plus petite dimension selon un critère de performance.

La sélection de variables est un processus utilisé en apprentissage automatique et en traitement de données. C'est une méthode essentielle pour le prétraitement de données afin de supprimer les variables bruitées ou inutiles. Le choix d'un ensemble optimal de descripteurs, ne signifie pas obligatoirement la sélection d'un ensemble composé seulement des variables jugées pertinentes et utiles. Il peut y contenir des variables non-pertinentes, mais qui ont de meilleures performances, prises avec d'autres variables.

Des sous-ensembles de variables plus petits permettent une meilleure généralisation des données en évitant le sur-apprentissage. Le choix d'un sous ensemble de variables pertinentes permet de réduire le temps d'apprentissage et d'exécution et améliore parfois la précision.

A partir de l'ensemble initial des variables, le processus de sélection détermine un sous-ensemble de variables qu'il les considère comme les plus pertinentes. Le sous ensemble est ensuite soumis à une procédure d'évaluation. En fonction du résultat de la procédure d'évaluation, un critère d'arrêt du processus détermine si le sous-ensemble de variables peut être soumis à la phase de validation. Si c'est le cas, le processus de sélection s'arrête, sinon, un autre sous-ensemble de variables est généré.

Pour un nombre de variables raisonnable m, le nombre de sous-ensemble a étudier, $(2m - 1)$, est donc considérable. Pour affronter ce problème de taille de l'espace de recherche, les algorithmes se divisent en trois grandes familles : Exhaustive (Complète), Heuristique et Aléatoire.

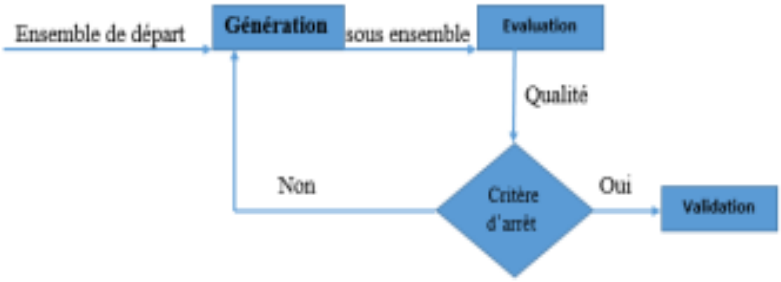


FIGURE 3.1 – Processus de sélection de variables

3.2.2.1 Génération Exhaustive (Complète)

Le problème majeur de l'approche exhaustive est que le nombre de combinaisons croît exponentiellement en fonction du nombre de caractéristiques. Pour un ensemble de N caractéristiques beaucoup plus grand, les combinaisons possibles rendent la recherche exhaustive impossible, [34].

3.2.2.2 Génération heuristique

L'approche heuristique ne garantit pas le résultat optimal, et les algorithmes qui utilisent cette approche sont généralement des algorithmes itératifs, dont chaque itération permet de sélectionner ou de rejeter une ou plusieurs caractéristiques. Les avantages de ces algorithmes simples à mettre en œuvre et rapides à produire des résultats. En revanche, ils ne permettent pas de parcourir totalement l'espace de recherche, [34].

Les trois sous catégories les plus utilisées sont : Sélection Ascendante (Forward), Élimination Descendante (Backward), et Sélection Progressive (Stepwise Selection).

3.2.2.2.1 Méthode Ascendante

Cette approche est également appelée ascendante, son principe est de commencer avec un ensemble de caractéristiques vide. Elle consiste en l'ajout à chaque étape la variable qui conduit à l'optimisation du critère de choix. Si aucune variable ne permet l'optimisation du critère de choix ou que toutes les variables sont intégrées, alors le processus s'arrête.

La catégorie ascendante cherche les meilleur variables et s'ajoute pas à pas à l'ensemble vide, au même temps élimine les variable les moins pertinente avec la soustraction

entre l'ensemble initial des variables et l'ensemble de meilleur variables jusqu'à une bonne qualité des variables sélectionnées ou un ensemble initial vide, alors on arrête. Nous cherchons donc une méthode permettant d'obtenir un modèle ne contenant que des variables significatives. C'est-à-dire que chaque variable retenue a un coefficient significatif sur la variable Y. Cette méthode examine un modèle avec une seule variable explicative puis introduction une à une d'autres variables explicatives.

Les étapes de choix d'un meilleur modèle pour la régression par la méthode ascendante sont :

Etape1 : Effectuer les k régressions possibles avec une seule variable explicative. Pour chacune d'elles, effectuer le test de Student. Retenir le modèle pour lequel la variable explicative est la plus significative.

Etape2 : Effectuer les $(k-1)$ régressions possibles avec deux variables explicatives. Pour chacune d'elles, effectuer le test de Student pour la nouvelle variable. Retenir le modèle pour lequel la variable est la plus significative. Si aucune variable n'est retenue, alors on stoppe le processus.

Sinon

Etape3 : Réitérer le processus en effectuant les $(k-2)$ régressions possibles avec trois variables explicatives. Pour chacune d'elles, effectuer le test de Student pour la nouvelle variable. Retenir le modèle pour lequel la variable est la plus significative. Si aucune variable n'est retenue, alors on stoppe le processus.

Sinon

Etape4 : Réitérer le processus en effectuant les $(k-3)$ régressions possibles avec quatre variables explicatives.

Le processus se termine lorsqu'on ne peut plus introduire des variables significatives dans le modèle. La méthode ascendante évite le travaille avec plus de variables que nécessaire, et améliore l'équation à chaque étape. Une variable introduite dans le modèle ne peut plus être éliminée et le modèle final peut alors contenir des variables non-significatives.

3.2.2.2.2 Méthode Descendante

Cette méthode est l'approche inverse et part de l'ensemble total des caractéristiques. Elle consiste au retrait à chaque étape de la variable qui conduit à l'optimisation du

critère de choix. Si aucun retrait ne permet d'optimiser le critère ou que toutes les variables ont été retirées, alors e processus s'arrête.

Il s'arrête soit quand il n'y a plus de variables à enlever soit quand la précision est jugée trop variables.

Les étapes de construction de modèle par la méthode descendante sont :

Etape 1 : Calculer la régression pour le modèle incluant toutes les k variables explicatives à disposition.

Etape2 : Effectuer un test de Student pour chacune des variables explicatives. Deux cas se présentent :

- Les variables sont trouvées significatives. Ce modèle est alors choisi. On stoppe là notre analyse.

- Éliminer la variable la moins significative du modèle.

Etape3 : Recommencer le processus avec une variable en moins.

La méthode descendante est très satisfaisante pour l'utilisateur préférant avoir toutes les variables possibles afin de ne rien ignorer. C'est une procédure plus économique en termes de temps et d'interprétation. Mais il y a un inconvénient majeur. Il n'est plus possible de réintroduire une variable une fois qu'elle a été supprimée. Le modèle final est donc un modèle au sein duquel toutes les variables sont significatives.

3.2.2.2.3 Méthode Progressive

Cette procédure est presque identique à la méthode ascendante, c'est une version hybride des deux précédentes car elle consiste à ajouter ou retirer successivement des variables à l'ensemble déjà sélectionné.

La première décision à prendre est donc le point de départ de la recherche, il peut être de trois sortes :

- Un ensemble vide : il s'agit de la méthode progressive ascendante (Forward Stepwise Selection) ;

- Un ensemble complet : il s'agit de la méthode progressive descendante (Backward Stepwise Elimination) ;

- Un ensemble d'attributs choisis aléatoirement.

La procédure progressive propose, après l'introduction d'une nouvelle variable dans le modèle, la réexamination des tests de Student pour chaque variable explicative anciennement admise dans le modèle. Après, si des variables ne sont plus significatives, alors retirer du modèle la moins significative d'entre elles. Le processus continue jusqu'à ce que plus aucune variable ne puisse être introduite ni retirée du modèle.

Ces méthodes permettent de pallier le problème de l'irrévocabilité de la suppression ou de l'ajout d'une variable, problème présent dans les deux autres directions de recherche. Ces méthodes autorisent l'ajout et la suppression d'une variable de l'ensemble des variables à n'importe quelle étape de la recherche autre que la première ou la dernière.

3.2.2.3 Génération aléatoire

La procédure de recherche aléatoire consiste à générer aléatoirement un nombre fini de sous- ensembles de caractéristiques afin de sélectionner le meilleur. En outre, les stratégies de recherche aléatoires convergent en général rapidement vers une solution semi-optimale, ce qui est préférable pour éviter le phénomène de sur-apprentissage, [42].

L'évaluation des sous-ensembles de variables permet d'évaluer les performances et la pertinence du sous-ensemble. Elle est traitée de façons très diverses tout en précisant le type d'approche utilisé et la fonction d'évaluation utilisée.

3.2.3 Critères d'arrêt et procédure de validation

Le critère d'arrêt permet à la procédure de sélection de variables de s'arrêter et le meilleur sous-ensemble serait toujours l'ensemble complet des variables de départ. La procédure de validation n'est pas une étape du processus de sélection de caractéristiques lui-même, mais une méthode de sélection d'attributs (en pratique) doit être validée. Elle consiste à tester la validité des sous-ensembles de caractéristiques sélectionnées avec la réalisation de différents tests, et la comparaison des résultats obtenus avec les résultats précédents, ou avec des résultats d'autres méthodes de sélection de caractéristiques en utilisant des données artificielles, réelles.

Les méthodes d'élimination des caractéristiques sont classées en méthodes supervisés et non-supervisés.

3.2.3.1 Méthodes supervisées

Les méthodes supervisées de sélection de variables sont classées généralement en deux groupes : les méthodes de Filtrage dont la mesure de pertinence est indépendante de l'algorithme d'apprentissage, qui utilise ensuite les données, et les méthodes d'Emballage (Wrapper) qui évaluent la pertinence du sous-ensemble de variables à l'aide des performances du système que l'on construit.

3.2.3.1.1 Filtrage

Pour les méthodes de Filtrage, le critère d'arrêt couramment utilisé est basé sur l'ordre des caractéristiques, rangées selon certains scores de pertinence (généralement des mesures statistiques). L'approche Filtre sélectionne un sous ensemble de variables en-prétraitement des données d'un modèle en fonction de différents critères calculés pour chaque caractéristique afin de mesurer la pertinence de la caractéristique.

Les procédures Filtres sont généralement moins coûteuses en temps de calcul puis qu'elles s'évitent les exécutions répétitives des algorithmes d'apprentissage sur déférents sous ensemble de variables.

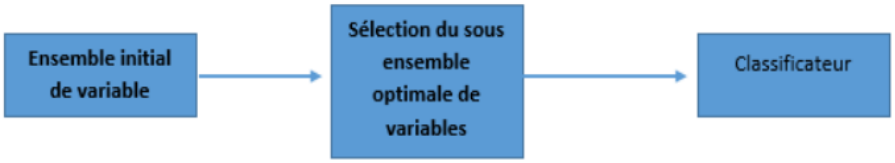


FIGURE 3.2 – Procédure de Filtrage

Les méthodes de Filtrages sont très rapides et de calculs simples, évolutifs, capables d'éviter les débordements, et indépendants du modèle d'apprentissage. Le principal avantage des méthodes de filtrage est leur efficacité calculatoire et leur robustesse face au sur-apprentissage. Malheureusement, ces méthodes ne tiennent pas compte des interactions entre caractéristiques et tendent à sélectionner des caractéristiques comportant de l'information redondante plutôt que complémentaire. De plus, ces méthodes ne tiennent absolument pas compte de la performance des méthodes. Les méthodes de Filtrage présentent un certain nombre d'inconvénients critiques qui affectent fortement leur performance par rapport à d'autres approches de sélection des fonctionnalités.

3.2.3.1.2 Emballage (Wrapper)

L'approche d'Emballage est la deuxième approche qui résout certains problèmes. Elle est appelée ainsi parce qu'elle est enveloppante autour de l'algorithme d'apprentissage. Le processus de recherche ici peut s'arrêter lorsqu'il n'y a plus d'amélioration de précision. Les Emballages ont été introduits par John et al. en 1994. Leurs principe est de générer des sous-ensembles candidats et de les évaluer grâce à un algorithme d'apprentissage. Cette évaluation est faite par un calcul d'un score. L'appel de l'algorithme d'apprentissage est fait plusieurs fois à chaque évaluation car un mécanisme de validation croisée est fréquemment utilisé.

Trois raisons font que les Emballages ne constituent pas une solution parfaite. D'abord, ils n'apportent pas vraiment de justification théorique à la sélection et ils ne nous permettent pas de comprendre les relations de dépendances conditionnelles qu'il peut y avoir entre les variables. D'autre part la procédure de sélection est spécifique à un algorithme d'apprentissage particulier et les sous-ensembles trouvés ne sont pas forcément valides si nous changeons de méthode d'induction.

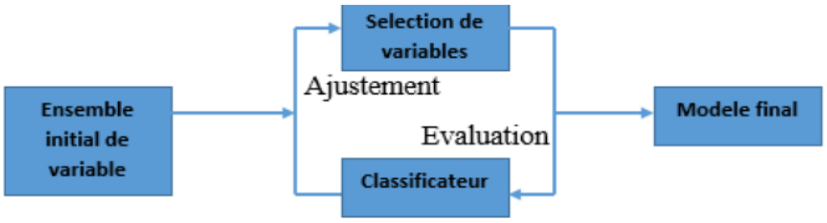


FIGURE 3.3 – Procédure d'Emballage

Les méthodes d'Emballage sont généralement considérées comme étant meilleures que celles du Filtrage, mais malheureusement plus long que les approches de Filtrage et d'autres approches de sélection de caractéristiques. Elles sont capables de sélectionner des sous-ensembles de caractéristiques de petite taille qui sont performants pour l'apprentissage utilisé.

3.2.3.2 Méthodes non-supervisées

Définir une mesure de pertinence pour l'apprentissage non-supervisé est un problème difficile car on ne dispose ni de valeur à prédire ni de classe à attribuer. En fonction des

mesures de similarité, les éléments redondants sont supprimés. On distingue généralement deux méthodes non-supervisées : Les méthodes Embarquées (Embedded) pour lesquelles la mesure de pertinence est directement incluse dans la fonction de coût optimisée par le système, et les méthodes de Régularisation.

3.2.3.2.1 Méthodes Embarquées (Embedded)

Contrairement aux méthodes supervisées, les méthodes Embarquées incorporent la sélection de variables lors du processus d'apprentissage qui est généralement automatique. Un tel mécanisme peut être trouvé, par exemple, dans les algorithmes de type SVM, AdaBoost, ou dans les arbres de décisions, [2], [18].

Les méthodes Embarquées tentent de combiner les avantages du Filtrage et d'Emballage. Ces méthodes ne sont pas chronophages, car ils évitent le recyclage de l'algorithme d'apprentissage que l'on voit dans les approches supervisées. Ils maximisent le rendement en interagissant avec le modèle d'apprentissage, comme dans l'approche d'Emballage. Le mélange de la sélection des fonctions pendant le processus d'apprentissage a des avantages à améliorer le coût de calcul, la précision d'apprentissage et aussi évite de former le modèle chaque fois qu'une nouvelle fonction est ajoutée. La sélection des caractéristiques pendant l'entraînement est efficace en termes d'utilisation des données, car il n'est pas nécessaire de diviser les données en ensembles d'entraînement et de validation.

Les méthodes Embarquées commencent par former le modèle d'apprentissage avec l'ensemble complet des caractéristiques, puis calculer un coefficient de corrélation pour chaque caractéristique. Ces coefficients sont utilisés pour classer l'importance des caractéristiques selon le modèle utilisé. Les valeurs élevées des coefficients reflètent une forte corrélation. L'approche Embarquée calcule un gain d'information pour chaque fonctionnalité comme dans l'apprentissage de l'arbre décisionnel, [2].

3.2.3.2.2 Régularisation

La régularisation, en général, est un large éventail de techniques d'apprentissage machine (AM). L'objectif de la méthode est de trouver le meilleur modèle qui minimise la fonction de perte ou de coût, dont le modèle correspond aux données, et qui maximise la performance,[11], [18].

La régularisation ajoute un autre critère à la fonction objective pour contrôler le niveau de complexité, [50], [23] comme suit :

$$L = E(Y_p, Y_a) + \lambda P(W) \tag{3.1}$$

tel que Y_p est la variable prédite, Y_a est la vrai variable, $E(.)$ est l'erreur de prédiction entre Y_p et Y_a ; W est le veteur des poids de l'élément x ; λ est le paramètre de régularisation contrôlant la complexité du modèle.

Le paramètre λ est utilisé pour contrôler le compromis entre la fonction objective et la pénalité. La pénalité est définie selon l'équation :

$$P(W) = \sum_i |w_i| \tag{3.2}$$

Lorsque la valeur λ change, la complexité du modèle change. L'objectif du paramètre de régularisation λ est de minimiser la perte L et de la maintenir à un minimum. Pour une très grande valeur de λ approchant ∞, les coefficients doivent être petits et s'approchent de zéro pour rendre la valeur totale aussi petite que possible.

Pour une valeur de $0 < \lambda < \infty$, il y aura quelques coefficients égaux à zéro, qui seront enlevés, mais pas beaucoup d'entre eux égaux à zéro. Il n'y a pas de valeur fixe pour λ et sa valeur peut être efficace calculé à l'aide de la validation croisée, [18].

Les plus populaires de ces méthodes sont la régression LASSO et RIDGE qui ont intégré des fonctions de pénalisation pour réduire le dépassement, [23]. La régression LASSO effectue une régularisation L_1 qui ajoute une pénalité équivalente à la valeur absolue de l'ampleur des coefficients. La régression RIDGE effectue une régularisation L_2 qui ajoute une pénalité équivalente au carré de l'ampleur des coefficients.

3.3 Critères de sélection

Dans la plupart des modèles statistiques en grande dimension (e.g. une régression linéaire multiple), on observe que la quantité d'information incluse est parfois superflue ou redondante, car la présence de nombreuses variables explicatives affaiblissent l'estimateur par moindres carrés à cause de l'explosion de sa variance. Donc, pour faciliter la sélection

du modèle statistique représentant le mieux possible la réalité, il existe plusieurs critères comme : le critère R^2 ajusté, l'erreur moyenne quadratique EQM, le Cp de Mallows, le critère de $PRESS$ de validation croisée, le F test, le critère d'information d'Akaïke AIC, et le critère d'information bayésien BIC.

L'objectif des critères de sélection de modèles est d'obtenir un modèle qui colle bien aux données tout en restant parcimonieux, c'est-à-dire avec un nombre limité de paramètres. En effet, on peut toujours améliorer l'ajustement d'un modèle en augmentant sa complexité. Une telle stratégie, appelée sur-ajustement, est contre-productif, car elle donne un modèle lourd avec des composantes difficiles à bien estimer. Le rôle d'un critère de sélection de modèles est donc de faire un compromis entre la qualité d'ajustement et la parcimonie du modèle. Selon McQuarrie et Tsai (1998), un bon modèle a des paramètres facilement interprétables et permet de bien prédire la variable d'intérêt, [23].

Le critère de sélection est utilisé pour arrêter la construction de la régression par étapes et également pour déterminer quand une variable déjà sélectionnée doit être supprimée du modèle. Leur but est obtenir un compromis satisfaisant entre un modèle trop simple (grand résidus) et un modèle faisant intervenir beaucoup de variables (donc très instable).

3.3.1 Coefficient de détermination R^2

Le coefficient de détermination R^2 est un moyen simple de tester l'aptitude linéaire d'un modèle, ayant une valeur entre zéro et l'unité. Plus la valeur est grand, plus le modèle correspond à un modèle de régression linéaire assez solide. Il est définit par :

$$R^2 = \frac{SCE}{SCT} = 1 - \frac{SCR}{SCT} \tag{3.3}$$

où :

$SCE = \|\hat{y} - \bar{y}\|^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$ est la somme des carrées expliquée par le modèle,

$SCT = SCE + SCR = \|y - \bar{y}\|^2 = \sum_{i=1}^{n}(y_i - \bar{y})^2$ est la somme totale des carrés,

$SCR = \|y - \hat{y}\|^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ est la somme des carrés des résidus,

$\hat{y}$ est la sortie estimée, $\bar{y}$ est la valeur moyenne des sorties y_i, et n est le nombre des observations.

Le modèle optimale, on utilisant le coefficient R^2, revient à choisir le modèle dont la

somme des carrés résiduelle SCR est la plus petite ou dont le R^2 est le plus grand. La somme des carrés résiduelle et le R^2 sont globalement de mauvais critères de sélection de variables.

Choisir entre deux modèles ayant le même nombre de variables explicatives exige l'utilité d'un test d'hypothèse sur l'augmentation de la significativité de la statistique R^2. Le R^2 augmente de façon monotone lorsqu'on ajoute une variable au modèle même si cette dernière est non-significative. Pour parer à cet inconvénient, il est conseillé de se tourner vers l'utilisation du coefficient de détermination ajusté R_a^2 qui est défini par :

$$R_a^2 = 1 - (\frac{n-1}{n-p}).(\frac{SCR}{SCT}) \tag{3.4}$$

où n est le nombre des observations, et p est le nombre de variables explicatives dans le modèle.

L'intérêt de la mesure du R_a^2 est très restreint car la protection qu'il offre contre le sur-apprentissage est extrêmement faible. En pratique son usage est donc déconseillé.

3.3.2 Erreur quadratique moyenne

L'erreur quadratique moyenne (EQM), est une mesure de la qualité d'un estimateur ou un prédicteur. Il est toujours un évaluateur proche de zéro et non-négatif.

$$EQM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5}$$

où n est le nombre de points de données,y_i est la sortie, et $\hat{y}_i$ est la sortie estimée.

L'EQM peut être écrit comme la somme de la variance de l'estimateur et le carré du biais de l'estimateur, fournissant un moyen utile pour calculer l'erreur quadratique moyenne. Ce qui implique que dans le cas des estimateurs sans biais, l'erreur quadratique moyenne et la variance sont équivalentes, [2].

$$EQM(\hat{y}_i) = Var(\hat{y}_i) + [Biais(\hat{y}_i)]^2 \tag{3.6}$$

L'objectif, ici, est de minimisé le EQM afin d'obtenir des prévisions aussi précises que possible.

3.3.3 Critère Cp de Mallows

Le Cp de Mallows s'inscrit dans un contexte de sélection de variables d'un modèle de régression linéaire. Il est utilisé lorsque l'objectif est de trouver le meilleur ensemble pour effectuer des prédictions. Il fournit une estimation de la capacité du modèle à prédire de nouvelles données. Il est une alternative à l'EQM.

Il existe une relation entre l'EQM et le Cp de Mallows, [2] donnée par :

$$Cp \simeq \sum_{i=1}^{n} \frac{EQM(\hat{y}_i)}{\sigma^2} \tag{3.7}$$

où n est le nombre de données d'observation, y_i est la sortie estimée, et σ^2 est la variance du modèle.

Supposons que le modèle de régression linéaire qui contient p variables sous sa forme complète, et soit aussi un sous modèle, appelé modèle réduit, contenant q variables ($q \leq p$). En utilisant la somme des carrés des résidus (SCR), donnée précédemment, on montre que le Cp de Mallows de ce sous modèle donne, [2] :

$$Cp \simeq q + \frac{(S_q^2 - \sigma^2)(n-q)}{\sigma^2} \tag{3.8}$$

avec

$$S_q^2 = \frac{\sum_i \sum_j (\hat{y}_i - y_i)^2}{n-q} = \frac{SCR}{n-q} \tag{3.9}$$

est le carré résiduel moyen du modèle à q paramètres, et $\hat{y}_i$ est la prédiction de y_i par le modèle réduit à q variables.

Si σ^2 est inconnu on le remplace par son estimateur $\hat{\sigma}^2$ représentant le carré résiduel moyen du modèle complet.

Dans ce cas, le Cp de Mallow devient :

$$Cp = \frac{SCR}{\hat{\sigma}^2} - n + 2q \tag{3.10}$$

Si le modèle ajuste bien les données alors $SCR \approx (n - |p|)\sigma^2$, et donc $Cp \approx |p|$.

Le meilleur modèle d'après le critère Cp de Mallows est celui dont la composante $(Cp - p)$ est la plus petite possible, c'est-à-dire le Cp est plus proche de p. Donc, Choisir

un modèle qui minimise le critère du Cp dans ces conditions revient alors à choisir un modèle qui en moyenne à une erreur quadratique moyenne minimale.

Un modèle qui ajuste mal les données aurait un Cp beaucoup plus grand que $|p|$.

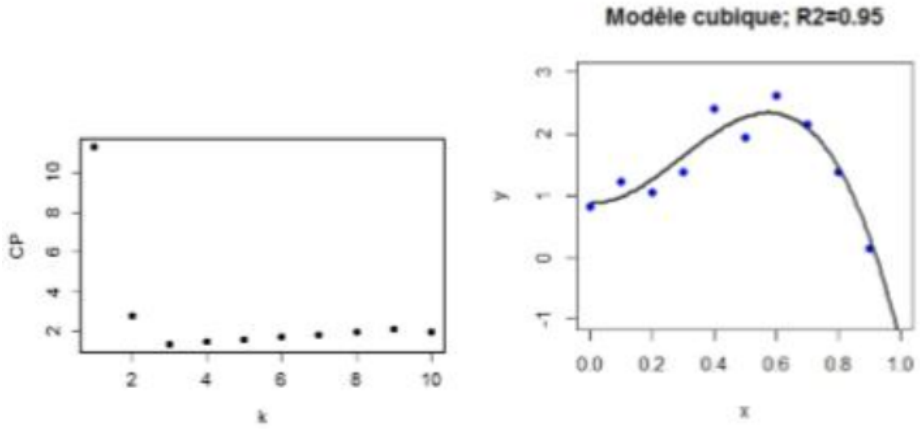


FIGURE 3.4 – Cp de Mallows en fonction du degré du polynôme et un modèle sélectionné de degré 3.

La figure 3.4 montre le comportement du Cp dans un exemple d'une régression polynomial. Ce critère décroît avec le biais jusqu'à un choix optimal de dimension 3 avant de ré-augmenter avec la variance.

3.3.4 Critère de validation croisée

La validation croisée est une méthode statistique qui partitionne les données en deux groupes. Un pour l'entraînement et l'autre pour la validation du modèle. C'est un des moyens les plus efficaces pour juger la qualité d'un modèle. Le principe de la validation croisée est d'estimer les paramètres à partir d'un sous-échantillon des données et d'évaluer leurs performances de prédiction sur les données mises a côté.

La version la plus simple est le critère $PRESS$, (prédiction de la somme des carrés), suggérée par Allen (1974), pour lequel une seule observation est laissée a côté.

Soit $\hat{y}_m^{(i)}$ la prédiction de y_i estimée dans le modèle m, à partir des données sans le y_i. Le critère $PRESS$ est défini par :

$$PRESS_m = \sum_{i=1}^{n} (\hat{y}_m^{(i)} - y_i)^2 \tag{3.11}$$

Le $PRESS$ représente la somme des erreurs de prévision et il mesure la capacité d'un modèle donné à bien prédire les nouvelles observations. Ainsi, plus le critère est faible, plus le modèle prédit est parfait.

Il existe heureusement un théorème qui permet d'obtenir le résidu $PRESS$ sans avoir à effectuer les n régressions.

Théorème 5. *Soit $H = X(X^TX)^{-1}X^T$ la matrice chapeau, [23], associée au modèle $Y = X\beta + \varepsilon$. Soit $\hat{Y} = YH$ le vecteur des valeurs prédites par le modèle.*

Le critère $PRESS$ peut encore s'écrire comme suit :

$$PRESS(m) = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n_i}\frac{(\hat{y}_{ij} - y_{ij})^2}{(1 - h_{(ij)(ij)})^2} \tag{3.12}$$

où $h_{(ij)(ij)}$ représente l'élément diagonal de la matrice H situé au rang (k, k), avec

$$\begin{cases} k = j & si \quad i = 1 \\ k = \sum_{l=1}^{i-1} n_i + j & si \quad i = 2, \cdots, m. \end{cases} \tag{3.13}$$

Dans une famille de modèles, celui qui possède les meilleures capacités prédictives est celui avec un $PRESS$ minimal. Si les $h_{(ij)(ij)}$ sont égaux, le critère $PRESS$ est proportionnel à la somme des carrés résiduels SCR, (voir l'equation 2.2). Ce critère produit en général de très bons résultats.

Le critère PRESS permet d'éviter les modèles en situation de sur-ajustement. En effet, les modèles en sur-ajustement ont tendance à avoir des petits résidus pour les données incluses dans le jeu de donnée d'entraînement et des grands résidus pour les données externes. Or le critère PRESS, par définition utilise les données qui n'ont pas servi à l'estimation du modèle. Par conséquent, un modèle sur-ajusté aura des grands résidus et donc un critère PRESS élevé. Le principal inconvénient de la validation croisée est son temps de calcul, qui peut parfois être prohibitif.

Plusieurs types de validation croisée sont décrient dans la suite.

3.3.4.1 Méthode d'attente ou Hold-out

La première méthode est très simple à mettre en œuvre, il suffit de diviser l'échantillon en deux parties, une pour estimer le modèle et l'autre pour estimer l'erreur de prévision.

Le modèle est bâti sur l'échantillon d'apprentissage et validé sur l'échantillon de test. L'erreur est estimée en calculant l'erreur quadratique moyenne. Un seul ajustement à exécuter par modèle.

La taille de l'échantillon d'apprentissage ($\frac{n}{2}$ ou $\frac{3n}{4}$) rend l'estimation moins précise que si on utilisait les n observations. Cela conduit à une sous-estimation de l'erreur de test par rapport à celle issue d'une estimation sur n individus. L'évaluation sur un seul échantillon de test rend l'estimation de l'erreur très variable (la variance est élevée). Pour minimiser ce dernier inconvénient, on peut recommencer la procédure Hold-out plusieurs fois (une dizaine de fois), en considérant plusieurs séparations aléatoires de l'échantillon, toujours de même taille. Le risque final, obtenu par la moyenne des risques issus de chaque découpage, est alors moins variable. Cette généralisation fait néanmoins perdre les avantages initiaux, en particulier le coût est supérieure puisqu'il faut estimer et évaluer chaque modèle autant de fois qu'il y a de découpages déférents, [25].

3.3.4.2 Méthode d'absence ou Leave-one-out (LOO)

La validation croisée Leave-one-out consiste à mettre de côté une observation, (on apprend sur $(n-1)$ autres observations), puis calculer le risque de prévision sur l'observation mise a côté, [11]. On répète cette opération n fois, de telle sorte que n erreurs de prévision sont finalement calculées. Leur moyenne est une estimation de l'erreur de test.

De façon plus formelle $\hat{y}_i^{(-i)}$ est la prévision de la sortie y_i à partir du modèle estimé sur toutes les observations sauf i. Le risque estimé par LOO est :

$$CV_{LOO} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{(-i)})^2 \tag{3.14}$$

La méthode LOO peut-être très coûteux à mettre en œuvre, car il nécessite en toute généralité n ajustements de chaque modèle testé. Il y a une exception notable : dans le cas d'une régression linéaire, CV_{LOO} s'exprime directement grâce à la seule estimation du modèle sur les n observations.

Tous les échantillons d'apprentissage sont très semblables (ils différent que par une observation) et donc le risque final par LOO ne moyenne pas suffisamment de situations différentes. En conséquence l'estimation du risque a une forte variance.

3.3.4.3 Méthode de sortie ou Leave-k-out

L'idée est la même que la méthode précédente, sauf qu'au lieu de mettre de côté une seule observation, on en met k et on calcule le risque de prévision sur les k observations mises de côté, [50]. On répète le processus pour tous les découpages possibles, il y en a $\binom{k}{n} = \frac{n!}{k!(n-k)!}$, et on moyenne tous les risques pour obtenir l'estimation finale.

Si $k = 1$, on retrouve évidemment le LOO. Pour k plus grand, l'estimation du risque est un peu plus biaisé qu'avec le LOO (car on ajuste le modèle sur moins d'observations), mais la variance est plus faible car les échantillons d'apprentissage sont plus diverses. Ainsi il existe un choix de k garantissant un bon compromis biais-variance (ce choix dépend de n et le consensus est de le prendre de l'ordre de $n/10$ ou $n/5$). Cependant, le défaut majeur de cette démarche est son coût : elle requiert d'estimer le modèle sur un nombre prohibitif d'échantillons d'apprentissage.

Cette méthode donne une bonne estimation du risque test pour un choix de k approprié, mais elle est généralement impossible à mettre en œuvre en pratique.

3.3.4.4 Méthode du k-plis ou k-fold

La version la plus couramment utilisée pour la validation croisée est k-fold. On divise l'échantillon en k parties égales (ou k est un nombre spécifiée par l'utilisateur, généralement 5 ou 10). Puis on apprend le modèle en utilisant $k-1$ parties de l'échantillon et on le test sur la partie restante, et on calcule l'erreur quadratique moyenne. En répétant cette opération pour tous les cas possibles, k fois pour qu'en fin de compte chaque sous-échantillon est utilisé exactement une fois comme ensemble de validation. La moyenne des k erreurs quadratiques moyennes est enfin calculée pour estimer l'erreur de prédiction, [24].

Lorsque $k = n$, il s'agit de la procédure de sortie LOO.

Lorsque $k = 2$, cela ressemble à une procédure d'attente (Hold-out) dans laquelle l'échantillon d'apprentissage et l'échantillon test ont même taille. Ce n'est cependant pas tout à fait équivalent car ici on inverse le rôle joué par chaque partie pour fournir deux estimations du risque que l'on moyenne, contrairement à la méthode d'attente (Hold-out) où une seule estimation du risque est effectuée.

Si k est choisi trop grand, il y aura un faible biais car les échantillons d'apprentissage auront une taille très proche de n, mais par contre l'estimation du risque souffrira d'une

grande variance comme pour la méthode *LOO*. Le désavantage majeure dans ce cas est similaire au *LOO* : le calcul peut-être très long car il faut estimer k modèles.

A l'inverse, si k est petit, le calcul est rapide, la variance d'estimation est moindre, car on moyenne des situations où les échantillons sont très différents. Mais l'estimation risque d'être biaisée car la taille de l'échantillon d'apprentissage est beaucoup plus petite que la vraie taille n.

Cette méthode donne une bonne estimation du risque de test pour un choix de k approprié ($k = 5$ à $k = 10$). Elle est relativement rapide à mettre en œuvre pour les choix de k précédents. L'estimation du risque ici n'est pas aussi bonne qu'avec une procédure de sortie (leave-k-out) optimale.

3.3.5 Critère F-test

Un test F est tout test statistique a une distribution F sous l'hypothèse nulle. Il est le plus souvent utilisé pour comparer les modèles statistiques qui ont été adaptés à un ensemble de données, [23]. Les F-tests exacts se produisent principalement lorsque les modèles ont été ajustés aux données en utilisant les moindres carrés. Le nom a été inventé par George W. Snedecor, en l'honneur de Sir Ronald A. Fisher, 1920.

La statistique F est simplement un rapport de deux variances qui mesure la dispersion, ou la distance à laquelle les données sont dispersées par rapport à la moyenne. Pour éviter de surcharger le modèle, (avoir une plus grande dispersion), il est suggèré d'utiliser la valeur α de $0,01$ ou $0,02$ pour un test F, plutôt que le choix classique de $0,05$.

On distingue en général deux hypothèses seulement : l'hypothèse nulle, dont la différence est considérée comme nulle, et l'hypothèse alternative ou complémentaire.

Le test F est utilisé dans l'analyse de régression pour tester l'hypothèse que tous les paramètres du modèle sont nuls. Il est également utilisé dans l'analyse statistique pour comparer les modèles statistiques qui ont été ajustés en utilisant les mêmes facteurs sous-adjacents et l'ensemble de données pour déterminer le modèle le mieux adapté. C'est à dire :

$$\begin{cases} H_0 : \beta_1 = \beta_2 = ... = \beta_k = 0, \\ H_1 : \exists \beta_j \neq 0, \forall j = 1 : k \end{cases} \tag{3.15}$$

où : H_0 est l'hypothèse nulle, H_1 est l'hypothèse alternative, et β_k sont les paramètres du

modèle.

La statistique F est simplement :

$$F = \frac{S_1^2}{S_2^2}, \tag{3.16}$$

où S_1^2 et S_2^2 sont les variance des échantillons 1 et 2, respectivement, données par :

$$S_i^2 = \frac{\sum_i (x_i - \bar{x_i})^2}{(n_i - 1)}, i = 1, 2. \tag{3.17}$$

où (x_i) sont les valeurs de l'échantillon i et $(\bar{x}_i)$ est leurs moyennes resperctives.

Une fois les hypothèses de test posées, nous devons choisir la statistique de test F. C'est en comparant la valeur de cette statistique observée dans l'échantillon à sa valeur sous l'hypothèse H_0 que nous pourrons prendre une décision. En essayant de prendre des décisions, nous commençons toujours par préciser l'hypothèse nulle par rapport à l'hypothèse alternative. La valeur calculée du test F avec sa valeur p associée est utilisée pour déduire s'il faut accepter ou rejeter l'hypothèse .

Nous comparons la valeur p du test F au niveau de signification alors :

- Si $p < 0.05$, nous disons que le test est significatif à 5% et nous pouvons rejeter l'hypothèse nulle et accepter l'hypothèse alternative.

- Si $p > 0,05$, nous disons que le test est n'est pas significatif à 5% et nous devrions accepter l'hypothèse et rejeter .

- Si $p < 0,01$, dans ce cas, nous disons que le critère est important à 1% (très forte preuve contre H_0).

3.3.6 Critère d'information Akaike

Le critère d'information Akaike (AIC) s'applique aux modèles estimés par une méthode de maximum de vraisemblance : les analyses de variance, les régressions linéaires multiples, les régressions logistiques, et la loi de poisson peuvent rentrer dans ce cadre.

On appelle fonction de vraisemblance pour un échantillon observé $(x_1, \cdots, x_n)$ la fonction de θ :

$$L(x_1, ..., x_n, \theta) = \prod_{i=1}^{n} f(x_i, \theta) \tag{3.18}$$

où

$$f(x_i, \theta) = \begin{cases} p_\theta(X = x) & si \quad X\text{est une v.a. discrète} \\ f_\theta(x) & si \quad X\text{est une v.a. à densité} f \end{cases} \tag{3.19}$$

Le critère d'information d'Akaike s'écrit comme suit :

$$AIC = 2k - 2.log(L) \tag{3.20}$$

où k est le nombre de paramètres à estimer du modèle et L est le maximum de la fonction de vraisemblance du modèle.

L'AIC représente un compromis entre le biais, (diminuant avec le nombre de paramètres), et la parcimonie, volonté de décrire les données avec le plus petit nombre de paramètres possible.

Le meilleur modèle est celui possédant l'AIC le plus faible. Quand le nombre de paramètres k est grand par rapport un nombre d'observation n, ($\frac{n}{k} < 40$), il est recommandé d'utiliser l'AIC corrigé :

$$AIC_c = AIC + \frac{2k(k+1)}{n-k-1} \tag{3.21}$$

L'utilisation de l'AIC est assez fréquente et cet indicateur est polyvalent. Grâce à lui, on évalue la bonne adéquation d'un modèle et surtout on peut comparer plusieurs modèles entre eux. Ce critère permet par exemple d'évaluer des régressions multiples (à l'instar du R^2 ajusté), des prévisions sur séries chronologiques ou encore des régressions logistiques, [23].

3.3.7 Critère d'information bayésien

Le critère BIC est une approximation du calcul de la vraisemblance des données conditionnellement du modèle fixé. C'est une méthode alternative à l'AIC. Elle pénalise davantage les modèles plus grands, visant pour éviter le problème de débordement. C'est un critère de sélection de modèles parmi un ensemble de modèles finis. Le modèle avec le BIC le plus bas est préféré.

Le critère d'information bayésien est défini par :

$$BIC = -2log(L) + klog(L), \tag{3.22}$$

avec L la vraisemblance du modèle estimée, n est le nombre d'observations dans l'échantillon et k est le nombre de paramètres libres du modèle. Le modèle qui sera sélectionné est celui qui minimise le critère BIC, [23].

L'une des difficultés du critère BIC est son interprétation. Les notions de probabilité a priori ou a posteriori d'un modèle sont peu explicites et ne donnent pas une idée intuitive de ce que le BIC est considéré à être un bon modèle ou non.

CHAPITRE 4

INITIALISATION DES RÉSEAUX DE NEURONES

4.1 Introduction

Un réseau de neurones (RN) est un approximateur universel qui permet d'exprimer au mieux la corrélation entre les données d'entrée et les données de sortie. L'une des principales tâches des RNs est leurs apprentissage à partir des données qui améliorent leurs performances, [57], [68].

L'apprentissage est défini comme un processus par lequel les paramètres libres du RN sont adaptés, [42]. Le rôle de tout processus d'apprentissage est d'utiliser l'ensemble d'exemples pour faire varier les valeurs des paramètres, depuis des valeurs initiales arbitraires jusqu'à atteindre des valeurs finales, [45], [8], [3]. Ces valeurs finales devraient, en principe, attribuer au réseau un comportement optimal, conformément à certains critères préétablis. Dans la grande majorité des méthodes neuronales, les ajustements de paramètres sont effectués de manière itérative, [7], [32], [68].

Dans le cadre théorique, il a été prouvé que les RNs à propagation avant sigmoïdes/linéaires à deux couches sont capables d'approcher n'importe quelle fonction arbitraire, étant donné qu'ils ont un nombre suffisant de neurones dans leurs couches cachées, [8], [68], [3], [38]. Les performances et la complexité des RNs augmenteront avec l'augmentation du nombre de neurones cachés, [17]. Mais en pratique, il n'est pas possible de connaître à l'avance le nombre de neurones cachés. Il faut présupposer un certain nombre et lancer l'entrainement, [68], [38]. Même si ce nombre serait le bon, il reste le risque de voir que l'entraînement

prendre trop de temps pour converger vers la solution désiré, [7], [46]. Le RN peut ne pas converger du tout, à cause de la disparition du gradient, [45], [57], [42], [68], [3]. Ces écueils sont dus à la procédure même de l'entraînement. Pour atteindre les valeurs finales des paramètres, il faut d'abord partir des valeurs initiales, [46], [3], [38]. Cependant, comment choisir ces valeurs initiales ?

Au début des méthodes neuronales, les chercheurs ont convenu que les valeurs initiales peuvent être générées par une loi Uniforme sur un petit intervalle centré sur zéro, [1], [46]. La proximité de cette valeur à zéro semble se traduire par de meilleurs taux de convergence, [46], [68]. Sans information préalable, il n'y a aucune raison de privilégier certaines valeurs par rapport à d'autres. Cependant, ce mode de sélection ne conduit pas nécessairement à des situations propices à une bonne entraînement du réseau. Les valeurs finales ciblées des paramètres peuvent être très éloignées des valeurs initiales aléatoires, [33], [46].

Généralement, les deux critères utilisés pour évaluer la qualité de l'entraînement sont le temps mis pour l'exécuter et la qualité du score obtenu par le réseau formé au regard de certaines fonctions de coût, [42], [17]. Cependant, il est bien connu qu'avec le même algorithme et le même ensemble d'entraînement, la répétition de l'entraînement ne prend pas la même durée ni ne fournit le même score, [57], [46], [32]. Certains facteurs, qui changent nécessairement d'une manipulation à l'autre, provoquent ces différences, [45], [3]. Le premier de ces facteurs est le choix des valeurs initiales des paramètres avant l'entraînement, [8], [46], [4], [38].

A l'avènement des méthodes neuronales, le principe admis est de donner de petites valeurs aléatoires initiales avant de commencer l'entraînement. Par la suite, très rapidement, l'initialisation des RNs est devenue un problème central et a été étudiée par de nombreux chercheurs, [1], [54], [7], [8], [57], [4], [38], etc. Cette question est toujours d'un grand intérêt. Périodiquement, depuis le début des années 90 jusqu'à aujourd'hui, de nouvelles méthodes sont proposées pour traiter des situations générales ou pour se concentrer sur des aspects particuliers. De nombreux travaux ont été développés sur la question.

La méthode de Nguyen et Widrow,[40], bien qu'ancienne, s'est imposée comme une méthode de référence. Elle est largement reconnu et est le plus utilisé, [1], [7], [8], [46], [38], etc. Nguyen et Widrow ont montré, dans le cas d'un réseau sigmoïde/linéaire à deux

couches, que les poids et les biais des valeurs initiales lorsqu'ils sont générés avec certaines contraintes produisent un meilleur réseau pour approximer une fonction arbitraire, [8], [68]. Il est maintenant bien établi que l'utilisation de la méthode Nguyen-Widrow entraîne souvent une diminution du temps d'entraînement de façon très appréciable, [8], [38].

Malheureusement, les avantages de la méthode Nguyen-Widrow ne sont pas toujours garantis. Il est fréquent de constater qu'en répétant la même expérience, avec des conditions apparemment identiques, certains essais donnent des résultats satisfaisants mais d'autres sont décevants, en termes de temps d'apprentissage, mais aussi en termes de précision des estimations, [1], [68]. Pour cette raison, il a toujours été recommandé de faire plusieurs tentatives d'entraînement et de retenir celui qui aurait produit les meilleurs résultats, [1], [46]. La table 4.1 montre comment l'échec de l'apprentissage survient insidieusement dans certaines manipulations plutôt que dans d'autres.

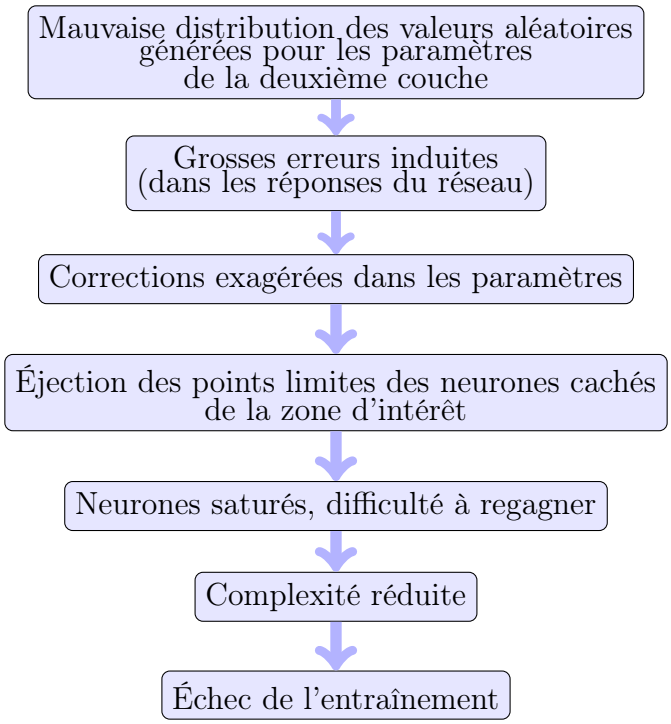


TABLE 4.1 – Organigramme du mécanisme d'échec d'entraînement

Le but de ce travail est de révéler les raisons insidieuses qui causent les essais infructueux et empêchent de tirer pleinement parti de la méthode Nguyen-Widrow. Il révèle l'existence d'un défaut, imperceptible lors de l'initialisation, mais qui s'installe au tout début de l'entraînement, ce qui aura un impact négatif sur la qualité de l'entraînement,

en terme de temps d'exécution, [46], et également en terme des performances, [1].

Notre analyse donne les mesures préventives qui garantiraient un fonctionnement correct et permanent de la méthode de Nguyen-Widrow. Ces mesures éliminent au préalable les situations qui conduiraient à un échec de l'entraînement. Ainsi, la qualité de tout réseau ne dépend que de son architecture. La détermination du nombre optimal de neurones cachés est ainsi grandement facilitée. Nous montrons également que, dans un contexte de régression, les situations de sur-ajustement sont plus facilement détectables par cette méthode.

Dans ce Chapitre nous donnant d'abord quelques méthodes d'initialisation des poids, suivie par une description générale de la méthode d'initialisation de Nguyen-Widrow et leur inconvénient dans la Section 2. La méthode proposé pour amélioré l'efficacité de la méthode de Nguyen-Widrow est présenté dans la Section 3, et dans la Section 4 nous exposons les résultats des manipulations produit pour montrer la performance de notre moyen d'amélioration proposé dans la Section précédente.

4.2 Quelques méthodes d'initialisation des poids

4.2.1 Méthode d'initialisation aléatoire

La méthode de génération aléatoire des poids est la stratégie la plus employée pour l'initialisation des poids. Les poids sont généralement initialisés avec de petites valeurs de distribution aléatoire Uniforme à l'intérieur d'un petit intervalle centré en zéro, par exemples, dans l'intervalle $[-0.1, 0.1]$. Ce fait semble paradoxal car nous laissons un sujet important au hasard.

Cependant, il est également bien connu que les valeurs d'initialisation particulières influencent les propriétés de la vitesse de convergence, et les performances de généralisation. Du point de vue de la vitesse de convergence, il est clair qu'une initialisation proche d'un minimum local supposera un trieur en nombre d'itérations et en temps de convergence, [15].

De plus, l'initialisation détermine le minimum local qui sera atteint après convergence, si l'erreur de ce minimum est trop élevée on considérera que le réseau de neurones n'a pas convergé. Enfin, le minimum local atteint, même en cas de convergence, influencera

les performances de généralisation. Compte tenu des arguments ci-dessus, nous pouvons facilement conclure que l'initialisation est un sujet très important.

4.2.2 Méthode de Chih-Liang Chen et Nutter Roy. S. 1991

En 1991, Chih-Liang Chen et Nutter Roy S. ont proposé d'attribuer différentes forces de connexion initiales en fonction de l'importance de chaque entrée. Une entrée plus importante doit être initialisée avec des poids plus importants. En particulier, les auteurs n'ont effectué des simulations qu'avec un problème de 8 entrées et ils ont initialisé le plus important au hasard dans l'intervalle $[0.5, 1]$, le moins important dans $[0, 0.5]$ et le reste dans $[0, 1]$. Le critère pour mesurer l'importance d'une entrée était la méthode d'élagage basée sur la sensibilité (SBP) des variables d'entrée, [62].

L'algorithme SBP calcule une mesure de sensibilité S_i pour évaluer le changement d'erreur d'apprentissage qui résulterait si l'entrée x_i était supprimée du réseau. La sensibilité du modèle de réseau à la variable i est définie comme :

$$S_i = \frac{1}{N} \sum_j S_{ij} \tag{4.1}$$

où S_ij est la sensibilité calculée pour l'exemplaire x_j.

Comme il y a généralement beaucoup moins d'entrées que de poids, une évaluation directe de S_i est possible :

$$S_ij = E(\bar{x}_i, w_\lambda) - E(x_{ij}, w_\lambda) \tag{4.2}$$

$$\bar{x}_i = \frac{1}{N} \sum_{j=1}^{N} x_{ij}$$

où S_i mesure l'effet sur l'erreur quadratique d'apprentissage, E, du remplacement de la i^{me} entrée x_i par sa moyenne $\bar{x}_i$ pour tous les exemplaires. Le remplacement d'une variable par sa valeur moyenne supprime son influence sur la sortie du réseau.

Dans le calcul de S_i, aucun réapprentissage n'est effectué dans l'évaluation de l'erreur quadratique d'apprentissage $E(\bar{x}_i, w_\lambda)$. Il n'est pas suffisant de définir $x_{ij} = 0, \forall j$, car la valeur du biais de chaque unité cachée a été déterminée pendant l'apprentissage et ne

serait pas correctement compensée en réglant arbitrairement l'entrée à zéro.

Si les entrées sont normalisées pour avoir une moyenne nulle avant l'apprentissage, alors définir une variable d'entrée à zéro revient à la remplacer par sa moyenne.

4.2.3 Méthode de Drago, G.P. et Ridella, S. 1992

Cette méthode est appelée SCAWI et elle a été proposée par Drago, G.P. et Ridella, S. en 1992, [9]. Les auteurs utilisent le concept de « Pourcentage de Neurones Paralysés » (PNP). Ce concept peut être défini en testant combien de fois un neurone est dans un état saturé et l'amplitude d'au moins une erreur de sortie est élevée. Ils proposent les équations suivantes, [26], pour calculer la valeur des poids.

$$w_{ij}^{entr} = \frac{1.3}{\sqrt{1 + N_{entr} \cdot \nu^2}}.r_{ij} \tag{4.3}$$

$$w_{ij}^{cach} = \frac{1.3}{\sqrt{1 + 0.3 \cdot N_{cach} \cdot \nu^2}}.r_{ij} \tag{4.4}$$

Dans les équations, ν est la valeur quadratique moyenne des entrées et r_{ij} est un nombre aléatoire uniformément distribué dans l'intervalle $[-1, +1]$.

4.2.4 Méthode de Li et al. 1993

Li, G., Alnuweiri, H., et Wu, Y. ont proposé une autre méthode qui diffère beaucoup de l'initialisation classique. Tout d'abord, le réseau multicouche est partitionné à chaque couche cachée en Perceptrons simples. Après cela, les poids des Perceptrons sont initialisés avec des valeurs nulles et il est effectué un apprentissage des Perceptrons indépendant en utilisant la règle Delta, [36]

$$\Delta W_{ji}(t+1) = \eta \cdot \delta_{pj} \cdot I_{pj} + \alpha \cdot W_{ji}(t) \tag{4.5}$$

$$\delta_{pj} = (D_{pj} - O_{pj}) \cdot f'(R_{pj}) \tag{4.6}$$

où : W_{ji} est le poids du $i^{\text{ème}}$ neurone d'entrée au $j^{\text{ème}}$ neurone caché, η est le taux d'apprentissage, I_{pj} est le $i^{\text{ème}}$ élément du modèle d'entrée p, α est le Momentum, D_{pj} est la

sortie souhaitée du $j^{\text{ème}}$ neurone de sortie correspondant au modèle d'entrée p, O_{pj} est la sortie du $j^{\text{ème}}$ neurone de sortie correspondant au modèle d'entrée p, f' est la dérivé de la fonction logistique standard f, et R_{pj} est l'entrée nette au neurone caché j correspondant au modèle d'entrée p.

Supposons le cas d'un réseau à propagation avant à une couche cachée, nous obtiendrons deux Perceptrons après la partition. Les entrées du premier Perceptron sont les entrées de l'ensemble d'apprentissage, mais nous devons fixer une valeur pour ses cibles.

Les auteurs ont proposé quatre méthodes pour choisir les cibles, [36]. Les deux premiers sont pour le cas où $N_{pal} < 2^{N_{cach}}$, N_{cach} est le nombre d'unités cachées et N_{pal} est le nombre d'échantillons dans l'ensemble d'apprentissage. Le reste est pour l'autre cas.

Nous devons choisir trois partenaires pour utiliser cette méthode, l'étape d'apprentissage η, le Momentum α de la règle Delta et le nombre d'itérations que ce pré-entraînement sera effectué.

Le cœur de la méthode de pré-traitement Delta (PTD) est que les poids initiaux pour les réseaux de rétro-propagation sont pré-traité avec la règle Delta, au lieu d'utiliser des poids aléatoires comme d'habitude. Pour réaliser le PTD, deux étapes doivent être franchies.

Premièrement, le réseau multicouche est partitionné à chaque couche cachée en Perceptrons. Ainsi, la règle Delta peut être démarrée avec un réglage de poids nul pour chaque Perceptron.

Deuxièmement, les modèles cachés doivent être définis de telle sorte que les sorties du Perceptron précédent soient les entrées du Perceptron actuel. Cela établit des connexions entre les Perceptrons adjacents.

Pour mettre en œuvre la deuxième étape, nous proposons les quatre méthodes présentées ci-dessous. Selon le problème à résoudre et la taille du réseau, l'un d'entre eux peut être utilisé, :

Méthode 1 : Avec suffisamment de neurones cachés, c'est-à-dire $2^j \geq p$, attribuez séquentiellement un motif caché parmi 2^j modèles possibles à l'un des modèles d'entrée, où J est le nombre de neurones cachés. Un modèle caché correspond à un modèle d'entrée. Cette méthode est adaptée à une implémentation matérielle.

Méthode 2 : Avec suffisamment de neurones cachés, c'est-à-dire $2^j \geq p$, attribuez

au hasard un modèle caché parmi 2^j modèles possibles à l'un des modèles d'entrée, où J est le nombre de neurones cachés. Un modèle caché correspond à un modèle d'entrée. Cette méthode est utile dans le cas où la méthode 1 n'est pas appropriée pour certains problèmes.

Méthode 3 : Avec moins de neurones cachés, c'est-à-dire $2^j < p$, attribuez au hasard un modèle caché à l'un des modèles d'entrée. Un modèle caché peut correspondre à des modèles à entrées multiples.

Méthode 4 : Initialisez aléatoirement les poids de la couche inférieure. Calculez ensuite les modèles cachés en présentant chaque modèle d'entrée. Cette méthode peut être utilisée lorsque $2^j < p$.

Étant donné les modèles cachés, la règle Delta, 4.5, 4.6, est utilisée pour pré-entraîner la couche caché-à-sortie et/ou l'entrée-à-caché couche. Le nombre d'itérations avec la règle Delta est une variable réglable. Dans Les expériences les auteurs l'ont défini dans la plage de $5p$ à $10p$. Le temps passé sur la règle Delta est supplémentaire à l'algorithme BP. Le nombre de présentations de motifs pendant la phase peut être utilisé pour estimer le temps pris par le DPT. Après DPT, les réseaux Perceptron, ainsi que les poids de PTD, sont combinés à la structure du réseau multicouche d'origine. Les poids pré-entrénés dans chaque Perceptron sont utilisés comme poids initiaux dans l'entraînement de rétro-propagation suivie.

4.2.5 Méthode de Shimodaira 1994

Ksashi Shimodaira a proposé, en 1994, une autre méthode basée sur des équations qui représente les caractéristiques du mécanisme de transformation de l'information d'une unité, [51]. L'idée est que la largeur de la région d'activation de la fonction sigmoïde dans un nœud, doit être plus large que la taille de la ligne diagonale de l'unité cube et les centres de la région d'activation et l'unité cube doivent être coïncide à chacune des autres.

La méthode peut être résumée par l'algorithme suivant, qui détermine les poids w_i , reliés a des unités de la couche inférieure au numéro d'unité i avec n poids d'entrée.

1) Calculez b par :

$$b = |f^{-1}(1-\varepsilon) - f^{-1}(\varepsilon)| \tag{4.7}$$

où f est la fonction de transfert dont $u = \sum_{i=1}^{n} w_i \cdot x_i + w_0$ est l'entrée nette du réseau, et ε est un paramètre, dont sa valeur appropriée semble être 0.1.

2) En utilisant n et le paramètre k, calculez $\hat{w}_j$ avec :

$$\hat{w} = \frac{b}{k - n} \tag{4.8}$$

où n est le nombre des entrée dans la couche inférieurs, et k est un paramètre à fixé préalablement pour contrôler la largeur de la région d'activation et donc la taille du vecteur des poids, [51].

3) En utilisant un autre paramètre γ, générer un nombre aléatoire uniforme a_i, dans la plage donnée par :

$$-\gamma \leq a_i \leq \gamma \tag{4.9}$$

4) En utilisant a_i, calculez w_i, avec l'équation 4.10, répétez les étapes 2 et 3 n fois pour calculer les différents n poids .

$$w_i = \hat{w}.\sqrt{a_i + 1} \tag{4.10}$$

5) Calculer le seuil w_0, avec :

$$w_0 = -0.5. \sum_{i=1}^{n} w_i \tag{4.11}$$

Nous devons fixer préalablement les valeurs des paramètres k et γ pour l'application de la méthode.

4.2.6 Méthode de Glorot et Bengio, 2010

En 2010, Glorot et Bengio publient une méthode de convergence du gradient dont l'initialisation peut prendre plusieurs forme, [14]. Il s'agit de donner des valeurs aléatoires aux poids, ou d'obtenir des valeurs de poids à l'aide d'un pré-entraînement d'apprentissage non-supervisé.

Glorot et Bengio proposent une initialisation appelée initialisation de Xavier, qui proposent d'attribuer aux poids du réseau les valeurs d'une distribution caractérisée par une

moyenne nulle, et une variance à valeur fixée. Deux distributions peuvent être utilisées, la distribution Uniforme et la distribution Gaussienne. Généralement, les logiciel utilisent par défaut la distribution Uniforme. Cette méthode est utilisé certainement pour fournir un cadre de comparaison avec les autres méthodes d'initialisation.

4.3 Méthode de Nguyen-Widrow, 1990

4.3.1 Description du Modèle

Notons X et Y les vecteurs d'entrée et de sortie du réseau. La relation entre X et Y d'un RN à deux couches peut être décrite par l'équation suivante :

$$Y = \sum_{i=0}^{k-1} v_i \cdot f(W_i^t X + w_{bi}) \tag{4.12}$$

où k est le nombre de nœuds cachés, f est la fonction sigmoïde, W_i est le vecteur de poids du $i^{\text{ème}}$ nœud de la couche cachée, w_{bi} est le biais du $i$ième nœud caché, et v_i est le poids de la couche de sortie qui relie la $i^{\text{ème}}$ unité cachée à la sortie.

La structure d'un réseau à propagation avant à deux couches est illustrée à la figure 4.1.

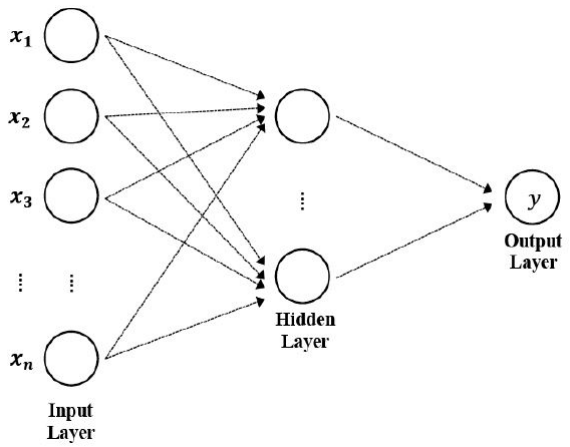


FIGURE 4.1 – Réseau de neurones à propagation avant à deux couches

Dans la couche cachée, chaque neurone est connecté à tous les neurones d'entrée, et chaque connexion a un poids qui lui est attaché. Le neurone de la couche de sortie prend

les activations de tous les neurones cachés comme entrée, [3].

Les réseaux à propagation avant apprennent en ajustant leurs poids de connexion afin de diminuer de manière itérative une mesure d'erreur associée à ses sorties pour un apprentissage simple donné, [7], [45], [17]. La fonction d'activation utilisée à chaque neurone caché est une fonction sigmoïde. Par exemple la fonction tangente hyperbolique, définie pour $x \in \mathbb{R}$ par l'équation 4.15, [38].

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.13}$$

où e désigne la fonction exponentielle.

4.3.2 Méthode d'initialisation de Nguyen-Widrow

Nguyen et Widrow, [40], ont proposé une idée simple mais efficace : rapprocher les valeurs initiales des emplacements où les valeurs finales sont supposées se trouver. De cette façon, de grandes distances seraient gagnées en faveur du temps d'entraînement.

L'idée d'initialisation proposée dans la méthode de Nguyen-Widrow est basée sur ce qui suit :

La fonction de transfert des neurones cachés est sigmoïde (définie comme une cartographie bornée, monotone croissante, continue et différentiable, [38]). Ce type de fonctions a la particularité d'être strictement monotones dans un certain intervalle et quasi constants en dehors, [46]. La taille de cet intervalle, où le neurone est actif, est liée au poids w qui pondère la variable d'entrée ; plus le poids est grand, plus l'intervalle est étroit, [29], [46], [38].

Par contre, la localisation du centre, I_C, de cet intervalle est déterminée par le rapport du biais w_b et du poids w, [1], [46], c'est-à-dire :

$$(I_C = -\frac{w_b}{w}). \tag{4.14}$$

Ainsi, connaître les paramètres d'un neurone dont la fonction de transfert est sigmoïde revient formellement à connaître l'intervalle où il est actif, [45], [46].

D'un autre côté, si un réseau est conçu pour approximer une fonction sur une certaine zone, l'intervalle actif de chacun de ses neurones cachés ne peut être que dans cette zone,

[1], [46]. Sans cela, il est simplement réduit à un biais supplémentaire et ne participe pas à l'approximation de la fonction.

A la fin de l'entraînement, les intervalles sur lesquels les neurones cachés sont actifs vont donc naturellement constituer un chevauchement de la région d'intérêt, [1], [54], [46].

À la lumière de cela, il faut s'arranger pour que les intervalles actifs de tous les neurones cachés soient à l'intérieur de la région d'intérêt. Autrement dit, les centres de ces intervalles, qui sont les points frontières des fonctions sigmoïdes, doivent tous être là et doivent être dispersés de manière judicieuse, [1], [45].

A titre d'illustration, considérons le cas qui est populaire pour la comparaison des méthodes de régression ; celui d'adapter la fonction

$$f(x) = \frac{sin(x)}{x} \tag{4.15}$$

sur l'intervalle $[-10, 10]$.

Un réseau sigmoïde/linéaire à deux couches avec 5 neurones cachés est proposé pour approximer cette fonction. L'algorithme d'entraînement de Levenberg-Marquardt est utilisé ici, qui est considéré comme efficace pour l'entraînement, [8], [3], et a de meilleures performances que tous les autres algorithmes d'entraînement, [32].

Dans la figure 4.2, nous pouvons voir la courbe de la fonction (4.15) ainsi que les cinq points frontières des neurones cachés que nous obtenons de deux manières différentes. La première façon est de faire le choix des valeurs initiales des poids et des biais avec une procédure purement aléatoire selon la loi Uniforme sur l'intervalle $[-0.5, 0.5]$, [1], [8], [46]. La seconde consiste à utiliser la méthode de Nguyen-Widrow.

Notez que les points frontières obtenus par la méthode de Nguyen-Widrow sont dispersés avec une bonne uniformité et ils sont tous dans la région d'intérêt. Alors que, par le choix purement aléatoire, quatre des points frontières obtenus ne sont concentrés que dans une petite partie de l'intervalle $[-10, 10]$, et en plus de cela, le cinquième point limite est résolument éloigné de cet intervalle.

Ainsi, la méthode de Nguyen-Widrow utilise toujours une procédure aléatoire dans le choix des valeurs initiales, mais avec la contrainte que les régions actives des neurones couvrent l'intérieur de l'intervalle d'intérêt et ne s'en éloignent pas.

Cette façon de faire les choses efficacement conduit à une réduction considérable du

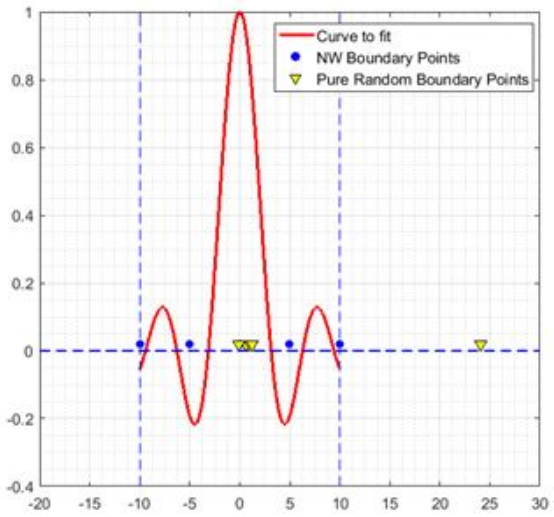


FIGURE 4.2 – La courbe à ajuster, ainsi que les points limites des neurones cachés obtenus par un processus purement aléatoire représentés en jaune, et ceux obtenus par la méthode de Nguyen-Widrow représentés en bleu.

temps d'entraînement pour une grande variété de problèmes. C'est ce qui fait qu'elle est aujourd'hui largement utilisée pour le choix des valeurs initiales lors du lancement de l'entraînement des réseaux à propagation avant, et la méthode la plus réputée pour la majorité des chercheurs, [4], [38], etc.

Cependant, la question qui se pose est de savoir pourquoi il n'y a pas persistance d'une même qualité d'entraînement lors d'expériences répétées.

4.3.3 Inconvénient de la méthode de Nguyen-Widrow

Les points frontières des neurones cachés sont situés dans la plage de la variable d'entrée lors de l'initialisation, par la méthode de Nguyen-Widrow, ils n'y restent pas toujours, essentiellement pendant les toutes premières époques d'apprentissage. Certains points frontières sont éjectés de cette plage, parfois même assez loin, [1], [46].

La figure 4.3 montre cette situation, pour l'exemple précédent, après une époque d'entraînement.

Les points frontières continuent de se déplacer perpétuellement pendant l'entraînement. Parfois, l'entraînement les ramène dans la gamme de la variable et l'entraînement se poursuit de manière satisfaisante. Mais souvent, certains points s'éloignent de plus en plus, [67].

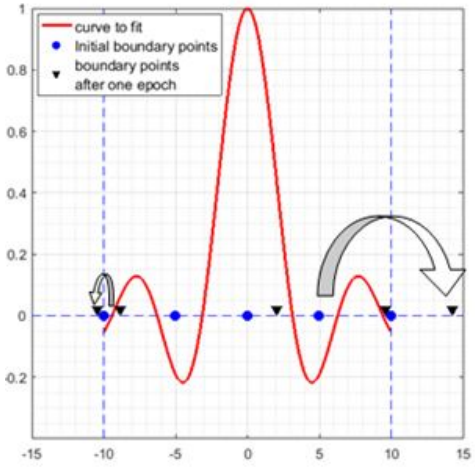


FIGURE 4.3 – Deux points limites sortent de la plage de la variable ; Le point 2 passe de $4,9948$ à $14,2726$ et le point 5 passe de $-9,999$ à $10,424$

Dans notre exemple, après 200 époques, le point 5 est piégé sur la valeur (-154.89). La sortie du réseau est illustrée à la figure 4.4.

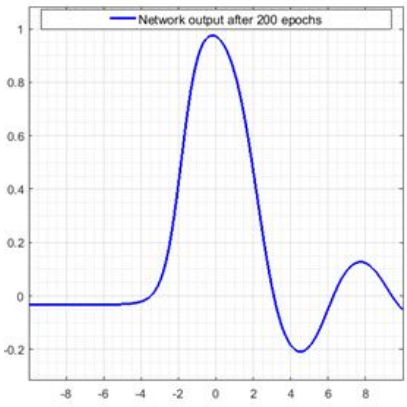


FIGURE 4.4 – Sortie du réseau après 200 époques

On voit que l'approximation de la fonction cible n'est pas satisfaisante. Le minimum de la surface de performance pour la régression est l'erreur quadratique moyenne (EQM), et c'est un outil important pour visualiser les effets de l'adaptation des poids et c'est aussi un terme de mesure de la qualité, [17], [68], [3], qui est ici égale à 0.0054.

Les sorties des cinq neurones cachés sont montrées sur la figure 4.5.

La figure 4.5 montre que le neurone 5 est saturé et prend la valeur 1 sur toute la plage de la variable d'entrée.

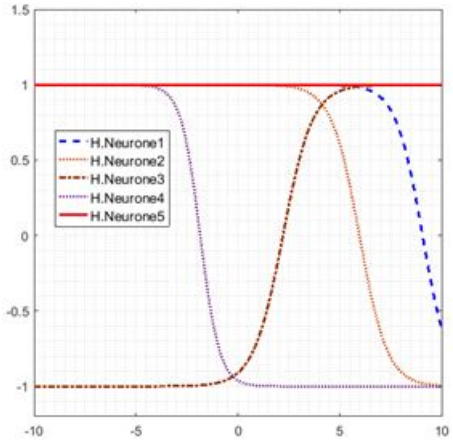


FIGURE 4.5 – Sorties des neurones cachés après 200 époques

En fait, ce ne sont que quatre neurones qui contribuent à la construction de l'approximation de la fonction cible. Le neurone 5 est presque neutralisé par saturation. Au fur et à mesure que l'entraînement progresse, les variations affectant ses paramètres deviennent de plus en plus insignifiantes, de sorte que l'intervalle actif reste en dehors de la région d'intérêt jusqu'à la fin de l'entraînement. Le rôle de ce neurone se réduit à celui d'un "biais" supplémentaire qui s'ajoute à celui du neurone de sortie, [31], [38].

C'est comme si, par accident, la complexité du réseau était réduite. Le réseau est censé avoir la flexibilité d'un réseau de cinq neurones cachés, mais en fait il se comporte comme un réseau de quatre neurones cachés. Cela a un impact négatif sur l'accordabilité du réseau, surtout lorsque les neurones cachés ne sont pas en excès, [32], [68], comme c'est le cas ici.

En mode supervisé, ce qui est commun aux algorithmes d'apprentissage des RN, c'est qu'ils utilisent les erreurs enregistrées entre les cibles et les réponses du réseau pour apporter les corrections appropriées aux différentes valeurs des paramètres, [8], [3]. Plus ces erreurs sont grandes et plus les corrections sont conséquentes, [42]. Pour cette raison, les changements les plus importants sont apportés au tout début de l'entraînement car c'est là que les erreurs sont les plus importantes, [57].

Pour un point frontière, Bp, les changements sont plus accentués par le fait que ce point est défini comme le rapport entre le biais w_b et le poids w, [40] :

$$Bp = -\frac{w_b}{w} \tag{4.16}$$

Lorsque les grandeurs de ces deux paramètres varient dans des directions opposées, le changement affectant le point frontière est substantiel, [1].

La fonction de coût, (EQM), utilisée pour évaluer la qualité de l'approximation est calculée uniquement avec les valeurs de la variable qui se trouvent dans la région d'intérêt, [45], [42], [17]. Si la zone active d'un neurone est suffisamment éloignée de cette région, les modifications de ses paramètres n'auront aucune influence sur la fonction de coût. Le déplacement du point frontière n'aura plus de répercussion sur l'entraînement, [13], [17].

La méthode de Nguyen-Widrow impose principalement des contraintes sur les valeurs initiales des paramètres de couche cachée. Ceux de la couche de sortie, lorsque la fonction de transfert est non bornée, sont choisis selon la loi Uniforme par un processus purement aléatoire, sans contraintes supplémentaires, [1], [45]. Cependant, ils sont déterminants car, s'ils sont inappropriés, ils provoquent des écarts importants entre les réponses du réseau et les cibles, [46]. Cela conduit à une grande erreur globale qui à son tour provoque une déstabilisation des points frontières des neurones cachés. Certains neurones peuvent être saturés sur la zone d'intérêt, [1], [8]. La complexité du réseau est ainsi automatiquement réduite, [38].

Ce que l'on remarque, c'est que lorsque nous effectuons une réinitialisation avec la méthode Nguyen-Widrow, les valeurs des paramètres de la couche de sortie varient considérablement. C'est pourquoi les résultats sont si différents d'une initialisation à l'autre, alors que les points frontières sont pratiquement sur les mêmes positions, [12], [8], [38].

Il n'est pas possible d'imposer des contraintes sur les valeurs initiales des paramètres de la couche de sortie car elles peuvent dépendre de la fonction cible qui est, pour la plupart, inconnue. Il faut se rappeler que même pour la couche cachée, les points frontières ne dépendent que très vaguement de la fonction cible, [45].

En effet, c'est seulement leur nombre, qui est celui des neurones cachés, qui est fixé au regard de ce qui est présupposé sur la complexité de la fonction cible, [7]. Leur étalement dépend de la variable d'entrée, [45].

CHAPITRE 5

AMÉLIORATION DE LA MÉTHODE DE NGUYEN-WIDROW

5.1 Préambule

Le choix des valeurs initiales des paramètres du réseau de neurones est d'une importance cruciale pour la conduite et la réussite de l'entraînement. La méthode la plus connue et la plus utilisée pour effectuer cette tâche est la méthode de Nguyen-Widrow. Elle a apporté un avantage bien établi sur la méthode traditionnelle de choix purement aléatoire. Cependant, cet avantage n'est pas toujours garanti. Un vice caché peut apparaître dans certaines situations entraînant une dégradation de la qualité de l'entraînement.

Dans cet ouvrage, l'existence de ce vice caché est révélée et la manière d'y remédier est proposée. Nous montrons comment une procédure simple, basée sur des réinitialisations conditionnelles, élimine les conditions de démarrage inadaptées et garantit la régularité d'une bonne qualité d'entraînement. De cette façon, la recherche de l'architecture optimale d'un réseau lors du traitement d'un problème donné devient plus rapide et plus fiable.

5.2 Méthode de Saadi F. et Chibat A. 2022

L'éjection des points frontières de la région d'intérêt est un indice qui peut révéler si l'entraînement a un bon départ ou non. En effet, cet indice peut être utilisé avanta-

geusement pour savoir si les valeurs initiales des paramètres de la deuxième couche, qui sont choisies par un processus purement aléatoire, sont proches des emplacements où les valeurs finales sont supposées être, [45]. S'ils le sont, alors les erreurs entre les cibles et leurs estimations ne seraient pas très importantes, et donc il n'y aurait pas ce phénomène d'éjection. Mais si elles sont éloignées, l'ampleur des erreurs résultantes conduirait à ce phénomène, [1].

Lorsque l'architecture du réseau est bonne (c'est-à-dire qu'elle a un nombre optimal de neurones cachés), les réinitialisations ne se produiront qu'au tout début de l'entraînement. Après cela, les points frontières ont tendance à se stabiliser progressivement dans la zone d'intérêt, sans perturbations ultérieures. Mais si la complexité du réseau est insuffisante, les perturbations peuvent perdurer jusqu'à la fin de l'entraînement, [45].

A titre d'illustration, revenons à notre exemple d'ajustement de la fonction (4.15).

En partant des mêmes valeurs initiales montrées dans la figure 4.3, et en permettant à l'entraînement de se poursuivre jusqu'à des époques de 1000, nous avons observé que l'EQM résultant était de 0.0019.

Mais, lorsque nous avons inclus le test et la réinitialisation conditionnelle dans le programme, nous avons constaté que l'EQM avait atteint $5,0792 \cdot e - 04$ sur le même nombre d'époques. Cette valeur ne serait pas atteinte sans initialisation conditionnelle, quel que soit le nombre d'époques attribué. On peut voir ici que la valeur de l'EQM avec réinitialisation est plus petite que celle sans réinitialisation et qu'il y a une grande différence entre ces deux valeurs.

La Figure 5.1 montre l'évolution des points frontières au cours de l'apprentissage. Il y a eu des éjections de 29 des points limites de la région d'intérêt et des réinitialisations successives de 29. Après cela, les valeurs sont devenues presque stables.

On voit aussi sur la Figure 5.1 que les réinitialisations conditionnels permettent d'éviter le problème de l'éjection des points frontières dès le début de l'entraînement.

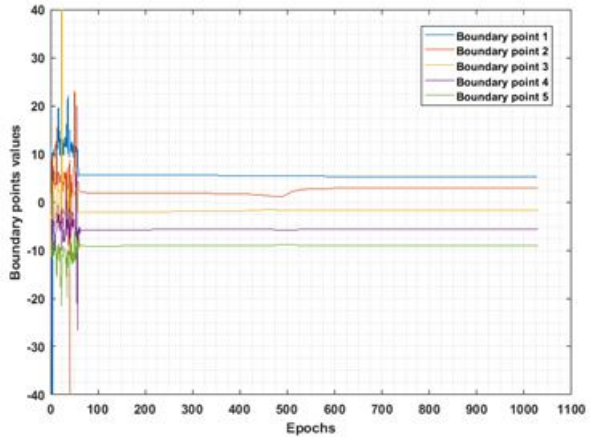


FIGURE 5.1 – Évolution des points limites au cours des époques

5.3 Détermination du nombre optimal de neurones cachés

Il est bien connu que déterminer le nombre optimal de neurones cachés est une opération délicate qui ne découle pas d'une procédure systématique, [42], [32].

En effet, dans l'apprentissage des RN, l'aléatoire intervient à plusieurs niveaux : dans le bruit qui altère les données, dans la subdivision de l'ensemble d'apprentissage en sous-ensembles (entraînement, test et validation), et dans l'initialisation des paramètres, [38].

Le remède est de recourir à des répétitions multiples afin de neutraliser cet effet d'aléatoire. L'objectif est de détecter le meilleur réseau, celui avec la meilleure architecture et les meilleurs paramètres, adaptés au problème traité. Il est évident que tous les neurones cachés de ce réseau doivent être actifs dans la zone d'intérêt ; sinon l'architecture n'aurait pas été optimale, [45].

La pratique a été que, pour déterminer le nombre optimal de neurones cachés, l'entraînement pour chaque nombre candidat est répété autant de fois jusqu'à ce que les résultats les plus réussis en termes de fonction de coût soient obtenus, [42].

Le succès de la méthode de Nguyen-Widrow vient du fait qu'elle assure de bonnes valeurs initiales. Mais cela n'est vrai que pour le début de l'entraînement. Car il y a un fait qui n'est pas tout à fait visible : ces conditions initiales favorables se dégradent au début de l'entraînement. Ainsi, la simple répétition de l'expérience ne fournit que de

bonnes conditions initiales, mais n'empêche pas une dégradation supplémentaire. En fait, c'est ce phénomène de dégradation, plutôt que les valeurs initiales, qui donne lieu aux répétitions de l'expérience.

En revanche, avec les remises conditionnelles les écueils sont détectés dès le départ et les répétitions sont automatisées et sélectives. Pour chaque nombre proposé de neurones cachés, une seule manipulation avec des réinitialisations conditionnelles équivaut en fait à une série de manipulations ordinaires qui se terminent par un bon résultat. Les configurations défavorables ont été éliminées dès le départ.

En réduisant une série de manipulations à une, on obtient un gain de temps considérable. Cela rend cette méthode appropriée pour trouver le nombre optimal de neurones cachés.

5.4 Résultats et discussion

Nous utilisons MATLAB $R2016a$ fourni par The MathWorks, Inc, pour construire et former nos réseaux.

La fonction d'entraînement du Levenberg-Marquardt utilisée dans le logiciel est trainlm.

L'exemple est toujours celui de l'ajustement de la fonction (4.15) sur l'intervalle $[-10, 10]$.

Deux ensembles sont créés, un pour l'entraînement et un pour le test. Chacun d'eux contient 5000 observations, (x_i, y_i).

Un bruit aléatoire généré par une loi Uniforme sur l'intervalle $[-0,2, 0,2]$ a été ajouté aux cibles y_i de l'ensemble d'apprentissage.

Un réseau sigmoïde/linéaire à deux couches est proposé pour effectuer cette tâche de régression. La procédure de réinitialisation conditionnelle a été mise en œuvre pour déterminer le nombre optimal de neurones cachés pour ce problème.

Les manipulations ont été faites pour un nombre de neurones cachés allant de 2 à 12. Pour chaque cas, l'expérience a été répétée 100 fois.

Dans chacune des expériences, deux réseaux identiques ont été créés. Dans chaque essai, les deux réseaux ont été initialisés avec les mêmes valeurs produites par la méthode de Nguyen-Widrow. Le nombre d'époques a été fixé à 1000. Le réinitialisations conditionnel

a été introduit dans l'apprentissage du premier réseau mais pas dans celui du second.

Dans toutes ces manipulations, les comparaisons ont été faites sur les valeurs de l'EQM enregistrées sur l'ensemble de test,[42].

La performance des données de test représente la précision de chaque prédicteur du RN. Les plus petites valeurs de l'EQM représentent le meilleur modèle, [17], [68].

Les résultats du EQM moyen, du Max EQM et du Rang EQM sur l'ensemble de test avec et sans réinitialisations sont rapportés dans la Table 5.1, et illustrés dans les Figures 5.2, 5.3 et 5.4.

	Mean EQM		Max EQM		Range EQM	
Number of neurons	Without reset	With reset	Without reset	With reset	Without reset	With reset
2	0.0188	0.0107	0.0994	0.0121	0.0890	0.0017
3	0.0086	0.0069	0.011	0.0102	0.0098	0.0082
4	0.0039	0.0026	0.0118	0.0058	0.0118	0.0057
5	0.0027	9.8423e-04	0.0102	0.0054	0.0101	0.0054
6	1.1853e-04	4.9514e-05	5.7625e-04	4.9971e-05	5.3577e-04	9.4713e-06
7	2.5091e-04	8.1336e-05	0.0011	5.7840e-04	0.0011	5.3501e-04
8	2.8698e-04	6.6727e-05	0.0011	8.5763e-05	0.0011	2.8189e-05
9	6.7849e-05	7.2449e-05	5.6317e-04	7.6878e-05	5.1948e-04	8.9150e-06
10	9.1642e-05	8.2559e-05	5.7088e-04	9.5167e-05	5.1588e-04	1.6699e-05
11	5.8677e-05	5.6352e-05	6.9573e-05	7.0012e-05	1.6785e-05	1.7895e-05
12	1.1642e-04	1.1597e-04	1.1856e-04	1.2482e-04	6.5363e-06	1.2801e-05

TABLE 5.1 – Valeurs des erreurs quadratiques moyennes

La Figure 5.2 illustre la comparaison entre les valeurs de la moyenne de l'EQM sur l'ensemble de test avec réinitialisation et celles sans réinitialisation.

La Figure 5.3 illustre la comparaison entre les valeurs du MAX EQM sur l'ensemble de test avec réinitialisation et celles sans réinitialisation.

La Figure 5.4 illustre la comparaison entre les valeurs de le rang des EQM sur l'ensemble de test avec réinitialisation et celles sans réinitialisation.

La Table 5.1 montre clairement les avantages de l'ajout de la réinitialisation conditionnelle à la méthode Nguyen-Widrow.

Premièrement, tout indique que le nombre optimal de neurones cachés est de 6. Il est déterminé sans ambiguïté, bien que dans cet exemple, les cibles de l'ensemble d'apprentissage soient modifiées avec des erreurs assez importantes.

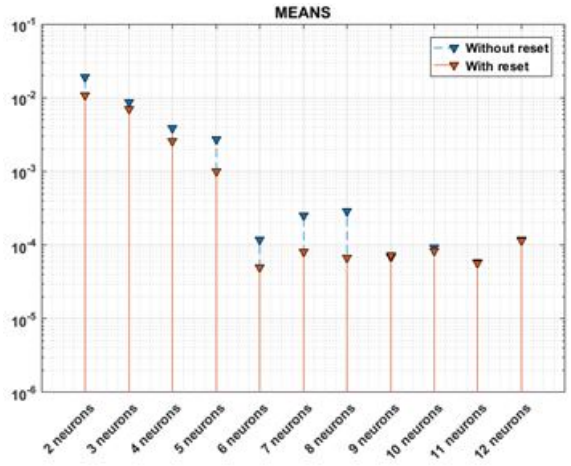


FIGURE 5.2 – Moyennes des erreurs quadratiques moyennes

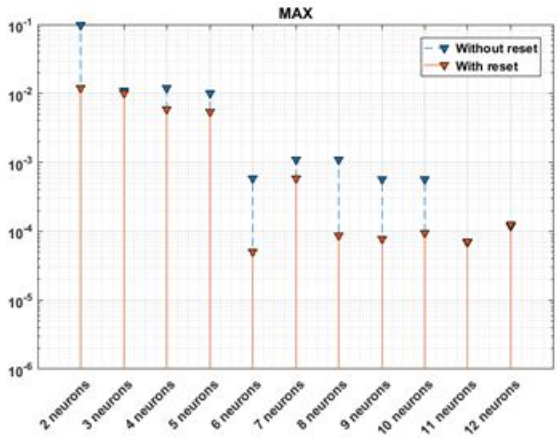


FIGURE 5.3 – Max des erreurs quadratiques moyennes

D'après la Table 5.1, on peut voir que les valeurs de la moyenne de EQM avec réinitialisation sont plus petites que celles sans réinitialisation et il y a une grande différence entre ces deux valeurs.

En même temps, également à partir du tableau 1, on peut voir que ce phénomène se répète dans les valeurs à la fois de Max EQM et du Rang des EQM.

Les Figures 5.2, 5.3 et 5.4 montrent mieux la grande différence dans les valeurs de chacun de la moyenne de EQM, Max EQM et Rang EQM, parlé ci-dessus, spécialement dans ce dernier de la Figure 5.4.

De plus, il convient de noter que, presque dans tous les cas, d'après la Table 5.1, l'EQM sur l'ensemble de test montre une diminution substantielle, ainsi que le nombre

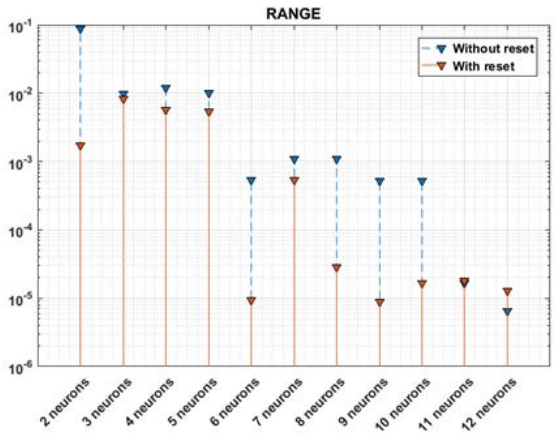


FIGURE 5.4 – Rang des EQM

de neurones cachés augmente, lorsque la réinitialisation conditionnelle est définie :

- En termes de moyenne de l'EQM sur 100 essais effectués dans les mêmes conditions et avec les mêmes valeurs de départ pour les paramètres.
- En termes de valeur maximale de l'EQM sur tous les 100 essais.

A noter qu'un réseau à 6 neurones cachés obtient les meilleurs scores selon ces deux critères.

Habituellement, la moyenne des EQM est un critère suffisant pour trancher en faveur d'une architecture par rapport aux autres, [45], [68]. Cependant, dans notre cas, le deuxième critère est également d'une grande importance. Car, en pratique, quand on n'a pas la possibilité de refaire les expériences, il faut savoir quelle est la situation la plus défavorable dans laquelle on est susceptible de se trouver. La réinitialisation conditionnelle, en filtrant et en éliminant les cas où les erreurs sont si importantes qu'elles déstabilisent le réseau, évite d'avoir une situation trop défavorable, même si elle est exceptionnelle.

Un autre problème, souvent rencontré lors de l'entraînement des RN, est la non persistance de la même qualité de ces réseaux dans des expériences répétées.

Cependant, comme nous pouvons le voir sur la Figure 5.5, les 100 valeurs de l'EQM pour les 6 neurones cachés du réseau sont dispersées sur un intervalle très étroit. Son envergure est de $9,4713e-06$.

Cela signifie que la qualité de ce réseau est très stable et qu'un seul essai aurait résumé les informations fournies par les 100 essais. Cela permet d'économiser le grand temps perdu par les répétitions futiles.

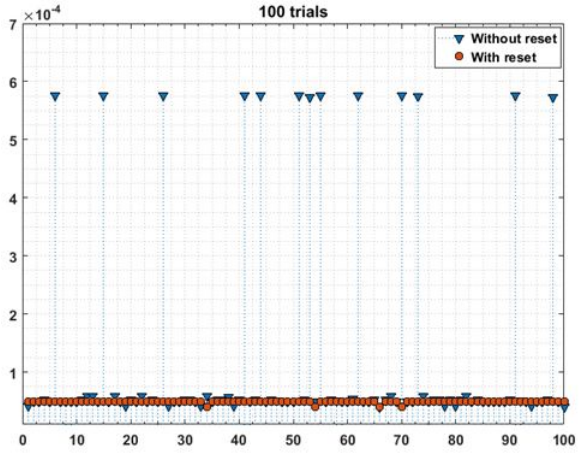


FIGURE 5.5 – Valeurs du MSE pour les 6 neurones cachés du réseau

À partir de la Figure 5.5, nous pouvons également voir la très grande différence entre les valeurs de l'EQM pour le réseau de 6 neurones cachés avec réinitialisation et sans réinitialisation dans tous les 100 essais. Ainsi, la réinitialisation conditionnelle des paramètres fournit un moyen important d'éviter le défaut caché dans la méthode Nguyen-Widrow. Surtout lorsque le nombre de neurones cachés dans le réseau est optimal.

Dans la zone où le sur-ajustement commence à s'installer, notamment pour les neurones cachés à 11 et 12, on constate que la réinitialisation conditionnelle n'apporte plus d'amélioration significative et peut même être contre-productive. Cela est dû au nombre excessif de neurones cachés qui empêchent les réseaux de se stabiliser et de provoquer des réinitialisations sur trop d'époques.

5.5 Conclusion

Ce travaille présente un moyen très simple de prévenir un défaut imperceptible dans la méthode de Nguyen-Widrow, conçu pour initialiser les paramètres des RNs avant l'entraînement, et permettre à tout entraînement de se poursuivre dans les meilleures conditions. Il s'agit d'une réinitialisation conditionnelle automatique des paramètres, qui se déclenche lorsqu'un mauvais démarrage de l'entraînement apparaît occasionnellement, lors de certaines répétitions plutôt que d'autres, et provoque l'échec de l'entraînement.

En se prémunissant des initialisations défavorables à un bon apprentissage, ce procédé réduit le doute sur la qualité du réseau formé et permet un gain considérable de temps

perdu par des répétitions inutiles.

En conséquence directe, trouver le nombre optimal de neurones cachés devient plus facile, plus rapide et plus fiable.

Abstract

Artificial neural networks (ANN) are universal approximators that allow to express the correlation between input data and output data. Learning by ANN is based on the adaptation of the free parameters of the network by iteratively varying the values of the latter, from arbitrary initial values until reaching final values. The latter should, in principle, attribute to the network an optimal behavior, in accordance with certain pre-established criteria.

With the advent of neural methods, the initialization of ANNs has become a central problem and has been studied by many researchers. The Nguyen-Widrow method (1990), although old, has established itself as a reference method. It is widely recognized and is the most used.

Unfortunately, the benefits of the Nguyen-Widrow method are not always guaranteed. The repetition of certain experiments, with apparently identical conditions, gives satisfactory results but others are disappointing, in terms of learning time, but also in terms of precision of the estimates. For this reason, it has always been recommended to make several training attempts and choose the one that would have produced the best results.

The purpose of this work is to reveal the insidious reasons that cause unsuccessful trials and prevent taking full advantage of the Nguyen-Widrow method. It reveals the existence of a defect, imperceptible during initialization, but which sets in at the very beginning of training, which will have a negative impact on the quality of training, in terms of training time. execution and performance.

Keywords : Non-parametric statistics ; Selection of variables ; Artificial neural network ; Initialization of parameters.

ملخص

الشبكات العصبية الاصطناعية (ANN) هي أدوات تقريب عالمية تسمح بالتعبير عن الارتباط بين بيانات الإدخال وبيانات الإخراج. يعتمد التعلم بواسطة ANN على تكييف العوامل الحرةللشبكة عن طريق التغيير المتكرر لقيم هده الأخيرة، من القيم الأولية التعسفية حتى الوصول إلى القيم النهائية. يجب أن ينسب الأخير ، من حيث المبدأ ، إلى الشبكة السلوك الأمثل ، وفقًا لمعايير محددة مسبقًا.

مع ظهور الأساليب العصبية، أصبحت تهيئة الشبكات العصبية الاصطناعية مشكلة مركزية وقد تمت دراستها من قبل العديد من الباحثين. طريقة Nguyen–Widrow (1990)، على الرغم من كونها قديمة ، أثبتت نفسها كطريقة مرجعية. إنه معترف به على نطاق واسع وهو الأكثر استخدامًا.

لسوء الحظ، فإن فوائد طريقة Nguyen–Widrow ليست مضمونة دائمًا. إن تكرار تجارب معينة، بشروط متطابقة ظاهريًا، يعطي نتائج مرضية، لكن البعض الآخر مخيب للآمال، من حيث وقت التعلم، ولكن أيضًا من حيث دقة التقديرات. لهذا السبب، يوصى دائمًا بإجراء العديد من المحاولات التدريبية واختيار المحاولة التي كانت ستحقق أفضل النتائج.

الغرض من هذا العمل هو الكشف عن الأسباب الخبيثة التي تسببت في المحاكمات غير الناجحة ومنع الاستفادة الكاملة من طريقة Nguyen–Widrow. إنه يكشف عن وجود عيب غير محسوس أثناء التهيئة، ولكنه يظهر في بداية التدريب، والذي سيكون له تأثير سلبي على جودة التدريب، من حيث وقت التدريب والتنفيذ والأداء.

كلمات مفتاحية: إحصائيات دون عوامل؛ اختيار المتغيرات؛ شبكة اعصاب صناعية؛ تهيئة العوامل.

BIBLIOGRAPHIE

[1] Adam, S.P., Karras, D.A., Magoulas, G.D. et Vrahatis, M.N. (2014), 'Solving the linear interval tolerance problem for weight initialization of neural networks', *Neural Networks*, Vol. 54, pp.17–37.

[2] Besse, P. (2006), 'Apprentissage Statistique et Data mining', *Institut de Mathématiques de Toulouse*.

[3] Corelli, A. (2020), 'Neural networks and technical analysis for price prediction : the case of Borsa Italiana SP MIB', *Int. J. Business Continuity and Risk Management*, Vol. 10, No. 1, pp. 1–22.

[4] Çatalbaş, B. et Morgül, Ö. (2018, May), 'A new initialization method for artificial neural networks : Laplacian', *IEEE 26th Signal Processing and Communications Applications Conference*, pp. 1–4.

[5] De Castro, L.N., Iyoda, E.M., Von Zuben, F.J. et Gudwin, R. (1998, December), 'Feedforward neural network initialization : an evolutionary approach', *IEEE Proceedings 5th Brazilian Symposium on Neural Networks*, Cat. No. 98EX209, pp. 43–48.

[6] Denoeux, T. et Lengellé, R. (1993), 'Initializing back propagation networks with prototypes', *Neural Networks*, Vol. 6, No. 3, pp. 351–363.

[7] De Sousa, C.A. (2016, July), 'An overview on weight initialization methods for feedforward neural networks', *IEEE International Joint Conference on Neural Networks*, pp. 52–59.

[8] Dolezel, P., Skrabanek, P. et Gago, L. (2016), 'Weight initialization possibilities for feedforward neural network with linear saturated activation functions', *IFAC-PapersOnLine*, Vol. 49, No. 25, pp. 49–54.

[9] Drago, G.P., Ridella, S. (1992), 'Statistically Controlled Activation Weight Initialization (SCAWI)'. *IEEE Transactions. on Neural Networks*, Vol. 3, No. 4, pp. 627-631.

[10] Dret, H. L. et Lamboley, J. (2018), 'Analyse fonctionnelle approfondie et calcul des variations', *Sorbonne université*.

[11] Dreyfus, G., Martinez, J. M., Samuelides, M., Gordon, M. B., Badran, F., et Thiria, S. (2008), 'Apprentissage statistique'.

[12] Erdogmus, D., Fontenla-Romero, O., Principe, J. C., Alonso-Betanzos, A., Castillo, E. et Jenssen, R. (2003, July), 'Accurate initialization of neural network weights by backpropagation of the desired response', *IEEE Proceedings of the International Joint Conference on Neural Networks*, Vol. 3, pp. 2005–2010.

[13] Erdogmus, D., Fontenla-Romero, O., Principe, J.C., Alonso-Betanzos, A. et Castillo, E. (2005), 'Linear-least-squares initialization of multilayer perceptrons through backpropagation of the desired response', *IEEE Transactions on Neural Networks*, Vol. 16, No. 2, pp. 325–337.

[14] Etienne, C. (2019), 'Apprentissage profond appliqué à la reconnaissance des émotions dans la voix', *Université de Paris Sud*.

[15] Fernández-Redondo, M. et Hernandez-Espinosa, C. (2000), 'A comparison among weight initialization methods for multilayer feedforward networks', *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. Neural Computing : New Challenges and Perspectives for the New Millennium*, Vol. 4, pp. 543–548.

[16] Fuchen, S., Kar-Ann, T., Manuel, G., R., et Kezhi, M. (2014), 'Extreme Learning Machines 2013 : Algorithms and Applications', *Springer Cham Heidelberg New York Dordrecht London*, Vol. 6.

[17] Ganesh, S. S., Arulmozhivarman. P., et Tatavarti, V. S. N. (2018), 'Air quality index forecasting using artificial neural networks- a case study on Delhi', *International*

Journal of Environement and WastEEE Assigning initial weights in feedforward neural networks, Vol. 22, No. 1-4, pp. 4–23.

[18] Gentle, J. E. (2002), 'Elements of Computational Statistics', *New York : Springer-Verlag.*

[19] Govindarajulu, Z. (2007), 'Nonparametric Inference', *University of Kentucky, USA.*

[20] Guang-Bin, H. , Qin-Yu, Z., et Chee-Kheong, S. (2006), 'Extreme learning machine : Theory and applications',*Neurocomputing*, Vol. 70, pp. 489–501.

[21] Guang-Bin, H., Dian, H. W., et Yuan, L. (2011), 'Extreme learning machines : a survey', *Int. J. Mach. Learn. and Cyber*, Vol. 2, pp. 107–122.

[22] Gustafson, K. et Sartoris, G. (1998), 'Assigning initial weights in feedforward neural networks', *IEEE Assigning initial weights in feedforward neural networks*, Vol. 31, No. 20, pp. 1053–1058.

[23] Hastie, T., Tibshirani, R., Friedman, J. (2008), 'The Elements of Statistical Learning : Data Mining, Inference, and Prediction', *Springer Series in Statistics, Second Edition, Stanford, California.*

[24] Haykin, S. (2009), 'Neural Networks and Learning Machines, Third Edition', *McMaster University, Canada.*

[25] Härdle, W. (1994), 'Applied Nonparametric Regression', *Humboldt-Universitãt zu Berlin.*

[26] Hernàndez-Espinosa, C. et Fernandez-Redondo, M. (2001), 'Multilayer feedforward weight initialization', *IEEE International Joint Conference on Neural Networks'01. Proceedings* , Vol. 1, No. 01CH37222, pp. 166–170.

[27] Huang. G.B. , Zhu. O.Y. , et Siew. C.K. (2006), 'Extreme learning machine : theory and applications'. *Neurocomputing*, Vol. 1, No. 70, pp. 489–501.

[28] Huang. G.B. , Wang. D.H., et Lan. Y. (2011), 'Extreme learning machines : a survey'. *Int. J. Mach. Learn. Cybern.* Vol. 1, No. 2, pp. 107–122.

[29] Jamett, M. et Acuña, G. (2006, November), 'An interval approach for weight's initialization of feedforward neural networks', *Mexican International Conference on Artificial Intelligence, Springer, Berlin, Heidelberg*, pp. 305–315.

[30] Kim, Y.K. et Ra, J.B. (1991, November), 'Weight value initialization for improving training speed in the backpropagation network', *IEEE [Proceedings] International Joint Conference on Neural Networks*, pp. 2396–2401.

[31] Kim, L.S. (1993, October), 'Initializing weights to a hidden layer of a multilayer neural network by linear programming', *IEEE Proceedings of International Conference on Neural Networks*, Vol. 2, pp. 1701–1704.

[32] Kumar, D. , A. et Murugan, S. (2018), ' Performance analysis of NARX neural network backpropagation algorithm by various training functions for time series data', *Int. J. Data Science*, Vol. 3, No. 4, pp. 308–325.

[33] Lee, Y., Oh, S.H. et Kim, M.W. (1991, July), 'The effect of initial weights on premature saturation in back-propagation learning', *IEEE Seattle international joint conference on neural networks*, Vol. 1, pp. 765–770.

[34] Lee, H. K. H. (2004), 'Bayesian Nonparametrics via Neural Networks', *University of California, Santa Cruz Santa Cruz, California.*

[35] Lehtokangas, M., Saarinen, J., Kaski, K. et Huuhtanen, P. (1995), 'Initializing weights of a multilayer perceptron network by using the orthogonal least squares algorithm', *Neural Computation*, Vol. 7, No. 5, pp. 982–999.

[36] Li, G., Alnuweiri, H., Wu, Y., (1993), 'Acceleration of Backpropagations through Initial Weight Re-Training with Delta Rule', *Roc. of the IEEE Int. Conference on Neural Networks, ICN93.* Vol. 1, pp. 580-585.

[37] Liu, Y., Zhou, C.F. et Chen, Y.W. (2006, August), 'Weight initialization of feedforward neural networks by means of partial least squares', *IEEE International Conference on Machine Learning and Cybernetics*, pp. 3119–3122.

[38] Mittal, A., Singh, A.P. et Chandra, P. (2020), 'A Modification to the Nguyen–Widrow Weight Initialization Method', *Intelligent Systems, Technologies and Applications, Springer, Singapore*, pp. 141–153.

[39] Mohamed Yassin, A. (2007), 'Mise en œuvre de réseaux de neurones pour la modélisation de cinetiques réactionnelles en vue de la transposition Bath/continu', *Ecole Nationale d'Ingénieurs de Sfax, Tunisie.*

[40] Nguyen, D. et Widrow, B. (1990, June), 'Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights', *IEEE International Joint Conference on Neural Networks*, pp. 21–26.

[41] Norou, D. (2019), 'Modern Statistical Methods for Spatial and Multivariate Data', *Department of Mathematics and Statistics Old Dominion University Norfolk, VA, USA*.

[42] Oyedotun, K., O., Olaniyi, E., O. and Khashman, A. (2017), 'A simple and practical review of over-fitting in neural network learning', *International Journal of Applied Pattern Recognition*, Vol. 4, No. 4, pp. 307–328.

[43] Parizeau, M. (2004), 'Réseaux de neurones', *Université LAVAL*

[44] Pei, J.S., Wright, J.P. et Smyth, A.W. (2005, July), 'Neural network initialization with prototypes-a case study in function approximation', *IEEE Proceedings. International Joint Conference on Neural Networks*, Vol. 3, pp. 1377–1382.

[45] Qiao, J., Li, S. et Li, W. (2016), 'Mutual information based weight initialization method for sigmoidal feedforward neural networks', *Neurocomputing*, Vol. 6, pp. 676–683.

[46] Ramos, E. Z., Nakakuni, M. et Yfantis, E. (2017), 'Quantitative measures to evaluate neural network weight initialization strategies', *IEEE 7th Annual Computing and Communication Workshop and Conference, Las Vegas, NV, USA*.

[47] Rencher, A. C., Schaalje, G. B. (2008), 'Linear models in statistics', *Wiley-Interscience, second edition.*

[48] Rumelhart, D. E., McClelland, J. L. et Hinton, G. E. (2008), 'Parallel distributed processing : Explorations in the microstructures of cognition, Vol. 1 : Fondations'.

[49] Saadi, F. et Chibat, A. (2022), 'A simple way to enhance the efficiency of Nguyen-Widrow method in neural networks initialisation', *Int. J. Mathematics in Operational Research*, Vol. 21, No. 4, pp. 497–514.

[50] Samarasinghe, S. (2007), 'Neural Networks for Applied Sciences and Engineering : From Fundamentals to Complex Pattern Recognition', *New York.*

[51] Shimodaira, . (1994), 'A weight value initialization mehod for improving learnng performance of the Backrobagation algorithm in neural networks', *Proceedings of the* 6^th *International Conference on Tools with Artificial Intelligence*, pp. 672-675.

[52] Silverman, B. W. (2018), 'Density estimation for statistics and data analysis', *School of Mathematics University of Bath, UK.*

[53] Snyman, J. A., Wike, D. N. (2018), 'Practical Mathematical Optimization, Basic Optimization Theory and Gradient-Based Algorithms, Second Edition', *Springer Optimization and Its Applications*, Vol. 133.

[54] Sodhi, S.S. et Chandra, P. (2014), 'Interval based weight initialization method for sigmoidal feedforward artificial neural networks', *AASRI Procedia*, No. 6, pp. 19–25.

[55] Sodhi, S. S., Chandra, P. et Tanwar, S. (2014, July), 'A new weight initialization method for sigmoidal feedforward artificial neural networks', *IEEE International Joint Conference on Neural Networks*, pp. 291–29.

[56] Sun, F., Toh, K. A., Romay, M. G., et Mao, K. (2014), 'Extreme Learning Machines 2013 : Algorithms and Applications ', *Adaptation, Learning, and Optimization*, Vol. 16.

[57] Sun, W., Su, F. et Wang, L. (2017), 'Improving deep neural networks with multilayer maxout networks and a novel initialization method', *Neurocomputing*, Vol. 9, pp. 34–40.

[58] Takezawa, T. (2006), 'Introduction to nonparametric regression', *Tsukuba-shi ibarakiken, Japan.*

[59] Thompson, J. R., Tapia, R. A. (1987), 'Nonparametric Function Estimation, Modeling, and Simulation', *Society for Industrial and Applied Mathematics.*

[60] Timotheou, S. (2009), 'A novel weight initialization method for the random neural network', *Neurocomputing*, Vol. 73, No. 1-3, pp.160–168.

[61] Touzet, C. (1992), 'Les réseaux de neurones artificiels, introduction au Connexionnisme : Cours, Exercices et Travaux Pratiques'.

[62] Ultans, J., Moody, J., Rehfuss, S., et Slegelmann, H. (1995), 'Input Variable Selection for Neural Networks :Application to Predicting the U.S. usiness Cycle', *IEEE 1995*

Conference on Computational Intelligence for Financial Engineering (CIFEr) - New York, NY, USA .

[63] Varnava, T.M. et MEADE JR, A.J. (2011), 'An initialization method for feedforward artificial neural networks using polynomial bases', *Advances in Adaptive Data Analysis*, Vol. 3, No. 03, pp. 385–400.

[64] Wessels, L. F. A., Barnard, E., (1992), 'Avoiding False Local Minima by Proper Initialization of Connections', *IEEE Transactions on Neural Networks*, Vol. 3, N. 6, pp. 899-905.

[65] Yam, Y.F., Chow, T.W. et Leung, C.T. (1997), 'A new method in determining initial weights of feedforward neural networks for training enhancement', *Neurocomputing*, Vol. 16, No. 1, pp. 23–32.

[66] Yam, J.Y. et Chow, T.W. (2000), 'A weight initialization method for improving training speed in feedforward neural network', *Neurocomputing*, Vol. 30, No. 1-4, pp. 219–232.

[67] Yam, J.Y. et Chow, T.W. (2001), 'Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients', *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, pp. 430–434.

[68] Zhang, L. et Li, H. (2019), 'A mixed-coding adaptive differential evolution for optimising the architecture and parameters of feedforward neural networks', *Int. J. Sensor Networks*, Vol. 29, No. 4, pp. 262–274.

Résumé

Les réseaux de neurones artificiels (RNA) sont des approximateurs universels qui permettent d'exprimer la corrélation entre les données d'entrée et les données de sortie. L'apprentissage par les RNA est basé sur l'adaptation des paramètres libres du réseau en variant itérativement les valeurs de ces derniers, depuis des valeurs initiales arbitraires jusqu'à atteindre des valeurs finales. Ces derniers devraient, en principe, attribuer au réseau un comportement optimal, conformément à certains critères préétablis.

A l'avènement des méthodes neuronales, l'initialisation des RNA est devenue un problème central et a été étudiée par de nombreux chercheurs. La méthode de Nguyen-Widrow (1990), bien qu'ancienne, s'est imposée comme une méthode de référence. Elle est largement reconnu et est la plus utilisé.

Malheureusement, les avantages de la méthode de Nguyen-Widrow ne sont pas toujours garantis. La répétition de certaines expériences, avec des conditions apparemment identiques, donnent des résultats satisfaisants mais d'autres sont décevants, en termes de temps d'apprentissage, mais aussi en termes de précision des estimations. Pour cette raison, il a toujours été recommandé de faire plusieurs tentatives d'entraînement et de retenir celui qui aurait produit les meilleurs résultats.

Le but de ce travail est de révéler les raisons insidieuses qui causent les essais infructueux et empêchent de tirer pleinement parti de la méthode de Nguyen-Widrow. Il révèle l'existence d'un défaut, imperceptible lors de l'initialisation, mais qui s'installe au tout début de l'entraînement, ce qui aura un impact négatif sur la qualité de l'entraînement, en terme de temps d'exécution et les performances.

Mots clés : Statistique non paramétrique ; Sélection de variables ; Réseau de neurones artificiels ; Initialisation des paramètres.