INTRODUCTION AUX RESEAUX DE NEURONES

Résumé

Un réseau de neurones non bouclé est constitué par plusieurs opérateurs algébriques élémentaires appelés neurones et réalise une fonction paramétrée, non linéaire par rapport aux entrées et aux paramètres.

La famille des réseaux de neurones à une couche de neurones cachés possède la propriété d'approximation parcimonieuse. Cela signifie qu'elle est capable d'approcher n'importe quelle fonction bornée et suffisamment régulière, en utilisant moins de paramètres ajustables que les familles de fonctions usuelles telles que les polynômes.

Dans l'optique d'une modélisation statistique, on utilise les réseaux de neurones pour approcher la fonction de régression du processus. L'intérêt de la parcimonie est alors de limiter le nombre d'exemples nécessaires pour obtenir une bonne estimation de la fonction de régression.

Par opposition aux modèles linéaires par rapport aux paramètres, pour lesquels la solution des moindres carrés s'obtient en résolvant un système d'équations, l'ajustement des paramètres d'un réseau de neurones (appelé aussi "apprentissage") nécessite la mise en œuvre d'algorithmes itératifs. À cet égard, il convient de bien distinguer l'étape de calcul du gradient de la fonction de coût par rapport aux paramètres (par exemple par "rétropropagation du gradient") de l'étape de modification des paramètres (méthodes du gradient simple, de quasi-Newton, de Levenberg-Marquardt, etc.).

1.1 Introduction

L'objectif de ce chapitre est double : il s'agit tout d'abord de rappeler les définitions de base relatives aux réseaux de neurones ainsi que les propriétés mathématiques de certains d'entre eux. Ensuite, nous nous attacherons à détailler certains aspects de leur mise en œuvre, et plus particulièrement de leur apprentissage.
Un réseau de neurones est une fonction paramétrée qui est la composition d'opérateurs mathématiques simples appelés neurones formels (ou plus simplement neurones) pour les distinguer des neurones biologiques. Afin de préciser ces notions, nous commencerons par présenter les définitions relatives aux neurones avant de détailler différentes architectures de réseaux de neurones.

1.1.1 Les neurones

On appelle neurone une fonction algébrique non linéaire, paramétrée, à valeurs bornées, de variables réelles appelées entrées.
Par soucis de commodité, on commet fréquemment un abus de langage en désignant par le terme "neurone linéaire" une fonction linéaire (et plus généralement affine).
On a pris l'habitude de représenter un neurone formel comme indiqué sur la figure 1.1.

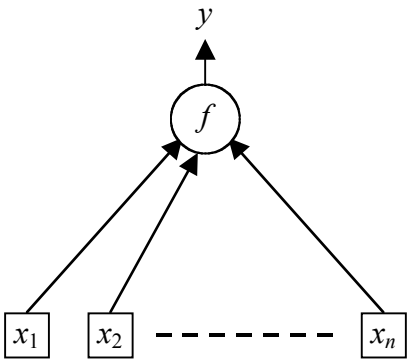

Figure 1.1 : Un neurone réalise une fonction non linéaire bornée $y = f(x_1, ..., x_n, \alpha_1, ..., \alpha_p)$ où les $\{x_i\}$ sont les entrées et les $\{\alpha_j\}$ sont les paramètres

Les paramètres dont dépend la valeur de *y* peuvent intervenir de deux manières :
- ils peuvent intervenir dans la fonction *f* elle-même,
- ils peuvent intervenir dans l'argument de la fonction *f*.

Les réseaux de fonctions radiales ou d'ondelettes (cf. [Oussar 98]) entrent dans la première catégorie : les paramètres ajustables sont le centre et la largeur (pour les fonctions radiales), ou le centre et la dilatation (pour les ondelettes).
Dans ce travail, nous avons toujours utilisé des neurones ou fonctions qui appartiennent à la seconde catégorie : l'argument de la fonction *f* est une combinaison linéaire des entrées du neurone (à laquelle on ajoute un terme constant, le "biais"), pondérées par les paramètres $\{\alpha_j\}$, qui sont fréquemment appelés "poids synaptiques" (ou plus simplement "poids") en référence à l'origine biologique des réseaux de neurones. Cette combinaison linéaire est appelée "potentiel".

$$v = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i \tag{1.1}$$

Le biais α_0 peut être envisagé comme le coefficient de pondération de l'entrée n° 0, dont la valeur est fixée à 1 :

$$v = \sum_{i=0}^{n} \alpha_i x_i \tag{1.2}$$

La valeur de la sortie du neurone est donc :

$$y = f(v) = f\left(\sum_{i=0}^{n} \alpha_i x_i\right) \tag{1.3}$$

La fonction *f* est appelée "fonction d'activation" ; les fonctions sigmoïdes et en particulier la tangente hyperbolique sont les plus utilisées.
Dans le présent mémoire, un neurone qui possède :
- un potentiel défini par la somme pondérée des entrées, y compris le biais,
- et une fonction d'activation sigmoïdale,

est représenté comme indiqué sur la figure 1.2 (a), la figure 1.2 (b) représentant un neurone linéaire.
Les fonctions réalisées par les neurones à fonction d'activation non linéaire, décrits ci-dessus, peuvent être combinées en un réseau de neurones. Dans un tel réseau, les entrées d'un neurone sont soit les entrées du réseau, soit les sorties d'autres neurones.

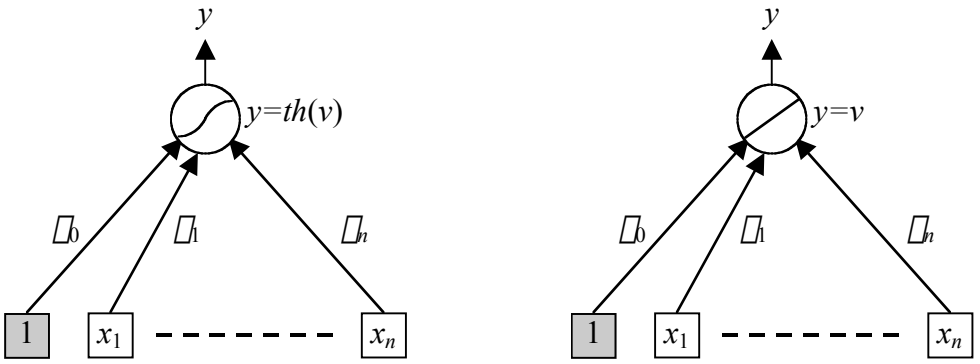

Figure 1.2 : Symboles de neurones à fonctions d'activation sigmoïde (a) et linéaire (b)

Les valeurs des poids du réseau sont en général déterminées par apprentissage ; certaines d'entre elles peuvent être fixées à l'avance si une étude préalable du problème le recommande.
Il existe deux types d'architectures de réseaux de neurones :

- les réseaux non bouclés (ou statiques),
- les réseaux bouclés (ou dynamiques).

1.1.2 Les réseaux de neurones non bouclés

Un réseau de neurones non bouclé réalise une (ou plusieurs) fonctions algébriques de ses entrées par composition des fonctions réalisées par chacun de ses neurones.
Dans un tel réseau, le flux de l'information circule des entrées vers les sorties sans "retour en arrière". Ainsi, si l'on représente le réseau comme un graphe dont les nœuds sont les neurones et les arêtes les connexions entre ceux-ci, le graphe d'un réseau non bouclé est acyclique.
Théoriquement, rien n'interdit de construire un réseau possédant plusieurs sorties. Cependant, dans la pratique, il est généralement souhaitable de décomposer un problème à N_s sorties en N_s problèmes à 1 sortie. En effet, sauf cas particulier[1], il n'y a pas de lien entre les différentes grandeurs que l'on cherche à modéliser.
Dans ce travail, nous ne considérons donc délibérément que des réseaux statiques ne possédant qu'une seule sortie. Par abus de langage, le neurone dont la sortie est la sortie du réseau est appelé "neurone de sortie".
Le nombre de possibilités de connexions de neurones pour former un réseau est infini. Nous ne présentons ici que les deux types de réseaux de neurones les plus fréquemment utilisés :

- les réseaux complètement connectés,
- les réseaux à couche.

Le réseau de neurones à une couche cachée et une sortie linéaire est un cas particulier de ce dernier type.
La figure 1.3 représente l'architecture d'un réseau complètement connecté. Il y a une hiérarchie parmi les neurones cachés : chacun d'entre eux est connecté aux entrées ainsi qu'aux sorties des neurones cachés précédents. Pour caractériser ce réseau, indépendamment des fonctions d'activations qui peuvent différer suivant les neurones, on peut utiliser deux formules équivalentes :

- réseau complètement connecté à N_c neurones,
- ou réseau complètement connecté à N_c-1 neurones cachés et 1 neurones de sortie.

---

[1] Le contre-exemple le plus fréquent est celui de l'utilisation des réseaux de neurones comme classifieurs, dont les sorties sont des estimations des probabilités d'appartenance d'un objet à une classe ; la somme des sorties doit alors être égale à 1.

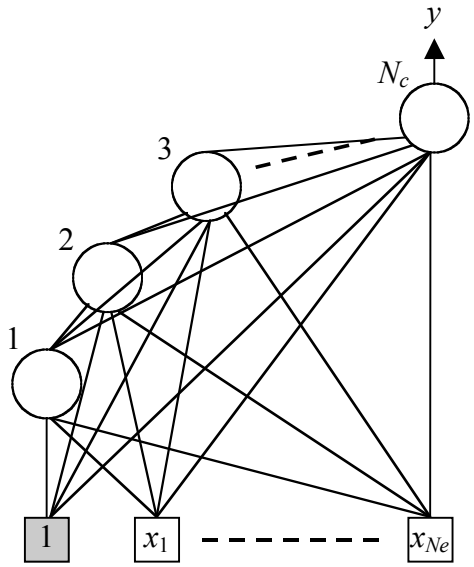

Figure 1.3 : Réseau de neurones non bouclé complètement connecté à N_e entrées et N_c neurones

C'est le type de réseau de neurones non bouclé le plus général possible car il possède, à nombre de neurones donné, le plus grand nombre possible de coefficients étant donné l'interdiction de cycles à l'intérieur du graphe.
Une autre façon d'assembler les neurones entre eux consiste à constituer des couches de neurones en interdisant toute connexion entre neurones de la même couche. Suivant les cas, on pourra décider d'éliminer ou non les connexions entre deux couches non consécutives. Parmi ces dernières, les connexions directes entre les entrées et la sortie (appelés *termes directs*) jouent un rôle particulier car elles permettent de prendre en considération, par la structure même du réseau une influence linéaire de certaines entrées sur la sortie du modèle.
La figure 1.4 présente un réseau de neurones à N_e entrées et une couche possédant N_c neurones cachés sans termes directs.

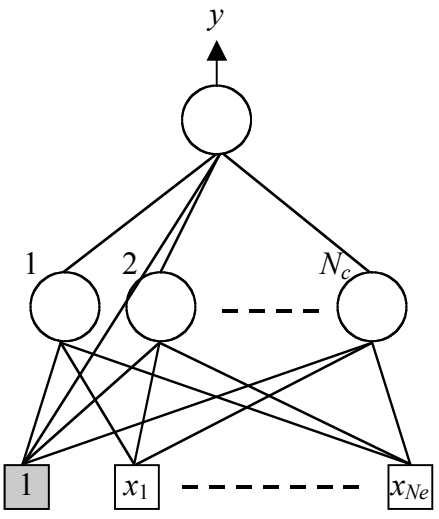

Figure 1.4 : Réseau de neurones non bouclé à N_e entrées et une couche de N_c neurones cachés

Historiquement (voir par exemple [Dreyfus 97]), les réseaux de neurones à couches sont aussi appelés perceptrons multicouches.

L'architecture représentée sur la figure 1.4, avec des neurones cachés sigmoïdaux et un neurone de sortie linéaire est particulièrement utilisée car elle possède des propriétés mathématiques intéressantes, que nous présenterons au paragraphe 1.2.

1.2 Propriété fondamentale des réseaux de neurones non bouclés

La propriété fondamentale des réseaux de neurones est l'approximation parcimonieuse. Cette expression traduit deux propriétés distinctes : d'une part, les réseaux de neurones sont des approximateurs universels et, d'autre part, une approximation à l'aide de réseau de neurones nécessite, en général, moins de paramètres ajustables que les approximateurs usuels.

1.2.1 L'approximation universelle

La propriété d'approximation universelle a été démontrée par [Cybenko 89] et [Funahashi 89] et peut s'énoncer de la façon suivante :
Toute fonction bornée suffisamment régulière peut être approchée uniformément, avec une précision arbitraire, dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche de neurones cachés en nombre fini, possédant tous la même fonction d'activation, et un neurone de sortie linéaire.

Cette propriété est vraie pour les neurones présentés précédemment, c'est-à-dire à fonction d'activation sigmoïdale, mais aussi pour les fonctions radiales et les ondelettes.
C'est cette propriété qui justifie notre choix de l'architecture de réseaux de neurones à une couche cachée. Le seul degré de liberté qui subsiste pour la détermination de l'architecture du réseau est alors le nombre de neurones cachés, ce qui simplifie l'optimisation de l'architecture de réseaux, comme nous le verrons plus loin ; ceci constitue un avantage supplémentaire pour les réseaux de neurones à une couche de neurones cachés.

1.2.2 La parcimonie

Lorsqu'on veut modéliser un processus à partir de données, on cherche toujours à obtenir les résultats les plus satisfaisants possibles avec un nombre minimal de paramètres ajustables. Dans cette optique, [Hornik 94] a montré que :
Si le résultat de l'approximation (c'est-à-dire la sortie du réseau de neurones) est une fonction non linéaire des paramètres ajustables, elle est plus parcimonieuse que si elle est une fonction linéaire de ces paramètres. De plus, pour des réseaux de neurones à fonction d'activation sigmoïdale, l'erreur commise dans l'approximation varie comme l'inverse du nombre de neurones cachés, et elle est indépendante du nombre de variables de la fonction à approcher. Par conséquent, pour une précision donnée, donc pour un nombre de neurones cachés donné, le nombre de paramètres du réseau est proportionnel au nombre de variables de la fonction à approcher.

Ces résultats s'appliquent aux réseaux de neurones à fonction d'activation sigmoïdale, puisque la sortie de ces neurones n'est pas linéaire par rapport à leurs coefficients. Ainsi, l'avantage des réseaux de neurones par rapport aux approximateurs universels (tels que les polynômes) est d'autant plus sensible que le nombre de variables de la fonction à approcher est grand : pour des problèmes faisant intervenir une ou deux variables, on peut généralement utiliser indifféremment des réseaux de neurones ou des polynômes. En revanche, pour des problèmes présentant trois variables ou plus, il est généralement avantageux d'utiliser les réseaux de neurones.
Bien entendu, cette propriété est démontrée de manière générale et pourrait se révéler inexacte pour un problème particulier. Elle constitue néanmoins une justification fondamentale de

l'utilisation des réseaux de neurones, et elle est avérée dans la très grande majorité des problèmes pratiques.
Rappelons que ces résultats concernent l'utilisation de réseaux de neurones pour l'approximation uniforme de fonctions connues ; il est pourtant rare que les réseaux de neurones soient mis en œuvre dans ce cadre. Nous allons montrer dans le paragraphe suivant que la technique des réseaux de neurones est généralement utilisée comme une méthode de modélisation statistique.

1.2.3 De l'approximation de fonction à la modélisation statistique

Les problèmes que l'on rencontre en pratique ne sont que très rarement des problèmes d'approximation de fonction connue. Dans la très grande majorité des cas, on cherche à établir un modèle à partir de mesures, ou, en d'autres termes, à trouver la fonction qui passe "au plus près" (en un sens qui sera précisé plus loin) d'un nombre fini de points expérimentaux, généralement entachés de bruit. On cherche alors à approcher la fonction de régression du processus considéré, c'est-à-dire la fonction que l'on obtiendrait en calculant la moyenne d'une infinité de mesures effectuées en chaque point du domaine de validité du modèle. Le nombre de points de ce domaine étant lui-même infini, la connaissance de la fonction de régression nécessiterait donc une infinité de mesures en un nombre infini de points.
Les réseaux de neurones, en raison de leur propriété fondamentale, sont de bons candidats pour réaliser une approximation de la fonction de régression. C'est ce qui justifie l'utilisation pratique des réseaux de neurones : la recherche d'une approximation de la fonction de régression à partir d'un nombre fini de points expérimentaux.
L'utilisation des réseaux de neurones entre donc complètement dans le cadre de méthodes statistiques d'approximation d'une fonction de régression (voir [Thiria 97]). De telles méthodes ont été largement développées pour les fonctions de régression linéaires. L'apport des réseaux de neurones réside dans leur capacité à modéliser des processus non linéaires.

1.3 Mise en œuvre des réseaux de neurones

La mise en œuvre des réseaux de neurones comporte à la fois une partie conception, dont l'objectif est de permettre de choisir la "meilleure" architecture possible, et une partie de calcul numérique, pour réaliser l'apprentissage d'un réseau de neurones.
Le premier aspect est l'objet principal de cette thèse : nous y reviendrons à partir du chapitre suivant.
Le second volet de la mise en œuvre des réseaux de neurones, l'apprentissage, consiste, à partir d'une architecture de réseau de neurones donnée et des exemples disponibles (la base d'apprentissage), à déterminer les poids optimaux, c'est-à-dire les poids tels que la sortie du modèle s'approche le plus possible de la fonction de régression inconnue.
Afin de détailler les principes fondamentaux de l'apprentissage, il est nécessaire d'introduire certaines notations, qui seront utilisées tout au long de ce mémoire.
Les processus considérés ici comportent n entrées non aléatoires, représentées par un vecteur $\boldsymbol{x} = {}^{t}[x_1, ..., x_n]$, et une sortie y_p considérée comme la réalisation d'une variable aléatoire Y_p.
On suppose qu'on peut écrire un modèle adéquat sous la forme :

$$y_p = r(\boldsymbol{x}) + w \qquad (1.4)$$

où w est un bruit d'espérance mathématique nulle et $r(\boldsymbol{x}) = \mathrm{E}(y_p \mid \boldsymbol{x})$ est la fonction de régression (inconnue). Pour estimer les paramètres du modèle, on dispose d'une base de

données de N exemples $\{\boldsymbol{x}^k, y_p{}^k\}_{k=1, ..., N}$. Dans tout ce qui suit, tous les vecteurs sont des vecteurs colonne et sont représentés en caractère gras (sauf pour les lettres grecques), comme par exemple les vecteurs $\boldsymbol{x}$ et $\{\boldsymbol{x}^k\}_{k=1, ..., N}$ de dimension n.
Un réseau de neurones, d'architecture fixée, réalise une famille de fonctions $f(\boldsymbol{x}, \boldsymbol{\theta})$, paramétrée par le vecteur des poids $\boldsymbol{\theta}$.
Avec les notations précédentes, l'apprentissage d'un réseau de neurones nécessite différents éléments, que nous allons détailler ci-après.

1.3.1 la fonction de coût

La définition d'une fonction de coût est primordiale, car celle-ci sert à mesurer l'écart entre la sortie du modèle et les mesures faites sur les exemples d'apprentissage. La fonction la plus couramment utilisée, et dont nous nous sommes servi lors de ces travaux, est la fonction dite des "moindres carrés", dont la définition est :

$$J(\boldsymbol{\theta}) = {}^t[\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta})]\,[\boldsymbol{y}_p - \boldsymbol{f}(X, \boldsymbol{\theta})] \qquad (1.5)$$

avec $\boldsymbol{f}(X, \boldsymbol{\theta}) = {}^t[f(\boldsymbol{x}^1, \boldsymbol{\theta}), ..., f(\boldsymbol{x}^N, \boldsymbol{\theta})]$, et X définie comme la matrice des observations ${}^t[\boldsymbol{x}^1, ..., \boldsymbol{x}^N]$ de dimensions (N, n).
L'objectif de l'apprentissage est donc de trouver un vecteur $\boldsymbol{\theta}$ qui minimise cette fonction.

1.3.2 Le calcul du gradient

Sauf dans le cas d'un modèle linéaire par rapport aux paramètres, la minimisation de J s'effectue de manière itérative, par une descente de gradient.
Il faut donc disposer d'une méthode de calcul du gradient de la fonction de coût par rapport aux poids du réseau. L'essor récent des réseaux de neurones tient en partie à la mise au point, par [Rumelhart 86], de l'algorithme de rétropropagation.
Cette méthode - que nous ne détaillerons pas ici - permet de calculer de manière particulièrement simple et rapide, en tout point de l'espace des entrées, le vecteur $\boldsymbol{z} = \dfrac{\partial f(\boldsymbol{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ de taille q, nombre de paramètres ajustables du réseau.
Nous noterons $\boldsymbol{z}^i$ la valeur de $\boldsymbol{z}$ au point $\boldsymbol{x}^i$ et Z la matrice ${}^t[\boldsymbol{z}^1, ..., \boldsymbol{z}^N]$. Cette matrice, de taille (N, q), est appelée matrice jacobienne de la fonction f. Notons que, dans le cas d'un modèle linéaire par rapport aux paramètres, la matrice jacobienne Z est égale à la matrice des observations X.

1.3.3 L'algorithme d'optimisation

L'algorithme d'optimisation permet d'ajuster de manière itérative les poids du modèle de façon à converger vers un minimum de la fonction de coût. Pour ce faire, on calcule la modification des coefficients à l'étape k par la formule suivante :

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \rho_{k-1}\boldsymbol{d}_{k-1} \qquad (1.6)$$

où ρ_{k-1} et $\boldsymbol{d}_{k-1}$ représentent respectivement le pas et la direction de descente.
Ainsi, on part d'une initialisation aléatoire des poids, et l'on poursuit de déroulement de l'algorithme jusqu'à satisfaction d'un critère d'arrêt, portant par exemple sur la norme du vecteur dérivée de J par rapport aux coefficients ou sur le nombre d'itérations. Sauf dans le cas linéaire, pour lequel la fonction de coût est quadratique, J possède plusieurs minima vers lesquels peut converger l'apprentissage, suivant l'initialisation des poids.

Sans entrer dans les détails, les algorithmes utilisés se différencient par le choix du pas et de la direction de descente :

- le pas peut être soit constant, soit ajusté en cours d'apprentissage, par exemple par la méthode de Nash [Nash 90] ou celle de Wolfe et Powell ([Wolfe 69] et [Powell 76]),
- dans un algorithme du premier ordre, la direction de descente est calculée à partir de la dérivée première de la fonction de coût par rapport aux coefficients,
- dans un algorithme du second ordre, on utilise le Hessien (c'est-à-dire la matrice des dérivées secondes de J par rapport aux coefficients), ou une approximation de celui-ci, pour déterminer la direction de descente.

Dans le cas général, il est fortement conseillé d'utiliser des méthodes du second ordre, beaucoup plus efficaces que les méthodes du premier ordre. C'est la raison pour laquelle nous avons utilisé, dans le présent travail, une méthode de quasi-Newton, en l'occurrence BFGS [Press 92], ainsi que l'algorithme de Levenberg-Marquardt ([Levenberg 44] et [Marquardt 63]), sur lequel nous reviendrons dans le chapitre 5.