Table des matières

Introduction

Les réseaux de neurones ont d'abord été développés pour résoudre des problèmes de contrôle, de reconnaissance de formes ou de mots, de décision, de mémorisation comme une alternative à l'intelligence artificielle, et en relation plus ou moins étroite avec la modélisation de processus cognitifs réels et des réseaux de neurones biologiques.
Dans le même temps, de nombreux auteurs ont remarqué qu'il existait des relations étroites entre les modèles neuronaux et la statistique (voir par exemple dans Hertz et al., 1991).

Dans ce cadre, notre travail s'intéresse donc à ces relations. On essaiera d'approcher les réseaux de neurones à des méthodes statistiques plus connues.
Après avoir défini ce que sont les réseaux neuronaux, nous allons successivement les comparer à la régression linéaire, puis à la régression logistique et enfin à l'analyse discriminante.

Chapitre 1

Présentation du CICT : Centre Inter-universitaire de Calcul de Toulouse

1.1 Présentation de l'entreprise

1.1.1 Présentation générale

Le Centre Inter-universitaire de Calcul de Toulouse est un centre de ressources informatiques (C.R.I.), service commun aux établissements universitaires toulousains suivants :

* U.T.1 : Toulouse I, Université des Sciences Sociales
* U.T.M. : Toulouse II, Université de Toulouse Le Mirail
* U.P.S. : Toulouse III, Université Paul Sabatier
* I.N.P.T. : Institut National Polytechnique de Toulouse
* I.N.S.A. : Institut National des Sciences Appliquées.

Destiné principalement aux équipes de recherche scientifique, il est à l'écoute des besoins des universitaires pour jouer un rôle fédérateur dans le domaine de l'équipement informatique. Il analyse ces besoins pour fournir, soit des moyens en complément de leur équipement, soit l'ensemble de la solution informatique.
Créé en 1972 avec une douzaine de personnes, il compte aujourd' hui 35 personnes.

1.1.2 Organisation

Le CICT est administré par un conseil de 39 membres représentant les établissements co-contractants et certains laboratoires et organismes régionaux.
Rattaché administrativement à l'université Paul Sabatier, il est placé sous la responsabilité d'un directeur nommé pour 5 ans, assisté d'un directeur technique. Le directeur est Yves RAYNAUD, professeur d'informatique à l'université Paul Sabatier et le directeur technique est Jean-Pierre SILVAIN.
Le CICT comprend un service administratif et comptable, et des services techniques qui travaillent en étroite collaboration.

1.1.3 Les services offerts

Le CICT offre des services dans de nombreux domaines de l'informatique :

· Mise à disposition de moyens matériels et logiciels

- des réseaux informatiques, des serveurs, de nombreux logiciels (le CICT est un centre de diffusion de logiciels dans le cadre d'une convention passée entre le ministère de l'Education Nationale et Microsoft)
- des imprimantes, un numériseur (scanner)
- plusieurs parcs de terminaux X
- des PC et Macintosh en libre service pour réaliser des transferts de fichiers.

· Formation

Le CICT organise des stages de formation pour ses utilisateurs. Il conçoit et donne des formations spécifiques à ses logiciels et matériels.
Pour répondre à une demande importante, le CICT propose également des stages de formation à l'utilisation de logiciels pour micro-informatique.
Ouvert sur l'extérieur, le CICT organise des formations dispensées par des intervenants extérieurs et peut également concevoir des formations pour des organismes extérieurs.

· Documentation, Assistance

En plus des manuels disponibles à la permanence, de nombreuses documentations sont accessibles "en ligne" sur les ordinateurs, pour consultation ou impression. Il s'agit de documentations fournies par les constructeurs ou écrites par le CICT.
Le CICT assure une assistance aux utilisateurs de micros et de réseaux de micros comprenant 5 axes : conseil, achat, installation, formation, mainte-

nance.

· Développement

Dans certains cas, le CICT peut prendre en charge le développement d'une application informatique.
Il est également associé à un projet d'enseignement multimédia à distance avec le service de la formation continue de l'UPS, l'Aérospatiale et Hewlett Packard : le projet FUDMIP (Formation Universitaire à Distance en Midi-Pyrénées) et un projet européen d'enseignement multimédia à distance : le projet ARIADNE.

· Exploitation, sauvegardes

Un des rôles les plus importants d'un centre informatique consiste à assurer la sécurité des données qui lui sont confiées (sécurité vis à vis des accidents et vis à vis des pirates). Elle est assurée par des matériels, des logiciels et des procédures de travail.

1.2 Présentation du service

Le stage a été réalisé au sein du service *Applications et Logiciels* dirigé par Mr. Thouzellier. Ce service a pour but d'installer et de suivre les logiciels d'application dans de nombreux domaines (graphiques, bases de données, statistiques, ...). Il s'occupe de la formation et de l'assistance des utilisateurs.
L'étude statistique suivante a été menée en étroite collaboration avec Mr. Saint Pierre, ingénieur, responsable de l'utilisation des logiciels de statistiques au sein de ce groupe.

Chapitre 2

Que sont les réseaux neuronaux ?

Dans le cadre de l'intelligence artificielle, de nouvelles approches apparaissent pour tirer parti de l'évolution des coûts et de la puissance de traitement des ordinateurs face à des problèmes définis de manière plus floue.
L'intelligence artificielle se propose de reconstituer le raisonnement humain soit à partir d'un apport externe de la connaissance et de l'utilisation de règles dans les systèmes experts, soit par l'imitation du fonctionnement du cerveau dans les réseaux neuronaux.

2.1 Les bases biologiques

Le neurone biologique est une cellule vivante spécialisée dans le traitement des signaux électriques.
Les neurones sont reliés entre eux par des liaisons appelées axones. Ces axones vont eux mêmes jouer un rôle important dans le comportement logique de l'ensemble. Ces axones conduisent les signaux électriques de la sortie d'un neurone vers l'entrée (synapse) d'un autre neurone.
Les neurones font une sommation des signaux reçus en entrée et en fonction du résultat obtenu, vont fournir un courant en sortie.

2.2 Origine et concepts de bases des réseaux de neurones artificiels (RNA)

Ainsi, les réseaux de neurones biologiques réalisent un certain nombre d'applications telles que la reconnaissance de formes, le traitement du signal, l'apprentissage par l'exemple, la mémorisation, la généralisation. Ces applications sont pourtant, malgré tous les efforts déployés en algorithmique et en intelligence artificielle, à la limite des possibilités actuelles.

C'est à partir de l'hypothèse que le comportement intelligent émerge de la structure et du comportement des éléments de base du cerveau que les réseaux de neurones artificiels se sont développés. Les RNA sont donc des modèles et à ce titre, ils peuvent être décrit par leur composants, leurs variables descriptives (voir définition paragraphe 2.2.4) et les interactions des composants.

2.2.1 Définition

Derrière le vocable de RNA se cache une grande diversité d'architectures de réseaux et d'algorithmes d'apprentissage. La définition proposée ci-dessous (Hecht-Nielsen, 1990) paraît bien rendre compte de l'état actuel du concept de réseau de neurones artificiels :

"Un RNA est une structure de traitement de l'information parallèle et distribuée, constituée d'unités de calcul (les neurones) interconnectées par des réseaux unidirectionnels appelés "connexions". Chaque unité de calcul n'a qu'une seule connexion de sortie qui peut être dupliquée en autant d'exemplaires que désiré, les duplicata transportant le même signal."

2.2.2 Le neurone artificiel

Le neurone artificiel (ou cellule) est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones appartenant à un niveau situé en amont (on parlera de neurones "amonts"). A chacune des entrées est associée un poids w représentatif de la force de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones appartenant à un niveau situé en aval (on parlera de neurones "avals"). A chaque connexion est associée un poids.

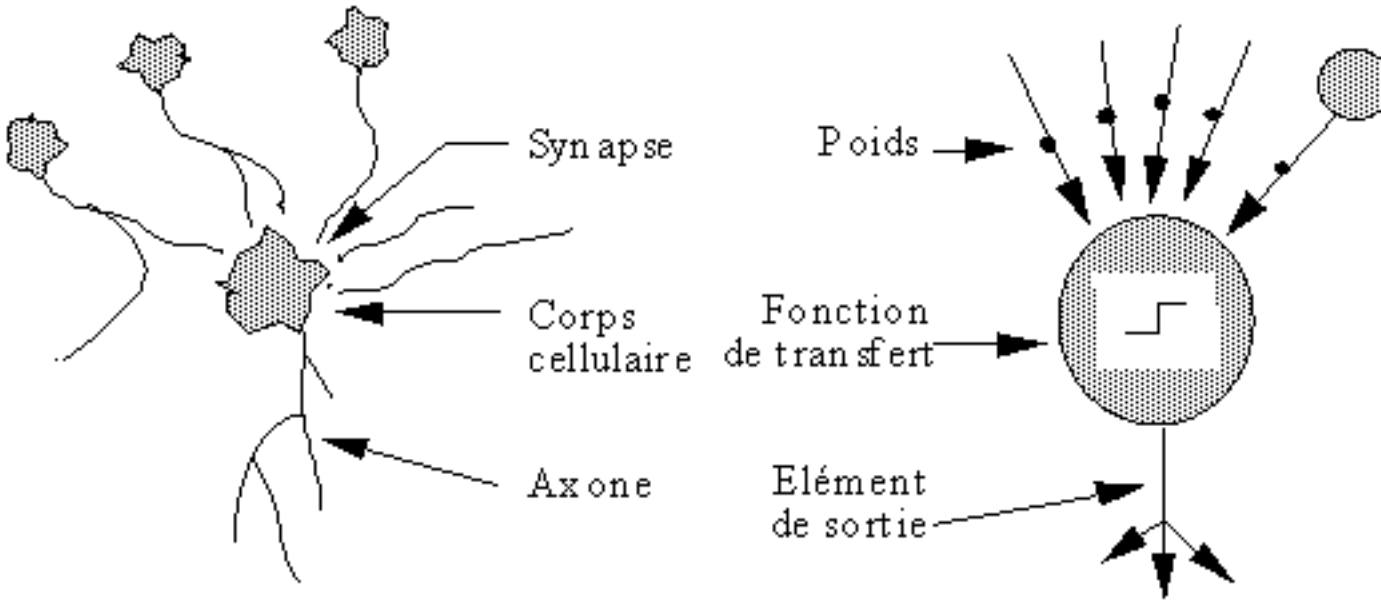

FIG. 2.1 – *Mise en correspondance neurone biologique / neurone artificiel*

2.2.3 Variables descriptives

Ces variables décrivent l'état du système. Dans le cas des réseaux de neurones qui sont des systèmes non autonomes, un sous-ensemble des variables descriptives est constitué par les variables d'entrée, variables dont la valeur est déterminée extérieurement au modèle à savoir le réseau.

2.2.4 Structure d'interconnexion

Les connexions entre les neurones qui composent le réseau décrivent la "topologie du modèle". Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité.

Réseau multicouche : les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones de couches avales. Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc définir les concepts de neurone d'entrée, neurone de sortie. Par extension, on appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelées couches cachées.

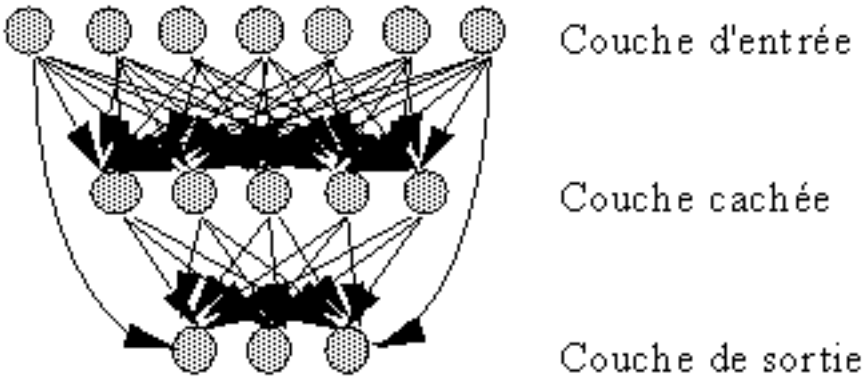

FIG. 2.2 – *Définition des couches d'un réseau multicouche*

Réseau à connexions locales : Il s'agit d'une structure multicouche, mais qui à l'image de la rétine conserve une certaine topologie. Chaque neurone entretient des relations avec un nombre réduit et localisé de neurones de la couche avale. Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique.

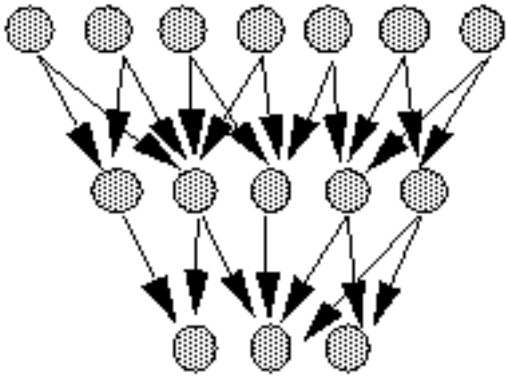

FIG. 2.3 – *Réseau à connexions locales*

Réseau à connexions récurrentes: les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales.

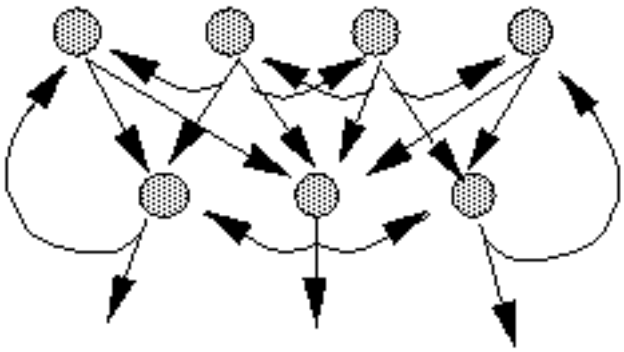

FIG. 2.4 – *Réseau à connexions récurrentes*

Réseau à connexion complète: c'est la structure d'interconnexion la plus générale. Chaque neurone est connecté à tous les neurones du réseau (et à lui-même).

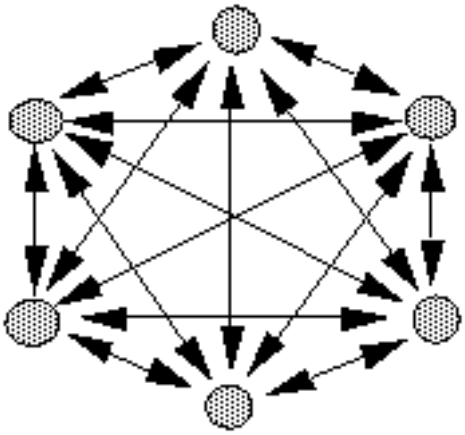

FIG. 2.5 – *Réseau à connexion complète*

2.3 Apprentissage

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Elle ne concerne cependant pas tous les modèles,

mais les plus utilisés.

Définition : L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.

Durant cette phase de fonctionnement, le réseau adapte sa structure (le plus souvent, les poids des connexions) afin de fournir sur ses neurones de sortie les valeurs désirées. Cet apprentissage nécessite des exemples désignés aussi sous l'appellation d'échantillon d'apprentissage ainsi qu'un algorithme d'apprentissage. Après initialisation des poids du réseau (en général des valeurs aléatoires), il y a présentation des exemples au réseau et calcul des sorties correspondantes. Une valeur d'erreur ou de correction est calculée et une correction des poids est appliquée.

L'apprentissage est dit supervisé lorsque les exemples sont constitués de couples de valeurs du type : (valeur d'entrée, valeur de sortie désirée). Tout le problème de l'apprentissage supervisé consiste, étant donné un ensemble d'apprentissage E de N couples (entrée-sortie désirée) (x_i, y_i) $i = 1, 2, \ldots, n$, à déterminer le vecteur des poids w d'un réseau F_w capable de mettre ces informations en correspondance, c'est à dire un réseau tel que :

$$F_w(x_i) = y_i,\ i = 1, 2, \ldots, n.$$

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 similaires.

Chapitre 3

Réseau de neurones et Statistiques

3.1 Relation avec la statistique

De nombreux auteurs ont remarqué qu'il existait des relations étroites entre les modèles neuronaux et la statistique (voir par exemple dans (Hertz et al., 1991), (Baldi and Hornik,1989) et(Bourlard and Kamp, 1988)).
Ces relations sont de diverse nature. Dans certains cas, les modèles neuronaux réalisent des taches traditionnelles de la statistique, comme la classification (perceptron, modèle compétitif), la régression et la prévision (perceptron multicouches). D'autre part, les chercheurs ont proposé des méthodes connexionistes qui permettent de réaliser, par des méthodes itératives, des analyses de données "classiques" telles que l'analyse en composantes principales ((Oja, 1982),(Oja, 1989), (Sanger, 1989) et (Földiak, 1989)) et l'analyse disciminante ((Gallinari et al., 1988)).

Dans ce cadre, notre travail s'intéresse plus particulièrement à la comparaison de différentes méthodes statistiques et des réseaux neuronaux. Ainsi, le seul réseau de neurones utilisé est le réseau multicouche.

3.2 Le perceptron multicouches

Le perceptron multicouches est un réseau comportant L couches, chaque neurone d'une couche étant totalement connecté aux neurones de la couche suivante.
On suppose que la rétine comporte un neurone particulier dont la sortie, constante et égale à 1, est reliée à une entrée de tous les neurones des couches supérieures (les poids de connexions issues de ce neurone d'entrée particulier joueront le rôle de seuils pour les automates receveurs). Chaque neurone k est un automate linéaire généralisé dont la fonction de transfert f_k est supposée sigmoïdale.

L'algorithme d'apprentissage par rétro-propagation du gradient de l'erreur est un algorithme itératif qui a pour objectif de trouver le poids des connexions minimisant l'erreur quadratique moyenne commise par le réseau sur l'ensemble d'apprentissage. Cette minimisation par une méthode du gradient conduit à l'algorithme d'apprentissage de rétro-propagation (Lippmann, 1987). Cet algorithme, qui présente l'avantage d'exister, reste discutable dans la mesure où sa convergence n'est pas prouvée. Son utilisation peut conduire à des blocages dans un minimum local de la surface d'erreur. Son efficacité dépend, en effet, d'un grand nombre de paramètres que doit fixer l'utilisateur : le pas du gradient, les paramètres des fonctions sigmoïdes des automates, l'architecture du réseau ; nombre de couches, nombre de neurones par couche ..., l'initialisation des poids ...

3.3 Modèle mathématique du perceptron multicouches

On dispose d'une variable quantitative ou d'une variable qualitative (notée y) à q modalités que l'on doit prédire à partir de p variables $(x_1, x_2, \ldots, x_p)$ prédictrices. On dispose par ailleurs de n individus ou observations (échantillon d'apprentissage) décrits par les p variables $(x_1, x_2, \ldots, x_p)$ et pour lesquels ont connaît les valeurs de y. On suppose que la couche d'entrée est formée de p entrées, auxquelles seront appliquées des coefficients appelés les poids synaptiques w_{jm}. De plus, il existe un terme constant en entrée qui, pour des raisons pratiques, prend la valeur 1. La couche cachée comprend c neurones qui seront chacun activés par une intégration (en général fonction monotone de la somme) des p signaux en provenance de la couche d'entrée. La même opération a lieu pour les q éléments de la couche de sortie mettant en jeu les poids synaptiques v_{mk}. Il existe aussi une connexion directe de l'entrée constante à la sortie.
L'introduction de la constante d'entrée unitaire, connectée à chaque neurone situé dans la couche cachée ainsi qu'à chaque sortie, évite d'introduire séparément ce que les informaticiens appellent un biais pour chaque unité. Les biais deviennent simplement parties intégrantes de la série de poids (les paramètres).
En termes de modèle analytique, on écrira :

$$y_k \;=\; \Phi_0\{a_k + \sum_{m=1}^{c} v_{mk}\Phi(a_m + \sum_{j=1}^{p} w_{jm}x_j)\}$$

Dans cette formule, la fonction Φ est la fonction logistique à savoir :

$$\Phi(z) \;=\; \frac{\exp(z)}{1+\exp(z)}$$

La fonction Φ_0 peut être selon les cas linéaire, logistique, ou à seuil.
Remarquons que l'équation correspond à une observation (i). On a en réalité n équations de ce type, faisant chacune intervenir q valeurs $y_k^{(i)}$ et p valeurs $x_j^{(i)}$.
L'estimation des paramètres se fait en minimisant une fonction de perte, qui peut simplement être la somme des carrés des écarts entre les valeurs calculées $\tilde{y}_k^{(i)}$ et les valeurs observées $y_k^{(i)}$ dans l'échantillon d'apprentissage.

3.4 Comparaison de termes utilisés en statistiques et dans les RNA

Réseaux de neurones	**Statistiques**
apprentissage	estimation
poids	paramètres
connaissance	valeur des paramètres
apprentissage supervisé	régression
classification	discrimination
apprentissage non supervisé	estimation de densité
clustering	classification
réseau de neurone	modèle
grand : 100 000 poids	grand : 50 paramètres
ensemble d'apprentissage	échantillon
grand : 500 000 exemples	grand : 200 cas

Chapitre 4

Mise en oeuvre des réseaux neuronaux

Nous allons suivre une démarche reprise par Wierenga et Kluytmans (1994) qui est composée de quatre étapes principales :

4.1 Étape 1 : fixer le nombre de couches cachées

Mis à part les couches d'entrée et de sortie, l'analyste doit décider du nombre de couches intermédiaires ou cachées. Sans couche cachée, le réseau n'offre que de faibles possibilités d'adaptation ; avec une couche cachée, il est capable, avec un nombre suffisant de neurones, d'approximer toute fonction continue (Hornik, 1991). Une seconde couche cachée prend en compte les discontinuités éventuelles.

4.2 Étape 2 : déterminer le nombre de neurones par couches cachées

Chaque neurone supplémentaire permet de prendre en compte des profils spécifiques des neurones d'entrée. Un nombre plus important permet donc de mieux coller aux données présentées mais diminue la capacité de généralisation du réseau. Ici non plus il n'existe pas de règle générale mais des règles empiriques. La taille de la couche cachée doit être :
· soit égale à celle de la couche d'entrée (Wierenga et Kluytmans, 1994),
· soit égale à 75% de celle-ci (Venugopal et Baets, 1994),
· soit égale à la racine carrée du produit du nombre de neurones dans la couche d'entrée et de sortie (Shepard, 1990).
Notons que le dernier choix réduit le nombre de degrés de liberté laissés au réseau, et donc la capacité d'adaptation sur l'échantillon d'apprentissage, au profit d'une plus grande stabilité/capacité de généralisation.

Une voie de recherche ultérieure consisterait soit à procéder à l'estimation d'un réseau comportant de nombreux neurones puis à le simplifier par l'analyse des multicolinéarités ou par une règle d'apprentissage éliminant les neurones inutiles ; soit à définir une architecture tenant compte de la structure des variables identifiée au préalable par une analyse en composantes principales.

4.3 Étape 3 : choisir la fonction d'activation

Nous considérerons la fonction logistique pour le passage de la couche d'entrée à la couche cachée. Le passage de cette dernière à la couche de sortie sera soit linéaire, soit logistique selon nos types de variables.

4.4 Étape 4 : choisir l'apprentissage

L'apprentissage de rétro-propagation nécessite la détermination du paramètre d'ajustement des poids synaptiques à chaque itération.
La détermination du critère d'arrêt est aussi cruciale dans la mesure où la convergence peut passer par des minima locaux.

Chapitre 5

Première approche pratique : étude de deux fichiers sous le logiciel Splus

5.1 La fonction "nnet()" de Splus

La fonction nnet de Splus utilise les réseaux multicouches à une seule couche cachée. Dans ce cas, le nombre de paramètres est donné par : $s = (p+2)c+1$, où p représente le nombre de variables d'entrée (en excluant la constante) et c le nombre de neurones cachés.

La commande est la suivante :
nnet(x, y, weights, size, Wts, linout=F, entropy=F, softmax=F, skip=T, decay=0, maxit=100)
Elle prend nécessairement comme arguments :
· **x** : la matrice des variables d'entrée (au nombre de p)
· **y** : la matrice des variables de sortie
· **size** : le nombre de variables dans la couche cachée qui vaut c. Remarquons que ce nombre peut être nul s'il existe une relation directe entre les variables d'entrée et les variables de sortie
· **weights** : le poids de chaque connexion (par défaut, il vaut 1).

Il existe aussi des arguments optionnels :
· la fonction de transfert Φ_0 qui peut être soit la fonction **linout**, soit la fonction **entropy**, soit la fonction **softmax** définies dans Splus (voir paragraphe 5.2 pour une explicitation de ces fonctions).
· l'option **skip** qui permet une influence directe des variables d'entrée.
· l'option **maxit** qui donne le nombre maximum d'itérations (par défaut, il vaut 100).
· l'option **decay** qui est une sorte de pénalisation. Son choix semble cru-

cial et il n'existe pas jusqu'à présent une méthode permettant de trouver la bonne valeur. Il est conseillé d'essayer plusieurs valeurs afin de voir si des résultats significatifs sont obtenus. Par défaut, elle vaut 0.

5.2 Étude d'un premier fichier de données

5.2.1 Présentation des données

Le fichier de données concernent les températures de chaque mois pour différentes villes de France.
Le but de cette étude est d'expliquer les températures du mois de décembre en fonction de celles des mois de janvier, avril et juillet grâce aux réseaux neuronaux et au logiciel Splus.

	Janv	Feb	Mars	Avr	Mai	Jun	Juil	Aou	Sep	Oct	Nov	Dec
ajac	7.7	8.7	10.5	12.6	15.9	19.8	22.0	22.2	20.3	16.3	11.8	8.7
ango	4.6	5.4	8.9	11.3	14.5	17.2	19.5	19.4	16.9	12.5	8.1	5.3
besa	1.1	2.2	6.4	9.7	13.6	16.9	18.7	18.3	15.5	10.4	5.7	2.0
biar	7.6	8.0	10.8	12.0	14.7	17.8	19.7	19.9	18.5	14.8	10.9	8.2
bord	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21.0	18.6	13.8	9.1	6.2
bres	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16.0	14.7	12.0	9.0	7.0
cler	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	3.6
dijo	1.3	2.6	6.9	10.4	14.3	17.7	19.6	19.0	15.9	10.5	5.7	2.1
gren	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	2.3
lill	2.4	2.9	6.0	8.9	12.4	15.3	17.1	17.1	14.7	10.4	6.1	3.5
limo	3.1	3.9	7.4	9.9	13.3	16.8	18.4	17.8	15.3	10.7	6.7	3.8
lyon	2.1	3.3	7.7	10.9	14.9	18.5	20.7	20.1	16.9	11.4	6.7	3.1
mars	5.5	6.6	10.0	13.0	16.8	20.8	23.3	22.8	19.9	15.0	10.2	6.9
mont	5.6	6.7	9.9	12.8	16.2	20.1	22.7	22.3	19.3	14.6	10.0	6.5
nanc	0.8	1.6	5.5	9.2	13.3	16.5	18.3	17.7	14.7	9.4	5.2	1.8
nant	5.0	5.3	8.4	10.8	13.9	17.2	18.8	18.6	16.4	12.2	8.2	5.5
nice	7.5	8.5	10.8	13.3	16.7	20.1	22.7	22.5	20.3	16.0	11.5	8.2
orle	2.7	3.6	6.9	9.8	13.4	16.6	18.4	18.2	15.6	10.9	6.6	3.6
pari	3.4	4.1	7.6	10.7	14.3	17.5	19.1	18.7	16.0	11.4	7.1	4.3
perp	7.5	8.4	11.3	13.9	17.1	21.1	23.8	23.3	20.5	15.9	11.5	8.6
reim	1.9	2.8	6.2	9.4	13.3	16.4	18.3	17.9	15.1	10.3	6.1	3.0
renn	4.8	5.3	7.9	10.1	13.1	16.2	17.9	17.8	15.7	11.6	7.8	5.4
roue	3.4	3.9	6.8	9.5	12.9	15.7	17.6	17.2	15.0	11.0	6.8	4.3
stra	0.4	1.5	5.6	9.8	14.0	17.2	19.0	18.3	15.1	9.5	4.9	1.3
tlse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	5.5

Dans notre exemple, nous avons donc :
· trois variables d'entrée (donc p=3) : x^1 , x^2 , x^3 où x^1 désigne la température

de janvier,x^2 celle d'avril et x^3 celle de juillet
· une variable de sortie y désignant la température de décembre.

5.2.2 Réseau de neurones sans couche cachée

Analogie entre le perceptron multicouches sans couche cachée et la régression linéaire

Le perceptron multicouches sans couche cachée peut être considérer comme une régression linéaire (avec une fonction d'activation linéaire). Nous allons donc, dans ce paragraphe, comparer la prédictivité d'un tel réseau avec celle de la régression linéaire.

Dans ce cas, le passage des variables d'entrée à la variable de sortie est linéaire. On a donc :

$$y_i = \alpha_0 + \sum_{r=1}^{p} \alpha_r x_i^r$$

Remarquons que cette équation correspond bien à l'écriture du modèle de régression où :
· y est la variable "réponse"
· $x^1, x^2 \ldots x^r$ sont des variables réelles. On veut étudier leur influence sur y.

Comparaison de ces deux méthodes dans notre exemple

$*$ Pour le réseau de neurones, on a ici :
· comme variables d'entrée la matrice "temp1[,1:3]"
· comme variable de sortie yy
· pas de variables dans la couche cachée. Donc "$size = 0$"
· une influence directe des variables d'entrée. Donc "$skip = T$"
· une fonction d'activation linéaire. Donc "$linout = T$"

D'où la commande Splus suivante :

```
> rl1_nnet(temp1[,1:3],yy,size=0,decay=0,linout=T,skip=T,maxit=100)
```

On obtient ainsi les résultats suivants :

```
# weights:  4
initial  value 1215.329472
iter  10 value 124.655154
final  value 124.655146
converged
```

Ici, 4 est le nombre de paramètres. L'algorithme converge vers la valeur 124.65 qui correspond à la valeur minimale de l'erreur quadratique.

Grâce à la commande "summary", on peut obtenir les poids des différentes connexions qui correspondent en fait à l'estimation des paramètres $\alpha_r,\ r = 0, 2, \ldots, p$

```
> summary(rl1)
a 3-0-1 network with 4 weights
options were - skip-layer connections  linear output units
  0->4  1->4  2->4  3->4
  7.69 -0.05  1.12 -0.75
```

On a ainsi :
$\hat{\alpha}_0 = 7.69$
$\hat{\alpha}_1 = -0.05$
$\hat{\alpha}_2 = 1.12$
$\hat{\alpha}_3 = -0.75$

$*$ Pour la régression linéaire, on a les résultats suivants :

```
Call: lm(formula = V1 ~ X2 + X3 + X4, data = mat.rl)
Residuals:
    Min      1Q  Median    3Q   Max
 -4.158 -1.339 -0.5256 1.55 3.862

Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  7.6897  5.2681     1.4597  0.1592
         X2 -0.0519  0.5481    -0.0946  0.9255
         X3  1.1208  1.9693     0.5691  0.5753
         X4 -0.7534  1.0825    -0.6960  0.4941

Residual standard error: 2.436 on 21 degrees of freedom
Multiple R-Squared: 0.05277
F-statistic: 0.39 on 3 and 21 degrees of freedom, the p-value is 0.7614
```

On remarque que nous avons les mêmes estimations des paramètres.
A l'aide de la commande "predict" et du résultat de la régression linéaire, on calcule la somme des carrés de l'erreur à savoir :

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

```
> critere_yy-predict(rl2,mat.rl)
> critere_critere^2
> critere_sum(critere)
[1] 124.6552
```

On remarque aussi que nous retrouvons le critère à minimiser dans le réseau de neurones.

Conclusion

On a ainsi, par cet exemple, montré que le perceptron multicouches sans couche cachée était équivalent au modèle de régression linéaire.

5.2.3 Réseau de neurones avec une couche cachée

Détermination du nombre de neurones dans la couche cachée

Étant donné que nous disposons de trois variables d'entrée et d'une variable de sortie, le nombre de neurones dans la couche cachée sera :
· soit 3
· soit 75% de 3 c'est à dire 2.25. Donc 2 neurones.
· soit $\sqrt{3}$ c'est à dire 1.73. Donc 1 neurone.

Notons que la variable z^j de la couche cachée s'exprime par :

$$z_i^j = \frac{\exp(\alpha_0^j + \sum_{r=1}^{p} \alpha_r^j x_i^r)}{1 + \exp(\alpha_0^j + \sum_{r=1}^{p} \alpha_r^j x_i^r))} \; i = 1, 2, \ldots, n \; et \; j = 1, \ldots, c$$

Utilisation de l'option "linout" sans influence des variables d'entrée

· Dans ce cas, le passage de la couche cachée à la variable de sortie est linéaire.
On a donc :

$$y_i = \alpha_0^* + \sum_{j=1}^{c} \beta_j^* z_i^j$$

· Considérons d'abord le nombre de neurones dans la couche cachée égal à deux et faisons varier la valeur du paramètre "decay".

Premier essai : decay=0.

```
> res11_nnet(temp1[,1:3],yy,size=2,decay=0,linout=T,maxit=100)
```

En raison de redémarrages consécutifs de l'algorithme d'ajustement, avec différents points de départ aléatoires pour les poids, trois minima ont été trouvé : 131.59, 111.79 et 79.85.

```
 # weights:  11
initial  value 684.716345
final  value 131.599985
```

```
converged

# weights:  11
initial  value 601.536740
iter  10 value 131.596179
iter  20 value 123.558993
iter  30 value 111.922908
iter  40 value 111.793906
final  value 111.793888
converged

# weights:  11
initial  value 638.150206
iter  10 value 130.223657
iter  20 value 116.147958
iter  30 value 112.617872
iter  40 value 95.387017
iter  50 value 80.762905
iter  60 value 79.992882
iter  70 value 79.962711
final  value 79.957258
converged
```

On vérifie que la "Hessienne" est positive pour les trois minima afin de s'assurer qu'ils représentent bien des minima locaux.
Même si nous relançons plusieurs fois l'algorithme à partir de différents points de départ, il n'est pas garanti que le troisième minima (valeur la plus petite) soit le meilleur.
Les valeurs prédites pour les trois modèles obtenus sont toutes différentes et aucune ne correspond aux vraies valeurs de la variable *yy* à expliquer.

```
Comparaison des valeurs prédites pour le troisième minimum avec la variable yy

> cbind(predict(res11,temp1[,1:3]),yy)
            V1   yy
ajac 4.9198174   8.7
ango 4.9544435   4.9
besa 4.0312395   5.3
biar 4.9198174   2.0
bord 8.2998409   8.2
bres 4.9198174   6.2
cler 6.9975820   7.0
dijo 4.0312395   3.6
gren 4.0312395   2.1
lill 0.6690568   0.5
```

```
limo 4.9256830   2.3
lyon 4.0312467   3.5
mars 4.9198174   3.8
mont 4.9198174   3.1
nanc 4.0312395   6.9
nant 4.9198174   6.5
nice 4.9198174   1.8
orle 5.4199123   5.5
pari 8.2825470   8.2
perp 4.9198174   6.5
reim 3.6450274   3.6
renn 4.9198174   4.3
roue 4.9198198   8.6
stra 4.0312395   3.0
tlse 4.9198174   5.4
```

Ici, V1 représente le vecteur des valeurs prédites de yy à savoir les températures du mois de décembre dans différentes villes de France. Il est obtenu à l'aide de la commande "predict". La commande "cbind" nous permet de concaténer par colonne les vecteurs yy et V1.

Remarquons que si l'on considère les poids fournis par le premier réseau, on obtient le même minimum, les mêmes poids (heureusement !) et les mêmes prédictions.
Pour le premier minimum obtenu, on a comme poids :

```
summary(res11)
a 3-2-1 network with 11 weights
options were - linear output units
  0->4  1->4  2->4  3->4  0->5  1->5  2->5  3->5  0->6  4->6  5->6
  0.34  0.66  0.17  0.64 -0.23 -0.01 -0.16 -0.68  2.09  2.77  0.47
```

Second essai : decay=0.001
Dans ce cas aussi, nous avons trois minima qui sont 112.18, 115.72 et 131.62. Notons que le troisième minimum obtenu ici est aussi un minimum obtenu dans le premier essai et nous obtenons par la même les mêmes prédictions. Quant aux deux autres minima, les valeurs prédites ne sont pas bonnes également.

· Considérons maintenant le nombre de neurones dans la couche cachée égal à 3 et faisons varier la valeur du decay.
Ici, nous avons pris les mêmes valeurs du paramètre "decay" que précédemment et dans les deux cas, on obtient trois minima dont l'un (la valeur étant égale à 131.6) est celui obtenu par les deux réseaux antérieurs.

Supposons maintenant qu'il existe un lien entre les variables d'entrée et la variable de sortie.
Alors, le passage de la couche cachée à la variable de sortie s'écrit :

$$y_i = \alpha_0^* + \sum_{j=1}^{c} \beta_j^* z_i^j + \sum_{r=1}^{p} \alpha_r^j x_i^r$$

Pour un nombre quelconque de neurones cachés (inférieur ou égal à trois), on obtient différents minima et les valeurs prédites pour les différents réseaux sont aussi mauvaises.

Que peut-on donc en conclure ?
Comme on n'obtient pas les bonnes valeurs prédites, on peut donc émettre la conclusion suivante à savoir que les températures des mois de janvier, avril et juillet n'expliquent pas celles du mois de décembre.

Utilisation de l'option entropy et de l'option softmax

· Cas de l'option entropy
Le passage de la couche cachée à la variable de sortie s'exprime par :

$$y_i = \frac{\exp(\alpha_0^* + \sum_{j=1}^{c} \beta_j^* z_i^j)}{1 + \exp(\alpha_0^* + \sum_{j=1}^{c} \beta_j^* z_i^j))} \; i = 1, 2, \ldots, n$$

· Cas de l'option softmax
Le passage de la couche cachée à la variable de sortie s'exprime par :

$$y_i = \frac{\exp(\alpha_0^* + \sum_{j=1}^{c} \beta_j^* z_i^j)}{\sum_c(\alpha_0^* + \sum_{j=1}^{c} \beta_j^* z_i^j))} \; i = 1, 2, \ldots, n$$

Dans ces deux formules, il n'y a pas d'influence des variables d'entrée.
· Avec ces deux options, les valeurs prédites valent toujours 1.
En effet, les valeurs prédites sont toujours comprises entre 0 et 1 même si les valeurs observées de y sont négatives ou supérieures à 1.

5.2.4 Conclusion

Pour les variables quantitatives, il est préférable d'utiliser l'option linout surtout s'il n'y a qu'une variable à expliquer. En effet, pour les deux autres options, les valeurs prédites sont toujours comprises entre 0 et 1.

5.3 Étude du second fichier de données

Cette seconde étude porte sur un exemple fictif de n=16 individus et de 7 variables. Les 4 premières x_1, x_2, x_3, x_4 sont des variables quantitatives et les 3 autres y_1, y_2, y_3 représentent les probabilités d'appartenir à un ensemble ou pas. Leur somme vaut 1.
Voici le fichier de données :

	x1	x2	x3	x4	y1	y2	y3
1	1	1	0.00	0.00	0.72	0.12	0.16
2	2	8	0.69	2.08	0.00	0.40	0.60
3	3	2	1.10	0.69	0.83	0.03	0.14
4	4	13	1.39	2.56	0.00	0.80	0.20
5	5	4	1.61	1.39	0.52	0.14	0.34
6	6	12	1.79	2.48	0.02	0.13	0.85
7	7	3	1.95	1.10	0.92	0.00	0.08
8	8	11	2.08	2.40	0.05	0.45	0.50
9	9	5	2.20	1.61	0.69	0.21	0.10
10	10	6	2.30	1.79	0.55	0.15	0.30
11	1	7	0.00	1.95	0.00	0.15	0.85
12	12	6	2.48	1.79	0.73	0.12	0.15
13	3	4	1.10	1.39	0.18	0.54	0.28
14	11	10	2.40	2.30	0.20	0.70	0.10
15	2	9	0.69	2.20	0.00	0.30	0.70
16	10	3	2.30	1.10	0.96	0.04	0.00

Le but de l'étude consiste à expliquer la variable y_1 en fonction des 4 variables quantitatives x_1, x_2, x_3, x_4. On a regroupé nos 4 variables explicatives dans une même matrice que l'on appellera par la suite "bin9".

5.3.1 Détermination du nombre de variables dans la couche cachée

On dispose de 4 variables explicatives et d'une variable à expliquer. Donc, on peut avoir :
· 4 variables dans la couche cachée
· 75% de 4 donc 3 variables dans la couche cachée
· $\sqrt{4}$ donc 2 variables dans la couche cachée.

Par la suite, on se limitera à deux variables dans la couche cachée.
On considérera successivement les valeurs du paramètre "decay" égales à 0 puis à 0.01.

On va essayer, dans les différents cas, de chercher les valeurs prédites pour y_1.

5.3.2 Utilisation de l'option linout

· Decay=0

```
> res21b_nnet(bin9,y1,size=2,decay=0,linout=T,maxit=200)
# weights:  13
initial  value 2.692404
iter  10 value 0.081170
iter  20 value 0.060693
iter  30 value 0.048510
iter  40 value 0.021601
iter  50 value 0.006859
iter  60 value 0.003347
iter  70 value 0.003254
iter  80 value 0.003170
iter  90 value 0.003168
iter 100 value 0.003167
final  value 0.003167
converged
```

Grâce à la commande "predict", on obtient les valeurs prédites pour y_1 que l'on compare aux "vraies valeurs de y_1.
On voit ainsi que l'on a une bonne prédiction pour y_1.

```
> cbind(y1,predict(res21b,bin9),y1- predict(res21b,bin9))
     y1  val. predites   ecart
 1 0.72  0.7260235548 -0.0060235548
 2 0.00  0.0011627030 -0.0011627030
 3 0.83  0.8180949092  0.0119050908
 4 0.00  0.0022800944 -0.0022800944
 5 0.52  0.5109104514  0.0090895486
 6 0.02  0.0179234259  0.0020765741
 7 0.92  0.9136744142  0.0063255858
 8 0.05  0.0588501692 -0.0088501692
 9 0.69  0.6992198229 -0.0092198229
10 0.55  0.5885597467 -0.0385597467
11 0.00 -0.0015969445  0.0015969445
12 0.73  0.6994514465  0.0305485535
13 0.18  0.1771581173  0.0028418827
14 0.20  0.1878038198  0.0121961802
15 0.00 -0.0001405524  0.0001405524
16 0.96  0.9706301093 -0.0106301093
```

Pour comparer les valeurs de y_1 avec les valeurs prédites par le réseau, on peut aussi tracer les valeurs de y_1 en fonction des valeurs prédites. Le

graphique ainsi obtenu est linéaire ce qui montre bien une similarité entre les valeurs.

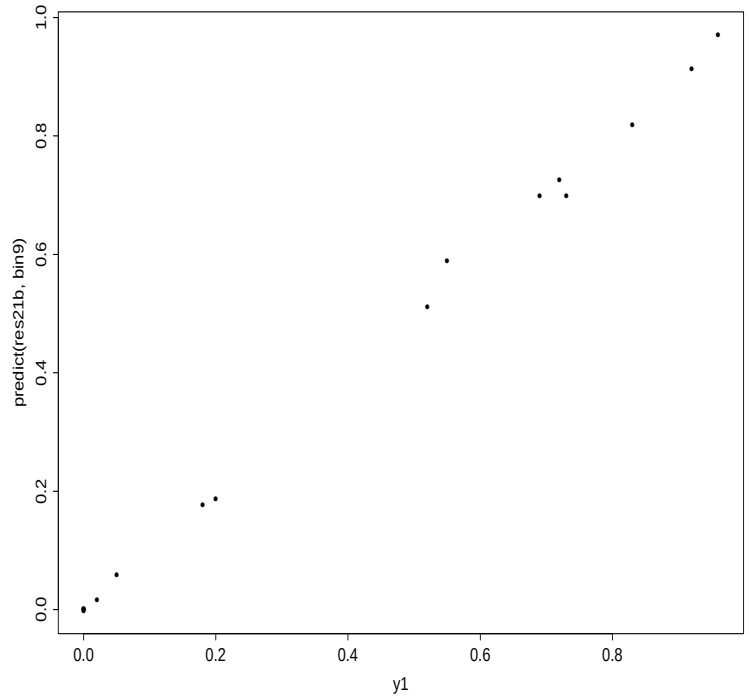

FIG. 5.1 – *Comparaison des valeurs de y_1 et des valeurs prédites obtenues*

· Decay=0.01

Pour cette valeur du "decay", on obtient les valeurs prédites suivantes :

```
> res21c_nnet(bin9,y1,size=2,decay=0.01,linout=T,maxit=200)
> cbind(y1,predict(res21c,bin9),y1- predict(res21c,bin9))
     y1   val. predites   ecart
 1 0.72  0.729235888 -0.009235888
 2 0.00  0.008777049 -0.008777049
 3 0.83  0.769778311  0.060221689
 4 0.00 -0.001242506  0.001242506
 5 0.52  0.494760662  0.025239338
 6 0.02  0.001582627  0.018417373
 7 0.92  0.894366324  0.025633676
 8 0.05  0.019064523  0.030935477
 9 0.69  0.679067373  0.010932627
10 0.55  0.616060615 -0.066060615
```

```
11 0.00  0.011309369 -0.011309369
12 0.73  0.704069793  0.025930207
13 0.18  0.268142045 -0.088142045
14 0.20  0.185469165  0.014530835
15 0.00  0.003297682 -0.003297682
16 0.96  0.986284196 -0.026284196
```

Notons que ces valeurs prédites sont moins précises que celles obtenues avec "decay"=0.

Dans la suite, nous considèrerons la valeur du paramètre "decay" égale à 0.

5.3.3 Utilisation de l'option entropy

Avec l'option entropy, on obtient comme valeurs prédites les valeurs suivantes :

```
> res22b_nnet(bin9,y1,size=2,decay=0,entropy=T,maxit=200)
> cbind(y1,predict(res22b,bin9),y1- predict(res22b,bin9))
     y1 val.predites   ecart
 1 0.72 0.720344484 -3.444839e-04
 2 0.00 0.004017418 -4.017418e-03
 3 0.83 0.830370486 -3.704858e-04
 4 0.00 0.003962993 -3.962993e-03
 5 0.52 0.517451704  2.548296e-03
 6 0.02 0.003988040  1.601196e-02
 7 0.92 0.915684402  4.315598e-03
 8 0.05 0.050039370 -3.936961e-05
 9 0.69 0.699828506 -9.828506e-03
10 0.55 0.547633708  2.366292e-03
11 0.00 0.004001075 -4.001075e-03
12 0.73 0.726522088  3.477912e-03
13 0.18 0.179853752  1.462477e-04
14 0.20 0.200233743 -2.337426e-04
15 0.00 0.003985559 -3.985559e-03
16 0.96 0.961960196 -1.960196e-03
```

On voit que l'on obtient les bonnes valeurs de y_1.

On peut retrouver ces valeurs prédites en appliquant la formule de reconstitution. On a donc :

$$\hat{y_1} = \frac{\exp(\alpha_0^7 + \alpha_5^7 x_5 + \alpha_6^7 x_6)}{1 + \exp(\alpha_0^7 + \alpha_5^7 x_5 + \alpha_6^7 x_6)}$$

$$x_5 = \frac{\exp(\alpha_0^5 + \alpha_1^5 x_1 + \alpha_2^5 x_2 + \ldots)}{1 + \exp(\alpha_0^5 + \alpha_1 x_1^5 + \alpha_2 x_2^5 + \ldots)}$$

$$x_6 = \frac{\exp(\alpha_0^6 + \alpha_1 x_1^6 + \alpha_2 x_2^6 + \ldots)}{1 + \exp(\alpha_0^6 + \alpha_1^6 x_1 + \alpha_2^6 x_2 + \ldots)}$$

```
> summary(res22b)
a 4-2-1 network with 13 weights
options were - entropy fitting decay=0.00
  0->5  1->5  2->5  3->5  4->5  0->6  1->6  2->6 3->6  4->6  0->7  5->7 6->7
  -1.20 -3.9  3.04  -1.13 0.28  0.52  0.19 -0.54 1.26 -1.56 -1.55 -3.97 5.45

ss1_exp(-1.20-3.9*bin9[,1]+3.04*bin9[,2]-1.13*bin9[,3]+0.28*bin9[,4])
zz1_ss1/(1+ss1)
ss2_exp(0.52+0.19*bin9[,1]-0.54*bin9[,2]+1.26*bin9[,3]-1.56*bin9[,4])
zz2_ss2/(1+ss2)
ss_exp(-1.55-3.97*zz1+5.45*zz2)
zz_ss/(1+ss)

On obtient :
> zz
 [1] 0.722576250 0.004056084 0.832753579 0.003997265 0.526203575 0.004025181
 [7] 0.917054818 0.052828969 0.708985285 0.559964739 0.004038044 0.736632025
[13] 0.185221883 0.203269392 0.004021779 0.962311145
```

Ce sont bien les valeurs prédites trouvées par la fonction “predict”.

Chapitre 6

Comparaison de la régression logistique avec les réseaux neuronaux

Cette partie est constituée de la comparaison des réseaux de neurones avec la régression logistique.

6.1 Présentation des données

L'étude qui va suivre est menée par l'équipe Psychologie Sociale du Développement et de la Santé de l'Université de Toulouse Le Mirail. Elle a été faite sur la base d'un questionnaire particulièrement long et fermé, il y a 5 questions "biographiques": le sexe, l'âge, l'année d'étude, la situation familiale et les ressources, et il y a 211 questions qui correspondent à des notes sur des échelles de 1 à 5, ces notes sont des appréciations sur l'importance que le répondant accorde à un mot ou un groupe de mots.

Vu le nombre important des variables, une ACP a été réalisée sur les variables quantitatives et les résultats ont montré que les 5 premiers axes semblaient avoir un intérêt d'interprétation.
On a donc constitué un nouveau fichier de données contenant 8 variables :
· la première variable n'est que le numéro de cas (sujet), il y a 407 observations.
· la deuxième variable est le sexe du sujet : 1 pour garçons et 2 pour filles.
· la troisième variable est la filière d'études : 1 correspond à "Mirail" et 2 à IUT.
· les 5 autres variables sont des scores qui correspondent aux 5 premiers axes de l'ACP.

C'est donc sur ce fichier que nous travaillerons.

Notons une très forte liaison entre le sexe et la filière.

6.2 Travail à effectuer

Compte tenu de la nature du questionnaire, nous allons considérer la variable filière comme une variable à expliquer par les 5 scores. Dans ce cas, nous effectuerons une régression logistique de la filière sur les 5 scores et sur le sexe. Puis nous comparerons les résultats obtenus avec ceux obtenus sur les réseaux de neurones.

6.3 Régression logistique de la filière sur les autres variables

La régression logistique a été effectuée en utilisant la procédure "genmod" du logiciel SAS. Le programme est le suivant :

```
data val2;
set sasuser.val;
filiere=filiere-1;
sexe=sexe-1;
run;
proc genmod data=val2;
class sexe;
model filiere = sexe sc1 sc2 sc3 sc4 sc5
                / link=logit waldci type3;
run;
```

Ici, la table "val" contient le fichier de données et la table "val2" correspond à la table où les variables sexe et filière sont codées 0 ou 1 au lieu de 1 ou 2.

Nous avons, à l'issu de ce programme, obtenus les résultats suivants :

Analysis Of Parameter Estimates

Parameter		DF	Estimate	Std Err	ChiSquare	Pr>Chi
INTERCEPT		1	-1.3014	0.1519	73.4197	0.0001
SEXE	0.00	1	1.2240	0.3117	15.4223	0.0001
SEXE	1.00	0	0.0000	0.0000	.	.

SC1		1	0.0019	0.0004	20.4468	0.0001
SC2		1	0.0001	0.0004	0.0288	0.8652
SC3		1	-0.0018	0.0005	14.1590	0.0002
SC4		1	-0.0013	0.0005	6.6488	0.0099
SC5		1	0.0009	0.0005	3.6411	0.0564
SCALE		1	0.4052	0.0142	.	.

NOTE: The scale parameter was estimated by maximum likelihood.

On voit ainsi que le score 2 (SC2) n'est pas du tout significatif à 5%. On décide donc de la supprimer.
On relance la procédure "genmod" sans cette variable et on obtient :

Analysis Of Parameter Estimates

Parameter		DF	Estimate	Std Err	ChiSquare	Pr>Chi
INTERCEPT		1	-1.2985	0.1506	74.3793	0.0001
SEXE	0.00	1	1.2137	0.3051	15.8246	0.0001
SEXE	1.00	0	0.0000	0.0000	.	.
SC1		1	0.0019	0.0004	20.4816	0.0001
SC3		1	-0.0018	0.0005	14.1465	0.0002
SC4		1	-0.0014	0.0005	6.7501	0.0094
SC5		1	0.0009	0.0005	3.6259	0.0569
SCALE		1	0.4052	0.0142	.	.

A 5%, on enlève aussi la variable SC5. On obtient finalement le modèle suivant :

Parameter		DF	Estimate	Std Err	ChiSquare	Pr>Chi
INTERCEPT		1	-1.2939	0.1519	72.5109	0.0001
SEXE	0.00	1	1.2540	0.3058	16.8143	0.0001
SEXE	1.00	0	0.0000	0.0000	.	.
SC1		1	0.0018	0.0004	19.6101	0.0001
SC3		1	-0.0018	0.0005	14.5947	0.0001
SC4		1	-0.0013	0.0005	6.2643	0.0123
SCALE		1	0.4071	0.0143	.	.

Ainsi, les seules variables qui interviennent dans l'explication de la variable filière sont le sexe et les 3 scores SC1, SC3 et SC4.

6.4 Comparaison avec les réseaux de neurones

6.4.1 Analogie entre la régression logistique et les réseaux de neurones

Le modèle de régression logistique linéaire repose sur l'hypothèse suivante :

$$p_i = \frac{exp(\beta_0 + \beta_1 x_i^1 + \ldots + \beta_k x_i^k)}{1 + exp(\beta_0 + \beta_1 x_i^1 + \ldots + \beta_k x_i^k)}$$

$x^1, \ldots, x^k$ sont les variables explicatives et $\beta = (\beta_0 \ldots, \beta_k)$ est le paramètre inconnu à estimer.

L'équation ci dessous est analogue fonctionnellement à la fonction de transfert softmax dans les réseaux. (cf 5.2.3). Ainsi un réseau de neurones sans couche cachée avec des fonctions softmax, permet d'estimer cette quantité.
L'utilisation d'unité de sortie du type softmax sur un réseau peut être interprétée dans le cadre d'un modèle non linéaire de régression logistique, le réseau servant à estimer les probabilités a posteriori.
Remarquons que pour la régression logistique, l'estimation usuelle est faite par maximum de vraisemblance. Pour les réseaux à unités softmax, on estime également les poids par ce critère.

6.4.2 Réseau de neurones sans couches cachée et à fonction softmax

Dans notre cas, on réalise le réseau avec le logiciel Splus.
On effectue le programme suivant :

```
> filiere_as.factor(table3[,3])
> sexe_as.factor(table3[,2])
> vexp6_cbind(sexe,table3[,4:8])

> resb_nnet(vexp6,filiere,size=0,decay=0,softmax=T,skip=T,maxit=100)
# weights:  7
initial  value 0.000000
final  value 0.000000
converged

>summary(resb)
```

```
a 6-0-1 network with 7 weights
options were - skip-layer connections  linear output units  softmax modelling
  0->7  1->7  2->7  3->7  4->7  5->7  6->7
 -0.64  0.43 -0.28  0.17  0.19 -0.09  0.04
```

Pour le réseau, on utilise donc la fonction softmax. On obtient comme résultats :
· l'algorithme converge vers la valeur 0 qui correspond à la valeur minimale de l'erreur quadratique.
· la commande "summary" nous fournit l'estimation du paramètre β. on voit que ces résultats sont différents de ceux fournis par la régression logistique!!! Ce qui est en contradiction avec le résultat énoncé au paragraphe 6.4.1.

Changeons la valeur du paramètre decay. Prenons decay=0.01 par exemple.

```
> resh_nnet(vexp6,filiere,size=0,decay=0.01,softmax=T,skip=T,maxit=100)
# weights:  7
initial  value 0.012719
final  value 0.012216
converged

> summary(resh)
a 6-0-1 network with 7 weights
options were - skip-layer connections  linear output units  softmax modelling
decay=0.01
  0->7  1->7  2->7  3->7  4->7  5->7  6->7
  0.00 -0.01  0.00  0.00  0.01  0.01 -0.01
```

On voit que les estimations des paramètres varient énormément en fonction de la valeur du paramètre decay. Plusieurs essais ont été effectués mais aucun n'a pu vraiment approché les résultats fournis par la régression logistique.

Nous allons maintenant supprimer les variables SC2 et SC5 et recomparer le nouveau réseau à la régression logistique avec uniquement les variables sexe, SC1, SC3 et SC4. On obtient ainsi les résultats suivants (fournis pour la valeur du paramètre decay=0.01) :

```
> vexp8_cbind(table3[,2],table3[,4:4],table3[,6:7])

> resi_nnet(vexp8,table3[,3],size=0,decay=0.01,softmax=T,skip=T,maxit=100)
# weights:  5
initial  value 0.005569
final  value 0.005349
converged
```

```
> summary(resi)
a 4-0-1 network with 5 weights
options were-skip-layer connections linear output units softmax modelling
decay=0.01
  0->5  1->5  2->5  3->5  4->5
  0.01 -0.01  0.00 -0.01  0.01
```

On voit que l'on n'a toujours pas les mêmes résultats que la régression logistique.

Chapitre 7

Comparaison de deux logiciels sur un fichier de données

Cette partie va consister à comparer les deux logiciels Splus et Matlab sur le même fichier de données tant au niveau des résultats qu'au niveau de la rapidité de la convergence des algorithmes mis en oeuvre.

Sur le même fichier que précédemment, nous allons comparer les résultats des deux logiciels Splus et Matlab.

7.1 Introduction

Dans l'étude qui va suivre, on va transformer les variables qualitatives en matrices d'indicatrices comme suit :

```
mindic1_matind(cbind(table3[2:2],table3[,3:3]))
vfil1_mindic1[,3:4]
vsexe1_mindic1[,1:2]
Regroupons les variables explicatives dans un tableau.
vexp7_cbind(vsexe1,table1[,4:8])
```

Ainsi, la variable à expliquer filière est un tableau à 2 colonnes et le tableau des variables explicatives a 7 colonnes.
D'où le nombre de variables dans la couche cachée peut soit être égal à :
· 7 qui est le nombre de variables en entrée
· 5 qui est la valeur approchée de 75% de 7 (5.25)
· 3 qui est la valeur approchée de $\sqrt{(2 \times 7)}$.

Par la suite, nous considérerons 3 variables dans la couche cachée.

Nous considérerons aussi comme fonction de transfert l'option "entropy".

7.2 Étude sous le logiciel Splus

On obtient les résultats suivants et en ayant choisi la valeur 0.01 pour le paramètre decay :

```
> res3_nnet(vexp7,vfil1,size=3,decay=0.01,entropy=T,maxit=1000)
# weights:  32
initial  value 750.364172
iter  10 value 467.482813
iter  20 value 461.535867
iter  30 value 457.448484
iter  40 value 448.970599
iter 170 value 420.947050
iter 180 value 419.751611
iter 190 value 419.109010
iter 200 value 418.688187
iter 210 value 418.653585
iter 220 value 418.650928
final  value 418.650914

> summary(res3)
a 7-3-2 network with 32 weights
options were - entropy fitting  decay=0.01
  0->8  1->8  2->8  3->8  4->8  5->8  6->8  7->8  0->9  1->9  2->9  3->9
 -0.77  0.00 -0.77  0.35 -0.51 -1.71 -0.37  0.32 -3.94 -1.73 -2.21 -1.99
  4->9  5->9  6->9  7->9 0->10 1->10 2->10 3->10 4->10 5->10 6->10 7->10
  0.78 -0.90 -0.08  0.56  0.54  0.00  0.54 -1.62 -1.60  1.27  1.42 -0.59
  0->11 8->11 9->11 10->11 0->12 8->12 9->12 10->12
   0.57 -0.93  0.89  1.12  -0.57  0.93 -0.89 -1.12
```

7.3 Etude sous le logiciel Matlab

La différence entre le logiciel Matlab et le logiciel Splus est que Matlab permet d'avoir deux couches couchées.
Nous allons d'abord définir les différentes fonctions qui existent dans Matlab puis étudier les réseaux à une et deux couches cachées.

7.3.1 Petite introduction

Différentes fonctions sous Matlab permettent de réaliser l'apprentissage d'un réseau entre autres :
· la fonction **"trainbp"** donne un apprentissage de rétro-propagation
· la fonction **"trainbpx"** fournit aussi un apprentissage de rétro-propagation du gradient mais celui est caractérisé de "rapide".

C'est sur cette fonction que nous travaillerons. Elle est de la forme :

```
[W1,B1,W2,B2,...,TE,TR] = TRAINBPX(W1,B1,F1,W2,B2,F2,...,P,T,TP)
```

Cette fonction prend comme arguments :
· W_i qui représentent la matrice des poids pour la i-ème couche
· B_i qui représentent le vecteur des "biais" (c'est en fait les coefficients de la constante) pour la i-ème couche
· F_i qui définit la fonction de transfert pour la i-ème couche
· P qui représente la matrice des variables d'entrée (explicatives)
· T qui représente la matrice des variables de sortie (à expliquer)
· TP qui représente les paramètres de l'apprentissage (il est optionnel).

Elle retourne :
· les nouveaux poids W_i
· les nouveaux biais B_i
· TE qui représente le nombre d'itérations nécessaire pour l'apprentissage
· TR qui caractérise l'erreur.

Pour déterminer les valeurs prédites des variables explicatives, on utilise la fonction **simuff**.

7.3.2 Réseau à une couche cachée

Nous allons comparer l'étude faite sur le logiciel Splus avec celle faite sur le logiciel Matlab.
Le programme sous le logiciel Matlab se présente comme suit :

```
%Ici, on considere une seule couche cachee a trois variables.

load dd;
X=cat(2,dd(:,2),dd(:,4:8));  % X matrice des variables explicatives
Y=dd(:,3);    %variable filiere a expliquer
%initialisations

EA=X(1:407,:);
```

```
[nla,nca]=size(EA);

P=EA(:,1:nca)'; % entrees
T=Y(1:407,:)';  %sorties
[R,Q]=size(P); %R : nbre entrees
S1=3; %S1 : nbre neurones couche cachee1
[S2,Q]=size(T); %S2 nbre sortie

%initialisation des poids
[W1,B1]=rands(S1,R); % couche entree
[W2,B2]=rands(S2,S1); % couche cachee

freq_aff=1000; %frequence d'affichage
max_iter=2000; %nb max iterations
err_but=0.5; %erreur mini pas trop petite

TP=[freq_aff max_iter err_but];
[W1,B1,W2,B2,Iter,TR]=trainbpx(W1,B1,'logsig',W2,B2,'logsig',P,T,TP);
```

Les résultats sont sous cette forme :

```
TRAINBPX: 0/2000 epochs, lr = 0.01, SSE = 311.97.
TRAINBPX: 1000/2000 epochs, lr = 2.99498e+11, SSE = 113.
TRAINBPX: 2000/2000 epochs, lr = 2.99498e+11, SSE = 113.

W1 =

    0.1700   -0.2272    0.9523   -0.6997    0.5620    0.6408
    0.3716    0.4748   -0.9678    0.7536   -0.9355   -0.4439
    0.9476    1.0040    0.0971    0.1749   -0.8509    1.8452

B1 =

   -0.4790
    0.4270
    0.4502

W2 =

   12.4181   12.6385    4.5830

B2 =
```

```
24.3383
```

7.3.3 Comparaison avec le logiciel Splus

Les résultats ne sont pas les mêmes: on n'a pas la même valeur du minima et les valeurs prédites sont différentes.
Ceci peut être dû au fait que nous n'avons pas lancé plusieurs fois le réseau pour pouvoir mieux approcher la variable à expliquer.
En effet, il faut noter que le temps de calcul est beaucoup plus long en Matlab qu'en Splus. .
Un de ses avantages par rapport à Splus c'est qu'il est plus varié au niveau des fonctions et surtout qu'il peut prendre en considération deux couches cachées.

7.3.4 Réseau à deux couches cachées

L'avantage d'utiliser deux couches cachées c'est que la seconde couche peut prendre en compte des discontinuités éventuelles, mais ceci n'est pas souvent nécessaire.
Il ne va pas sans dire que le temps de calcul est encore plus long.
Les résultats ici ne seront pas fournis : le programme est le même, il suffit de rajouter une couche et les résultats n'apportent rien de concluant.

Chapitre 8

Comparaison de l'analyse discriminante avec les réseaux de neurones

Dans cette partie, nous allons comparer l'analyse discriminante aux réseaux neuronaux.

8.1 Présentation des données

Nos données proviennent d'un organisme bancaire. Une segmentation de ses points de vente a été effectuée en cinq classes.
A l'issue de cette segmentation, l'organisme veut réaliser une analyse discriminante afin de mettre en place un modèle explicatif et un module de réaffectation permettant à l'avenir, de segmenter un nouveau point de vente à sa création mais aussi d'effectuer des réaffectations annuelles pour tenir compte des changements de profils au cours du temps.
Nous disposons ainsi de 16 variables toutes normales (au départ 90), de 82 individus et de notre partition en 5 classes.

8.2 Analyse discriminante

L'analyse discriminante a été effectuée avec le logiciel SAS. Le programme est le suivant :

```
proc discrim data=sasuser.pventend crosslisterr short;
class PIN5;
priors proportional;
run;
```

On obtient ainsi les individus mal classés ainsi que leurs nouvelles classes d'affectation.

Cross-validation Results using Linear Discriminant Function

Obs	From PIN5	Classified into PIN5	Posterior Probability of Membership in PIN5: 1	2	3
			4	5	
11	1	3 *	0.2832	0.0006	0.7162
			0.0000	0.0000	
12	2	3 *	0.0002	0.3060	0.6938
			0.0000	0.0000	
21	3	2 *	0.0001	0.9981	0.0018
			0.0000	0.0000	
29	3	2 *	0.0000	0.7960	0.2039
			0.0000	0.0000	
32	5	1 *	0.7895	0.0000	0.2105
			0.0000	0.0000	
56	1	2 *	0.0001	0.8029	0.1971
			0.0000	0.0000	
58	2	3 *	0.0032	0.0045	0.9923
			0.0000	0.0000	
66	2	4 *	0.0000	0.0000	0.0001
			0.9999	0.0000	
68	2	4 *	0.0000	0.1777	0.0000
			0.8223	0.0000	

* Misclassified observation

Cross-validation Summary using Linear Discriminant Function

Error Count Estimates for PIN5:

	1	2	3	4	5	Total
Rate	0.1053	0.2222	0.0606	0.0000	0.3333	
Priors	0.2317	0.2195	0.4024	0.1098	0.0366	

```
Rate      0.1098

Priors
```

D'après ces résultats, neuf individus sont mal réaffectés, ce qui nous donne un taux d'erreur de 10.98%. Par exemple, le point de vente numéro 11 est réaffecté dans la classe3 au lieu de la classe1.

8.3 Comparaison avec les réseaux de neurones

On peut démontrer qu'un réseau de neurones ayant que des neurones linéaires avec une couche cachée à m unités (où m représente la dimension de l'analyse discriminante) peut être assimilé à une analyse discriminante sur la première couche.

On va donc ici voir ce que font les réseaux neuronaux et comparer les résultats à l'analyse discriminante.

Notre variable qualitative à expliquer est à 5 modalités. Nous allons donc considérer un réseau à 5 variables dans la couche cachée.
Les variables d'entrée sont mises dans le tableau "vexp" et la variable à expliquer est nommée "vpart".

Le logiciel Splus nous donne les résultats suivants :

```
> res2_nnet(vexp,vpart,size=5,decay=0,linout=T,maxit=1000)
# weights:  91
initial  value 531.549007
final  value 75.246429
converged

> varron_round(predict(res2,vexp),digits=0)
[1,] 3 3 1 3 3 3 3 3 3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
[1,]  3  3  3  3  3  3  3  3  1  1  1  3  3  3  3  4  1  3  4  3  3  3  3  3
[1,]  3  3  1  1  3  3  3  3  3  3  3  3  3  3  3  3  3  4  4  1  3  3  3  3
[1,]  3  3  3  3  3  3  3

> vpart
 [1] 3 1 2 3 1 2 2 2 2 3 1 2 3 3 2 3 1 1 3 3 3 3 3 3 2 1 3 3 3 3 5 5 1 1 5 2 1 1
[39] 1 3 1 4 4 1 4 4 3 2 3 4 1 3 4 2 1 1 2 2 2 2 1 3 2 3 3 2 3 2 4 4 1 3 4 3 3 3
[77] 3 1 3 3 3 3
```

D'après les sorties, on voit que 40 individus sur 82 sont mal reclassés ; ce qui correspond quand même à la moitié de la population.

On voit donc que les résultats fournis par l'analyse discriminante sont meilleurs.
Remarquons qu'en changeant la valeur du paramètre decay, les résultats changent mais pas énormément à savoir qu'il y a un ou deux individus de plus ou de moins dans la population des mal classés.

Conclusion

Dans différents domaines d'application, les RNA sont parfois considérés comme un outil magique. L'évolution la plus sensible de leur utilisation, inspirée de celle de l'analyse des données, concerne la possibilité de traiter non seulement des données quantitatives mais aussi des données qualitatives ainsi que des données temporelles.

Les défenseurs des RNA les présentent souvent comme une "boîte noire" qui permet d'explorer des données sans connaissance a priori du sujet traité. Cependant, si cette approche est possible, il faut souligner que le problème de l'optimisation du réseau, de façon à améliorer sa convergence, nécessite le plus souvent une adaptation de son architecture et, dans ce cas, une connaissance a priori du sujet est souvent souhaitable.

Les avantages de ces "boîtes noires" résident dans leur meilleure capacité prédictive issue de la meilleure représentation du phénomène (variables plus nombreuses, relations non linéaires), dans leur capacité d'adaptation et de généralisation au delà de l'échantillon étudié, dans leur absence d'hypothèses sur la distribution des variables (comme la normalité en régression linéaire par exemple) mais aussi dans leur respect de contraintes telles que des bornes $[-1, 1]$ ou $[0, 1]$ sur la variable à expliquer.
Mais il faut aussi noter la difficulté du choix des paramètres pour pouvoir obtenir la bonne convergence de l'algorithme mais aussi pour éviter le risque de sur-apprentissage du réseau.

Les réseaux de neurones, outils séduisants par leur conception radicalement différente, offrent des possibilités nouvelles déjà très appréciées dans des domaines scientifiques et techniques tels que: la reconnaissance des formes, le contrôle qualité, le traitement du signal. Ils fournissent aussi un outil utile dans l'aide à la décision et la prévision : credit scoring, prévision financière notamment.

Ce stage m'a permis de mettre en oeuvre une technique nouvelle qui est les réseaux de neurones mais aussi d'approfondir des méthodes statistiques vues pendant l'année universitaire.

La difficulté rencontrée au cours de ce stage fut de pouvoir mettre en oeuvre les réseaux neuronaux compte tenu du nombre de choix à effectuer (nombre de couches cachées, nombre de neurones par couche, fonction de

transfert) et de leurs conséquences éventuelles sur la qualité des résultats. Il n'existe pas encore, à ma connaissance, de présentation claire des options et de leurs conséquences ce qui semble pourtant constituer une étape indispensable à une utilisation plus large des réseaux de neurones.

Bibliographie

· Networks and Chaos-Statistical and probabilist Aspects *(Chapman and Hall)*

· Pattern Recognition and Neural Networks *(Ripley Brian)*

· Statistiques et Méthodes Neuronales *(Edition Dunod)*

· Neural Network Toolbox User's Guide *(The MathWorks, Inc)*

· Revue Recherche et Applications en Marketing *(volume 11, numéro 2/1996)*

· Classification, analyse des correspondances et méthodes neuronales *(Thèse de Smaïl Ibbou (Paris I, 1998))*

· Apprentissage et Diagnostic de systèmes complexes : réseaux de neurones et réseaux bayésiens *(Thèse de Philippe Leray (Paris 6, 1998))*

· Page Web de Ripley *(http://www.stats.ox.ac.uk/ ripley/)*

· Page Web suivante : *http://www.supelec-rennes.fr/acth/divers/NNad.html*

· Faq sur les réseaux de neurones : *ftp://ftp.sas.com/pub/neural/FAQ.html*