N° d'ordre : 08/2014-M/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Des Sciences et de la Technologie Houari Boumediène

Faculté de Mathématique

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MAGISTER**

En : Probabilité et statistiques

Spécialité : statistiques

Par : M[me] CHATER Amina

Sujet

Etude Comparative Des Trois Modèles De Scoring Sur Données D'entreprise

Soutenu publiquement, le 29/11/2014, devant le jury composé de :

M[r] MOULAI Mustapha	Professeur A l'USTHB	Président
M[me] DJABALLAH Khadidja	Maitre de conférence A l'USTHB	Examinateur
M[r] DJEDOUR Mohamed	Professeur A l'USTHB	Directeur de mémoire

Table de matière

Chapitre 3 : les réseaux de neurones

Remerciements

En premier lieu je remercié الله le tout puissant de m'avoir permis de mener à bien ce travail.

Mes vifs remerciements et ma profonde gratitude vont à tous ceux qui m'ont aidé de près ou de loin à l'élaboration de ce mémoire. Particulièrement ils s'adressent à :

- M^{r} DJEDOUR.M, mon promoteur qui m'a guidé et accompagné avec dévouement tout en long de mon travail. Ses conseils, orientations ainsi que son attention particulier accordé à mon projet.
- M^{r} MOULAI.M pour avoir accepté d'assurer la présidence du jury chargé d'évaluer mon travail.
- M^{me} DJABALLAH.K de m'avoir fait l'honneur d'examiner mon travail.
- Une immense gratitude à l'ensemble des enseignants du département « Statistiques» qui ont contribué à la formation de magister.

Enfin, je remercié toute personne ayant contribué de près ou de loin à l'élaboration de ce modeste travail.

A toutes ces personnes, je leurs dit merci infiniment.

Dédicaces

Grace à dieu je dédie ce modeste travail ; et je tiens à rendre un grand hommage :

A mes parents qui ont œuvré et contribué à ce que j'emprunte la voie des études et du savoir, pour tout leur soutien moral et leur amour et affection.

A mon mari Fayçal et ma fille Asma.

A mon frère Rafik, sa fille Inssaf et ma belle-sœur Ibtissem.

A toute ma famille et ma belle-famille.

A mes collègues du travail.

Enfin, je remercie toutes les personnes, qui de près ou de loin, m'ont aidé à la réalisation de ce travail.

Amina

Chapitre 1
Introduction

1. Généralités sur le crédit scoring

Le Credit Scoring est une discipline relativement jeune ; son apparition remonte à environ soixante (60) ans. En poursuivant le travail pilote de Fischer sur l'analyse discriminante, Durand (1941) fut le premier à reconnaître la possibilité d'utiliser les techniques statistiques pour discriminer entre bons et mauvais emprunteurs.

Fair, Isaac, and Company est habituellement cite comme la firme à avoir développé les premiers systèmes de Credit Scoring pour les crédits de consommation dans les années cinquante aux Etats-Unis. Cette firme continue d'être leader dans l'industrie du Credit Scoring (Thomas and al. 2002).

Depuis lors, l'industrie continue de croître et les domaines d'application des techniques de Scoring s'élargissent au crédit immobilier, au secteur des cartes de crédit, au marketing etc.

Altman (1968) a significativement contribué au développement, à la promotion et à une meilleure compréhension du Credit Scoring et de ses techniques.

Eisenbeis (1996) présente une vue générale sur l'histoire et l'application des techniques de Credit Scoring au portefeuille des banques « business portfolios ». **[1]**

Les développements faits ci-haut concernent surtout les pays développés car l'application du Credit Scoring est assez rare dans les recherches ont été menées pour analyser la probabilité de défaut mais la plupart n'avait pas pour objet de faire des prédictions sur les prêts futurs. **[1]**

Les banques comme beaucoup d'entreprises, sont soumises aux risques. Le risque qui nous intéresse ici est un risque de crédit aussi appelé risque de contrepartie ; s'il existe plusieurs types de risque de crédit, celui de non remboursement est un risque majeur. (Manchon ,2001)

Le risque de contrepartie pour le banquier peut être défini comme le risque de voir son client qui n'a pas respecté son engagement financier, a savoir, dans la plupart des cas, un remboursement de prêt.

Le crédit scoring est une méthode statistique utilisée pour prédire la probabilité qu'un demandeur de prêt ou un débiteur existant fasse défaut.**[2]**

Parmi les modèles utilisés en statistiques et recherche opérationnelle le crédit scoring a eu un grand succès dans le monde de la Finance et des Banques.

Le crédit scoring a vu le jour suite aux travaux pionniers de BEAVER (1966) et D'ALTMAN (1968).

Le premier BEAVER (1966) utilise une méthode de classification dichotomique et observe la capacité de six ratios à classer correctement les entreprises : il s'agit d'un modèle rudimentaire d'analyse discriminante, quasi artisanal. **[2]**

Mais Altman (1968) qui met un point de première fonction score grâce à l'utilisation d'une analyse discriminante multivarie : la fonction Z. **[2]**

2. Les modèles du crédit scoring

L'analyse discriminante, parfois appelé classification supervisée, couvre un large domaine de techniques. Des comparaisons détaillées de ces techniques ont été menées dans plusieurs études.

Des réflexions sur l'adéquation des modèles aux données économiques et financières d'entreprises peuvent être trouvées dans cette littérature. **[6]**

Les principaux modèles étudies sont l'analyse discriminante de Fisher linéaire et quadratique, la régression logistique et quelques méthode non paramétriques comme DISQUAL 2, arbre de décision (CART), CHAID, Foret aléatoires, réseaux de neurones, la méthode du plus proche voisin (k – nearest Neighboor), classification, la méthode du noyau, les supports Vectors Machines .

Le but de la construction d'un score peut se limiter à vouloir identifier les clignotants du risque. Mais si on veut obtenir un seuil opérationnel, sa construction est fondée sur une règle de décision. Celle-ci est généralement définie à partir de la désignation d'un seuil de décision nécessaire à l'évolution des taux de bons classements. Ceux-ci permettent de choisir la meilleure fonction discriminante parmi plusieurs estimations. **[4]**

Dans ce mémoire, nous comparons trois modèles de notion **scoring** basés sur les arbres de décisions, les réseaux de neurones et la régression logistique. Pour l'étude d'une demande de crédit auprès d'une institution bancaire. Le problème est connu sous le nom de **crédit scoring.**

3. Méthodologie

Dans ce qui suit, nous présentons les principales méthodes de credit scoring proposées ce mémoire.

3.1 Les arbres de décision

Les arbres de décision ont été introduits par Breiman et al. (1984). Cette technique a été utilisée à des fins de credit scoring pour la premier fois par Frydman et al. (1985). La technique des arbres de décision est une technique très simple et flexible. Son intérêt majeur réside dans la gestion de données hétérogènes ou manquantes (Tuffery, 2007).

Giudici (2003) définit un arbre de décision comme étant une procédure récursive dans laquelle un ensemble de n individus est progressivement divisés dans des groupes. La procédure commence par le choix d'une variable explicative $X_j \in X$ qui permet de séparer les individus dans des sous-groupes homogènes appelés nœuds. Chaque nœud contient des individus d'une seule classe, l'opération est répétée jusqu'à ce que la division en sous-populations ne soit plus possible.**[9]**

Cette procédure est utilisée par les banques seulement comme un outil de soutien pour les méthodes paramétriques de credit scoring, décrites précédemment, au cours de la phase de sélection de variables (Devaney, 1994). Tufféry (2007) recommande l'utilisation de cette technique que si, on dispose d'un nombre suffisant d'observations pour éviter le risque de sur-apprentissage (i.e. modèle avec d'excellentes performances dans la phase d'apprentissage mais trouve des difficultés de généralisation pour les nouveaux échantillons).

3.2 Les réseaux de neurones [10]

La technique des réseaux de neurones date des travaux de Mcculloch et Pitts (1943). Il s'agit d'un outil d'optimisation non-linéaire composé d'un ensemble d'unités élémentaires, appelées neurones. Chaque neurone reçoit en entrée la description d'une observation i, $\{x^1_i,..., x^d_i\}$, appelées signaux. Ensuite, il effectue une somme pondérée sur ces entrées dont le résultat est soumis à une transformation appelée fonction d'activation notée f (Devaney, 1994). La fonction d'activation d'un neurone est définie par :

$$f(x_i) = \sum_{j=1}^{d} \beta j \, x^j_i + \beta_0 = \beta^T x_i$$

Ou $\{x^1_i,..., x^d_i\}^T$, est le vecteur des entrées. Chaque signal d'apport est associé à un poids β_j, $j=1,...,d$. Le poids détermine l'importance relative du signal d'apport dans la

production de l'élan final transmit par le neurone. La sortie yi du neurone est décidée en fonction du signe de $f(x_i)$. Pour un individu i :

$$y_i = \begin{cases} 0 \text{ si } \beta^T x_i < 0 \\ 1 \text{ si non} \end{cases}$$

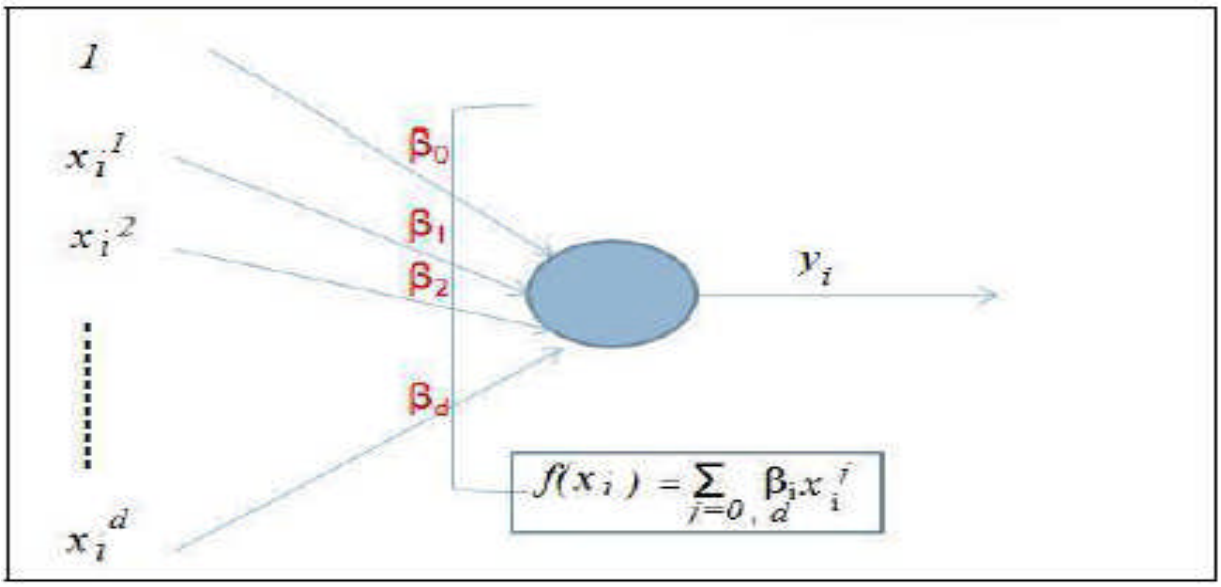


Schéma d'un perceptron à un " neurone"adapté de Cornuéjols (2002)

Ce mécanisme de prédiction permet le traitement des données d'une manière simple sans aucune compétence statistique. Toutefois, cette méthode est une sorte de boîte noire, qui ne permet pas d'expliquer la manière dont on arrive au résultat. Dans le cas des réseaux de neurones, il y a un risque énorme de sur-apprentissage si le nombre d'observations est faible. Cette procédure n'assure toujours pas une convergence vers la solution cherchée et ne permet pas de traiter les problèmes avec un nombre important de variables explicatives.

3.3 La régression logistique

L'origine de cette méthode remonte aux travaux de Cox (1970), (1989). Un modèle de régression logistique fournit une fonction linéaire de descripteurs comme outil de discrimination, Fan et Wang (1998) et Sautory et al. (1992) recommandent l'emploi de la régression logistique binaire, lorsque les conditions d'application de l'analyse discriminante ne sont pas réunies. Ce choix s'impose dans le cas où des variables qualitatives interviennent dans le modèle (Bardos, 2001). **[9]**

Le modèle de régression logistique permet de prédire la valeur prise par la variable dépendante *Y*. Cette technique cherche à déterminer une probabilité a posteriori notée p_i, qui permet d'affecter un individu i à son groupe d'appartenance. Cette probabilité est définie par :

$$P_i = \frac{1}{1 + exp^{-\beta 0 - \beta T x i}}$$

Les coefficients obtenus par cette technique ont les mêmes valeurs que ceux obtenus par l'analyse discriminante linéaire. Néanmoins, ils sont obtenus sous des hypothèses moins contrariantes (Vojtek et Kocenda, 2006). Toutefois, la mise en place de cette technique nécessite une grande masse de données d'apprentissage.

Chapitre 2

Les Arbres De Décisions

On a un ensemble X de N individus notés (x_i) décrits par P attributs quantitatifs ou qualitatifs. Chaque individu x est étiquetté, c'est-a dire qu'il lui est associé une "classe "ou un «attribut cible » que l'on note $y \in Y$.

A partir de ces individus, on construit un arbre dit" de décision" tel que :

- ✓ Chaque nœud correspond a un test sur la valeur d'un ou plusieurs attributs ;
- ✓ Chaque branche partant d'un nœud correspond a une ou plusieurs valeurs de ce test ;
- ✓ A chaque feuille est associée une valeur de l'attribut cible.

De l'arbre de décision on extrait un jeu de règles de décision qui peut être ensuite exploité pour classer (étiquetté) un nouvel individu.**[9]**

1.1 Notations [16]

On notera χ un ensemble de données. Chaque donnée est décrite par un ensemble A d'attributs. Chaque attribut $a \in A$ prend sa valeur dans un certain ensemble de valeurs V_a. Ainsi, on peut considérer l'ensemble des données χ dont les coordonnées balayent toutes les valeurs possibles des attributs : c'est l'espace des données que nous noterons D. Si l'on note a_1, ..., a_P les P attributs,
$D = V_{a1} \times V_{a2} \times \ldots \times V_{ap}$. Toute donnée appartient à cet ensemble et on a $\chi \subset D$
Il est souvent utile d'avoir une représentation géométrique de l'espace des données ; chaque attribut correspondant à un axe de coordonnées. S'il y a P attributs, l'espace des données est un espace euclidien à P dimensions.

1.2 Les différentes natures d'attributs

Une donnée est un enregistrement au sens des bases de données, que l'on nomme aussi "individu" (terminologie issue des statistiques) ou "instance" (terminologie orientée objet en informatique) ou même " tuple" (terminologie base de données) et " point " ou " vecteur " parce que finalement, d'un point de vue abstrait, une donnée est un point dans un espace euclidien ou un vecteur dans un espace vectoriel. Une donnée est caractérisée par un ensemble de "champs ", de" caractères ", ou encore d'"attributs"

Un attribut peut être de nature qualitative ou quantitative en fonction de l'ensemble des valeurs qu'il peut prendre.

Attribut qualitatif

Un attribut est qualitatif si on ne peut pas en faire une moyenne ; sa valeur est d'un type défini en extension (une couleur, une marque de voiture, ...).

Attribut quantitatif

L'attribut est de nature quantitative : un entier, un réel, ... ; il peut représenter un salaire, une surface, un nombre d'habitants, ...

On peut donc appliquer les opérateurs arithmétiques habituels sur les attributs quantitatifs, ce qui n'est pas le cas des attributs qualitatifs.

Un attribut peut également être un enregistrement (une date par exemple), donc compose lui-même de sous-attributs (jour, mois, année dans le cas d'une date), sur lequel on peut définir les opérateurs arithmétiques habituels : donc quantitatif ne signifie pas forcement "numérique" et, réciproquement, numérique ne signifie pas forcément quantitatif : un code postal est numérique, mais pas quantitatif.

- Lorsque l'attribut ciblé est qualitatif on a un problème de ***classification***.
- On a un problème de ***régression*** lorsque l'attribut est quantitatif.

2. Arbre de décision

Plusieurs algorithmes ont être proposés, notamment CART dans les années 1980 par Breiman et al. [1984]. On décrit ici l'algorithme ID3.**[9]**
Quinlan propose en 1986 qui a être raffiné par la suite (C4.5 puis C5) (Quinlan [1993]). On constate expérimentalement que ces algorithmes sont très performants : ils construisent rapidement des arbres de décision qui prédisent avec une assez grande fiabilité la classe de nouvelles données.
ID3 ne prend en compte que des attributs nominaux. Son successeur, C4.5,

prend également en charge des attributs quantitatifs.

Dans la suite, on suppose que tous les attributs sont quantitatifs et la classe est binaire.

Précisons tout de suite que la construction d'un arbre de décision optimal est un problème NP-complet optimal au sens où il minimise le nombre d'erreurs de classification. Aussi, il ne faut pas avoir l'espoir de construire l'arbre de décision optimal pour un jeu d'exemples donnée. On va se contenter d'en construire un qui soit correct. **[28]**

2.1 Principe de construction d'un arbre de décision par l'algorithme ID3 [27]

Les tests placés dans un nœud par l'algorithme ID3 concernent exclusivement le test de la valeur d'un et seul attribut. ID3 fonctionne récursivement : il détermine un attribut a placé en racine de l'arbre. Cette racine possède autant de branches que cet attribut prend de valeurs.
A chaque branche est associé un ensemble d'exemples dont l'attribut prend la valeur qui étiquette cette branche ; on accroche alors au bout de cette branche l'arbre de décision construit sur ce sous-ensemble des exemples et en considérant tous les attributs excepte celui qui vient d'être mis a la racine. Par cette procédure, l'ensemble des exemples ainsi que l'ensemble des attributs diminuent petit à petit le long de la descente de l'arbre. **[28]**

Ayant l'idée de l'algorithme, il reste à résoudre une question centrale : quel attribut placer à la racine ? Une fois cette question résolue, on itérera le raisonnement pour les sous arbres.

L'intuition de la réponse à cette question est que l'on tente de réduire l'hétérogénéité à chaque nœud : les données qui atteignent un certain nœud de l'arbre de décision doivent être plus homogènes que les données atteignant un nœud ancêtre. Pour cela, on a besoin de pouvoir l'homogénéité d'un ensemble de données. En physique, on mesure l'homogénéité par l'entropie.

Pour cela, nous avons besoin de définir quelques notions. Commençons par l'entropie introduite initialement par Shannon [1948], notion héritée de la thermodynamique où l'entropie d'un système est d'autant plus grande qu'il est désordonné, hétérogène.

2.2 Entropie de Shannon : H(χ)

Définition 1 : Pour une classe prenant *n* valeurs distinctes, notons $p_{i \in [1, n]}$ la proportion d'exemples dont la valeur de cet attribut est *i* dans l'ensemble d'exemples considéré χ. L'entropie de l'ensemble d'exemples χ est :

$$H(\chi) = - p_i \sum_{i=1}^{n} Ln_2\ p_i$$

Définition 2 : Soit une population d'exemples χ. Le gain d'information de χ par rapport à un attribut a_j donné est la réduction d'entropie causé par la partition de χ selon a_j.

$$Gain_H(\chi, a_j) = H(\chi) - \sum_{v \in valeur\ (a_j)} \frac{|\chi_{a_{j=v}}|}{|\chi|} H(\chi_{a_{j=v}})$$

Où $\chi_{a_j=v} \subset \chi$ est l'ensemble des exemples dont l'attribut considéré a_j prend la valeur v, et la notation $|\chi|$ indique le cardinal de l'ensemble χ. **[16]**

Remarques

1. $0 \leq H(\chi) \leq 1$
2. $H(\chi)$ est minimum (=0), si une seule classe est représentée.
3. $H(\chi)$ est maximum (=1), si les classes sont équi-réparties.
4. Plus l'entropie est faible, plus la partition en sous ensemble est pure.

D'après la remarque (.4) ci-dessus, il faut pour chaque nœud de l'arbre choisir la variable explicative qui a le gain le plus élevé pour la population du nœud.

2.3 Indice de Gini : Gini(χ) [28]

Le coefficient de Gini est employé dans le cadre de certaines méthodes d'apprentissage supervisé, comme les arbres de décision. Par définition l'indice de Gini noté est donné par :

$$Gini(\chi) = 1 - \sum_{i=1}^{k} p_i^2 \quad (1)$$

On sait que :

$$\left(\sum_{i=1}^{k} p_i\right)^2 = 1^2$$

Avec :

$$\left(\sum_{i=1}^{k} p_i\right)^2 = \sum_{i=1}^{k} p_i^2 + 2 * \sum_{i<j} p_i\, p_j$$

Et par conséquent, on obtient :

$$1\text{-}\sum_{i=1}^{k} p_i^2 = 2 * \sum_{i<j} p_i\, p_j \qquad (2)$$

Afin de pouvoir comparer la courbe l'indice de Gini avec celle de l'entropie, on doit retrouver une valeur maximale pour l'indice de Gini (=1), et donc, on se ramène à une normalisation de l'expression (2).

On a aussi Le gain d'information de χ par rapport à un attribut a_j :

$$Gain_{Gini}\,(\chi, a_j) = Gini(\chi) - \sum_{v \in valeur\,(a_j)} \frac{|\chi_{a_{j=v}}|}{|\chi|}\; Gini(\chi_{a_{j=v}})$$

On choisit a_p qui donne un gain maximal.

Remarque

Considérons le cas de deux classes (k=2) et appelons *x* la proportion d'éléments de classe 1 en position *p*. On a donc la fonction d'entropie définie par :

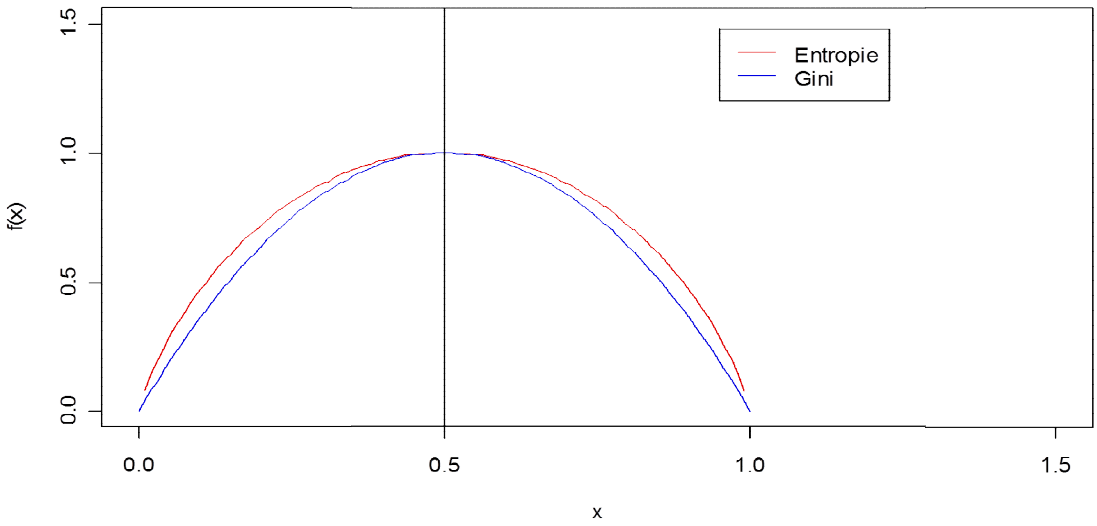
$$H(p) = -\, x \log(x) - (1\text{-}x) \log (1\text{-}x).$$

Cette fonction de *x* prend ses valeurs dans l'intervalle [0,1], a son minimum pour *x*=0 et *x*=1 qui vaut 0 et a son maximum pour *x*=1/2 qui vaut 1.

La fonction de Gini est définie par :

$$Gini(p) = 2\, x(1\text{-}x).$$

Cette fonction de *x* prend ses valeurs dans l'intervalle [0,1/2], a son minimum pour *x*=0 et *x*=1 qui vaut 0 et a son maximum pour *x*=1/2 qui vaut 1/2. Ces deux fonctions sont symétriques par rapport à *x*=1/2.



2.4 Généralités sur l'apprentissage des arbres de décision [27]

Diviser récursivement et le plus efficacement possible les exemples de l'ensemble d'apprentissage par les tests définis à l'aide des attributs jusqu'à obtenir des sous-ensembles d'exemples ne contenant (ou presque) que des exemples appartenant tous à la même classe ou majoritaire (avec une certaine proportion) dans un nœud terminal

Dans toutes les méthodes on retrouve les trois opérations suivantes :

1. **Décider si un nœud est terminal**, c'est-à-dire décider si un nœud doit être étiqueté comme une feuille. Par exemple : tous les exemples sont dans la même classe, il y a moins d'un certain nombre d'erreurs, ...

2. **Sélectionner un test à associer à un nœud** (qui n'est pas terminal).Par exemple : aléatoirement, utiliser des critères statistiques, …

3. **Affecter une classe à un nœud terminal ou feuille**. On attribue la classe majoritaire sauf dans les cas où l'on utilise des fonctions coût ou risque.

Les méthodes vont différer par les choix effectués pour ces différents opérateurs, c'est-à-dire sur le choix d'un test (par exemple, utilisation du gain et de la fonction entropie) et le critère d'arrêt (quand arrêter la croissance de l'arbre, soit quand décider si un nœud est terminal).

3. Différents algorithmes d'apprentissage [20]

Algorithme ID3 (Quinlan 1979)

- Arbres de discrimination c'est-à-dire des variables uniquement **qualitatives**.
- Critère d'hétérogénéité utilisée est **l'indice d'entropie**.

Algorithme C4.5 (Quinlan 1993) Qui n'est rien d'autre qu'une amélioration d'ID3, permettant notamment arbres de « régression » (gestion des **variables continues**) et valeurs manquantes.

Algorithme CART (Breiman et al.1984) Arbres de « régression et classification ». Le critère d'hétérogénéité utilisé est **l'indice de Gini**.

Nous allons détailler un algorithme en particulier ID3.

3.1 Algorithme ID3 (Quinlan 1979)

ID3 est un algorithme de classification supervisé, c'est-à-dire qu'il se base sur des exemples déjà classés dans un ensemble de classes pour déterminer un modèle de classification. Le modèle que produit ID3 est un arbre de décision. Cet arbre servira à classer de nouveaux échantillons.

Algorithme ID3:

Nous adopterons les notations suivantes :

R : ensemble de variables, X : la variable de classement, S : la population

Début

{

1. **Si** S est vide **retourner** une feuille vide
2. **Sinon si** S est un ensemble d'individus prenant la même valeur pour X.
3. **retourner** une feuille avec cette valeur
4. **Sinon si** R est vide
5. **retourner** une feuille contenant le mode le plus courant de X

pour les individus de S

6. **Sinon**
7. **Soit** D la variable avec le plus grand gain d'information ;

8. **Soient** d1, d2, …, dm les modalités de D ;
9. **Soient** S1, S2, …, Sm les sous ensembles des individus prenant la valeur dj pour D ;
10. **retourner** un arbre ayant D pour racine, des arêtes partant de D et étiquetées d1, d2, …, dm ayant respectivement comme sous arbre ID3 (R\D, X, S1), …, ID3 (R\D, X, Sm)

}

Fin

Remarque:

Un autre test d'arrêt peut être lorsque le nombre d'individus d'un nœud est inferieur à un seuil fixé, le nœud devient alors une feuille.

3.2 Algorithme C4.5 (Quinlan 1993)

Algorithme de base

On suppose toujours que le langage de représentation est constitué d'un certain nombre d'attributs. Ces attributs peuvent être binaires, qualitatifs (à valeurs dans un ensemble fini de modalités) ou continus (à valeurs réelles).

Pour les attributs continus, on utilise des heuristiques qui permettent de les discrétiser. On utilise pour cela des critères statistiques qui permettent d'atteindre les deux objectifs suivants :

- Un nombre de classes pas trop important et une bonne répartition entre les différentes classes.
- On peut par exemple utiliser la fonction entropie pour atteindre ces objectifs.

Nous supposons maintenant que les attributs ont été discrétisés.

Nous supposons prédéfini un ensemble de tests *n*-aires. Pour définir l'algorithme, nous allons définir les trois opérateurs utilisés par l'algorithme C4.5 pour calculer un bon arbre de décision (phase d'expansion), puis nous verrons la phase d'élagage. On dispose en entrée d'un ensemble d'apprentissage *A*.

• **Décider si un nœud est terminal**. Un nœud *p* est terminal si tous les

éléments associés à ce nœud sont dans une même classe ou si aucun test n'a pu être sélectionné (voir ci-après).

• **Sélectionner un test à associer à un nœud**. À chaque étape, dans l'ensemble des tests disponibles, ne peuvent être envisagés que les tests pour lesquels il existe au moins deux branches ayant au moins deux éléments (cette valeur par défaut peut être modifiée). Si aucun test ne satisfait cette condition alors le nœud est terminal.

• **Affecter une classe à une feuille**. On attribue la classe majoritaire. S'il n'y a aucun exemple on attribue la classe majoritaire du nœud père.

Rapport de gain

En présence d'attribut numérique ou d'attribut d'arité élevée, ceux-ci sont automatiquement favorisés pour être sélectionnés comme test dans les nœuds. Pour contrecarrer cet effet, C4.5 utilise le rapport de gain au lieu du gain d'information pour déterminer l'attribut à utiliser dans un nœud.

Le rapport de gain est défini par :

$$\text{Rapport de gain}(\chi, a) = \frac{Gain(\chi,a)}{SplitInfo(\chi,a)}$$

Où :

$$SplitInfo(\chi, a) = \sum_{v \in valeurs\,(a)} \frac{|\chi_{a=v}|}{|\chi|} \log_2 \frac{|\chi_{a=v}|}{|\chi|}$$

Contrairement à ID3 qui choisit l'attribut ayant le gain maximal, C4.5 choisit l'attribut ayant le rapport de gain maximal pour le placer à la racine de l'arbre de décision.

- **Valeurs d'attributs manquantes**

Le problème des exemples dont la valeur de certains attributs est manquante, est un problème concret extrêmement fréquent. ID3 ne dispose d'aucune possibilité pour gérer ce manque d'information, par contre, l'algorithme C4.5 dispose d'un mécanisme pour résoudre ce type de problème.

Dans ce contexte, on distingue deux cas :

- Certain attributs des exemples de l'ensemble d'apprentissage sont non valués ;
- Certain attributs de la donnée à classer sont non valués.

- **Attributs non valués dans l'ensemble d'apprentissage**

Plusieurs solutions peuvent être envisagées, les plus générales étant :

On laisse de coté les exemples ayant des valeurs manquantes ;

Le faite que la valeur de l'attribut soit manquante est une information en soit : on ajoute alors une valeur possible à l'ensemble des valeurs de cet attribut qui indique que la valeur est inconnue ;

La valeur la plus courante pour l'attribut en question parmi les exemples classées dans ce nœud est affectée à la place de la valeur manquante ;

Les différentes valeurs observées de cet attribut parmi les exemples couvert par le même nœud sont affectées avec des poids différents en fonction de la proportion d'exemples de l'ensemble d'apprentissage couvert par ce nœud par les différentes valeurs de cet attribut.

C'est cette dernière possibilité qui est utilisée par C4.5. Dés lors, des fractions d'exemples sont utilisées pour continuer la construction de l'arbre. Il faut alors adopter le calcul du gain d'information.

Pour calculer le gain d'information, on ne tient compte que des exemples dont l'attribut est valué.

Définition3 : Soit χ l'ensemble d'exemples couverts par le nœud courant (dont on est en train de déterminer l'attribut à tester) et $\chi_{\text{sans ?}} \subset \chi$ les exemples dont l'attribut est valué. On redéfinit :

$$H(\chi) = H(\chi_{\text{sans ?}})$$

Et

$$Gain_H(\chi, a) = (H(\chi) - \sum_{v \in valeurs(a)} \frac{|\chi_{sans?,a=v}|}{|\chi_{sans?}|} H(\chi_{\text{sans?}})) \frac{|\chi_{sans?}|}{|\chi|}$$

- **Classification d'une donnée ayant des attributs non valués**

On se place ici non pas dans la phase de construction de l'arbre de décision, mais lors de son utilisation pour prédire la classe d'une donnée.

Lors de sa descente dans l'arbre, si un nœud test un attribut dont la valeur est inconnue, C4.5 estime la probabilité pour la donnée de suivre chacune des branches en fonction de la répartition des exemples du jeu d'apprentissage

couverts par ce nœud. Cela détermine une fraction de donnée qui poursuit sa descente selon chacune des branches.

Arrivé aux feuilles, C4.5 détermine la classe la plus probable à partir de ces probabilités estimées. Pour chaque classe, il fait la somme des poids ; la classe prédite est celle dont le poids est maximal.

- **ID3 vs. C4.5**

Dans cette section, on résume Les différences entre ID3 et C4.5 :

ID3 construit un arbre de décision :

- les attributs doivent tous être qualitatifs et ils sont considérés comme nominaux ;
- ne peut pas prendre en charge des exemples ou des données dans lesquels il manque la valeur d'attribut ;
- utilise les notions d'entropie et de gain pour déterminer l'attribut à tester dans chaque nœud.

En ce qui concerne C4.5 :

- les attributs peuvent être qualitatifs ou quantitatifs ;
- peut utiliser des exemples dans lesquels la valeur de certains attributs est inconnue lors de la construction de l'arbre décision;
- peut prédire la classe de donnée dont la valeur de certains attributs est inconnue ;
- utilise le rapport de gain pour déterminer l'attribut à tester dans chaque nœud.

3.3 Algorithme CART rpart

L'algorithme CART est le même qu'ID3 à une modification près. Ci-dessus sont mises les étapes modifiées :

7. **Soit** D la variable avec le plus grand gain d'information (ou la plus petite variance résiduelle) ;

8. **Soient** $d_1, d_2, \dots, d_m$ les modalités de D et d_i celle qui partage en deux la

population telle que D est le plus grand gain d'information (ou la plus petite variance résiduelle) ;

9. **Soient** S_{gi} (respectivement S_{di}), le sous ensemble des individus tel que la valeur de D soit inferieure ou égale (respectivement supérieure) a d_i ;

10. **Retourner** un arbre ayant D pour racine, deux aretes partant de D et étiquetées $\leq d_i$, $> d_i$ ayant respectivement comme sous arbre ID3(R, X, S_{gi}), ID3(R,X, S_{di})

Remarque

Lorsque les variables sont toutes quantitatives, on peut aussi ajouter comme test d'arrêt que l'écart-type d'une feuille doit être inférieure ou égale à *x*% (*x* ≈ 5) de l'écart- type total.

4 . Validation d'un arbre de décision[16]

4.1 Validation croisée

Une méthode plus sophistiquée est la « validation croisée ». Pour cela, on découpe l'ensemble des exemples en *n* sous-ensembles mutuellement disjoints.
Il faut prendre garde à ce que chaque classe apparaisse avec la même fréquence dans les *n* sous-ensembles. Si n = 3, cela produit donc 3 ensembles A, B et C. On construit l'arbre de décision $AD_{A\cup B}$ avec A ∪B et on mesure son taux d'erreur sur C, c'est à dire, le nombre d'exemples de C dont la classe est mal prédite par $AD_{A\cup B}$: E_C.
Ensuite, on construit l'arbre de décision $AD_{B\cup C}$ avec B∪C et on mesure l'erreur sur A :E_A. Enfin, on construit l'arbre de décision $AD_{A\cup C}$ avec A∪C en mesurant l'erreur sur B :$E_{B.}$
Le taux d'erreur E est alors estimé par la moyenne de ces trois erreurs mesurées :

$$E = \frac{E_A + E_{B.} + E_C}{3}$$

Habituellement, on prend *n* = 10.
Cette méthode est dénommée « validation croisée en *n*-plis » (*n-fold cross-validation*).

4.2 Technique du *leave-one-out*

Cette technique consiste à effectuer une validation croisée à *n* = N plis en laissant à chaque fois un seul exemple de coté (N est le nombre d'exemples dont on dispose).

L'erreur est estimée par la moyenne des N erreurs mesurées.

4.3 Technique de bootstrap

Le jeu d'apprentissage est constitué en effectuant N tirages avec remise parmi l'ensemble des exemples. Cela entraine que certains exemples du jeu d'apprentissage seront vraisemblablement sélectionnés plusieurs fois, et donc que d'autres ne le seront jamais. En fait, la probabilité qu'un certain exemple ne soit jamais tiré est simplement :

$$(1 - \frac{1}{N})^N$$

La limite quand $N \to +\infty$ de cette probabilité est $e^{-1} \approx 0,368$.

Les exemples qui n'ont pas été sélectionnés constituent le jeu de test.

Le jeu d'apprentissage contient 63,2 % des exemples du jeu d'exemples initial(en moyenne).

L'erreur est calculée en combinant l'erreur d'apprentissage E_{app} et l'erreur de test E_{test} par la formule suivante :

$$E = 0,632 \times E_{test} + 0,368 \times E_{app}$$

Ensuite, cette procédure est itérée plusieurs fois et une erreur moyenne est calculée.

5. Erreur de prévision [9]

Un arbre de décision est construit à partir d'un échantillon de la base de données appelé *échantillon d'apprentissage*. Il est ensuite testé sur un autre échantillon, appelé *échantillon test.*

En général, l'échantillon d'apprentissage représente $\frac{2}{3}$ de la population et l'échantillon test le reste, soit $\frac{1}{3}$

Lorsque la population est trop petite pour permettre de construire l'arbre sur $\frac{2}{3}$ de ses représentants, on procède par validation croisée, c'est-à-dire que l'on répète le schéma *apprentissage/test* sur différentes fractions constituées à partir des données initiales.

6. Elagage

Lorsque l'erreur de prévision apparente est bien plus faible que l'erreur de prévision théorique, on dit qu'il y a en *apprentissage par cœur* (overfitting). Il faut donc élaguer l'arbre, c'est-à-dire réduire certaines branches, afin de diminuer l'erreur de prévision théorique (ce qui aura bien sûr pour effet d'augmenter l'erreur de prévision apparente).

L'idée est d'élaguer les branches avec beaucoup de feuilles (diminution de la complexité de l'arbre) et qui font peut varier le taux d'erreur.

Pour cela, on construit une suite d'arbres emboités, obtenus à partir de l'échantillon d'apprentissage, par élagages successifs. Puis, on sélectionne parmi cette suite d'arbres, l'arbre ayant l'erreur de prévision théorique la plus faible.

Elaguer un nœud signifie transformer ce nœud en feuilles, c'est-à-dire éliminer tout le sous arbre maximal enraciné en ce nœud.

Chapitre 3

Les Réseaux De Neurones

Les modèles neuronaux présentent plusieurs avantages par rapport aux modèles paramétriques traditionnels.

Ils permettent de ne poser aucune hypothèse, a priori, sur la forme de la relation qui unit les variables d'entrée et les variables de sortie, mais, de la déduire à partir des données.

Ils ont l'avantage de ne pas se fonder sur des hypothèses spécifiques concernant les dynamiques des variables d'état (prix de l'actif sous-jacent et la volatilité) et, donc, ils sont robustes aux erreurs de spécification qui affectent, au contraire, les modèles paramétriques. **[24]**

Par contre, le défi auquel les modélisateurs doivent faire face est qu'il n'y a pas de théorie qui permet de déterminer le modèle neuronal optimal.

Etant donné la flexibilité et le pouvoir des réseaux de neurones d'approximer des fonctions complexes et non linéaires, il est normal de l'appliquer à l'évaluation des produits dérivés dont les formules d'évaluation sont fortement non linéaires. Parmi les réseaux de neurones qui sont dotés de cette propriété, il y'a le perceptron multicouche (MLP) qui se base, en général, sur des fonctions d'activation de type sigmoïde et les fonctions radiales de base (RBF) qui se basent sur des fonctions d'activation de type gaussienne dont les centres sont à déterminer au cours de l'apprentissage. L'approche la plus utilisée, actuellement en finance est le perceptron multicouche. Associée à l'algorithme de rétropropagation du gradient, elle peut être présentée comme une forme d'extension des modèles de régression dans un contexte non linéaire. Cette approche a été utilisée dans plusieurs travaux qui ont traité le sujet de l'évaluation des options par les réseaux de neurones. On peut citer, à cet effet, Malliaris (1993), Hutchinson (1994), Fiordaliso (1997), Anders & Korn (1998), Garcia & Gençay (2000) et Amilon (2001)**[32]**

1. Théorie Des Réseaux De Neurones

1.1 Présentation de l'approche [10]

Un réseau de neurones artificiel est un processus composé d'unités de traitement simples, parallèlement, distribuées, destinées à accumuler la connaissance expérimentale et la rendre opérationnelle (Aleksander et Morton (1990)).
Il ressemble au cerveau, par les deux caractéristiques suivantes:

1. Le réseau acquière la connaissance de son environnement, à travers un processus d'apprentissage.
2. L'intensité d'une connexion, entre les neurones, connue sous le nom de poids synaptique, est utilisée pour accumuler la connaissance acquise

Le problème majeur, pour un réseau de neurones, consiste à identifier un modèle capable de reproduire, le plus fidèlement possible, la réalité imposée par le modélisateur, le plus souvent exprimée par des objectifs clairs et précis et définis dans un environnement spécifique.

Cependant, la connaissance de l'environnement nécessite :

- La connaissance des états de la nature basée sur les informations passées et présentes.
- L'observation est obtenue au moyen d'une analyse approfondie de l'environnement.
- Les observations obtenues fournissent un ensemble d'informations, à partir duquel, les nœuds d'inputs du réseau de neurones sont alimentés.

En fait, la connaissance, fournie au réseau de neurones, va être stockée dans l'échantillon d'apprentissage. Ce dernier est constitué, quant-à-lui, d'un ensemble de combinaisons d'inputs et d'outputs. Chaque combinaison est constituée de signaux d'inputs, auxquels correspondent des réponses désirées.

1.2 Neurone formel

Un neurone est considéré comme un dispositif qui reçoit, à partir d'autres neurones ou de l'extérieur, des stimulations par des entrées (inputs), au Φnombre de n, et les pondère grâce à des valeurs réelles appelées coefficients synaptiques ou poids synaptiques. Ces coefficients peuvent être positifs, et l'on parle alors de synapses excitatrices, ou négatifs pour des synapses inhibitrices. Un neurone j calcule ainsi, un potentiel P_j égal à la somme des ses entrées (x_1, x_2,…, x_n) pondérées, par les coefficients synaptiques respectifs (w_1, w_2, …, w_n) à laquelle on ajoute un terme constant : le biais b_j La valeur du potentiel P_j est donnée par l'équation suivante :**[32]**

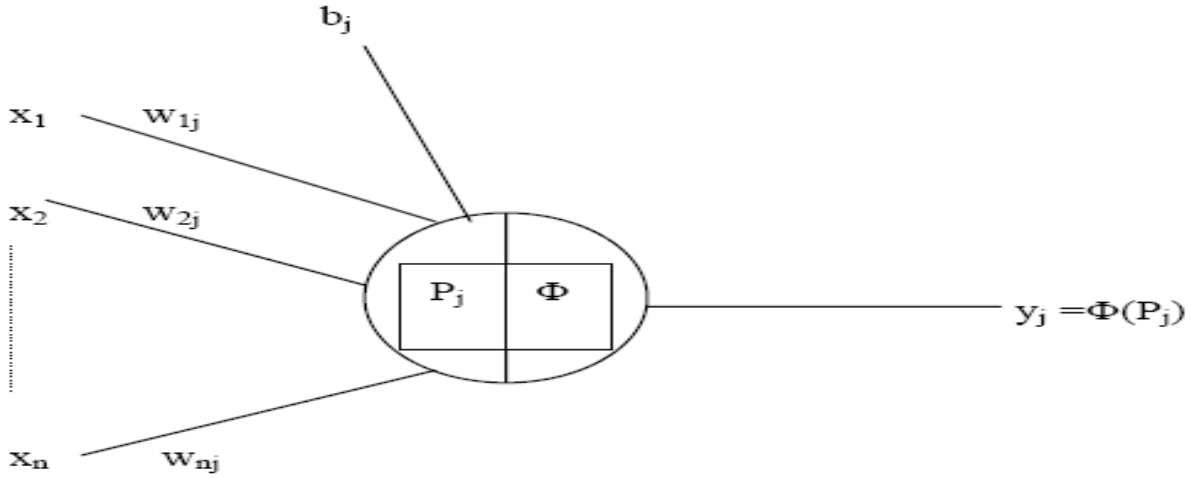
$$Pj= \sum_{i=1}^{n} wij\ xi + bj$$

A ce potentiel, le neurone applique une fonction d'activation Φ, de manière à ce que la sortie y_j , calculée par le neurone soit égale à Φ(p_j) tel que :

$$Yj= \Phi(Pj)=\phi(\sum_{i=1}^{n} wij\ xi + bj)$$

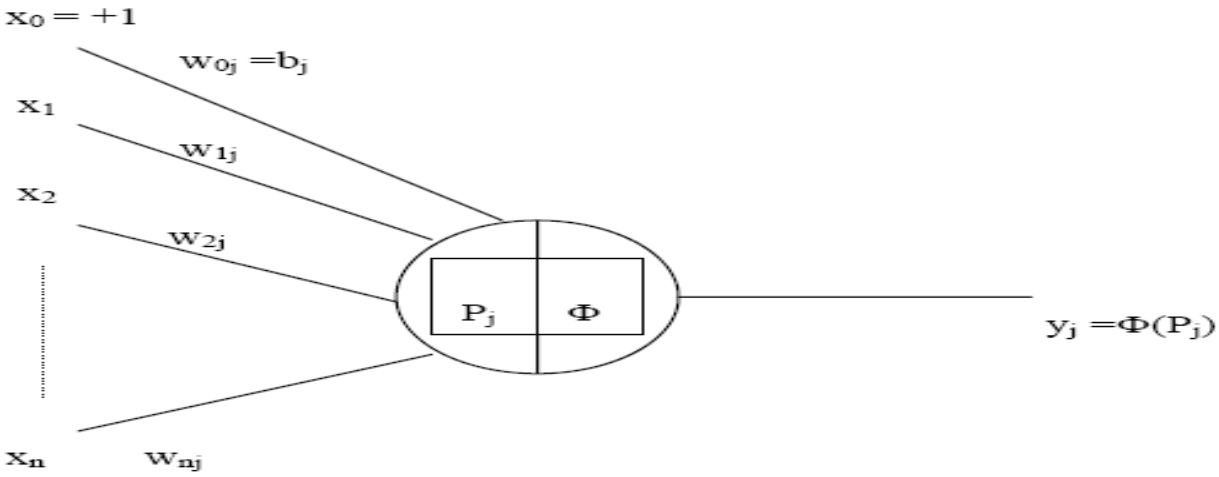
La valeur de sortie y_j (output) est émise par le neurone vers d'autres neurones ou vers l'extérieur. Ainsi, un neurone est caractérisé par trois concepts : son état interne qui est son potentiel, ses connexions avec d'autres neurones et sa fonction de transfert.

Le schéma d'un neurone formel peut être présenté comme suit:



Neurone formel : inputs et bias bj : x_{0j}=bj et w_{0j}=+1

L'utilisation du biais est de nature à appliquer une transformation affine au potentiel. En fait le biais est un paramètre externe du neurone j, il peut être intégré dans l'équation du potentiel, comme étant le signal x_0 qui prend la valeur 1, pondéré par le poids w_{0j} dont la valeur est égale au biais b_j La présentation du modèle neuronal, où le biais est considéré comme un nœud d'input, est la suivante :



Neurone formel : input et bias poids synaptique, input bias=+1

Le modèle devient , ainsi :

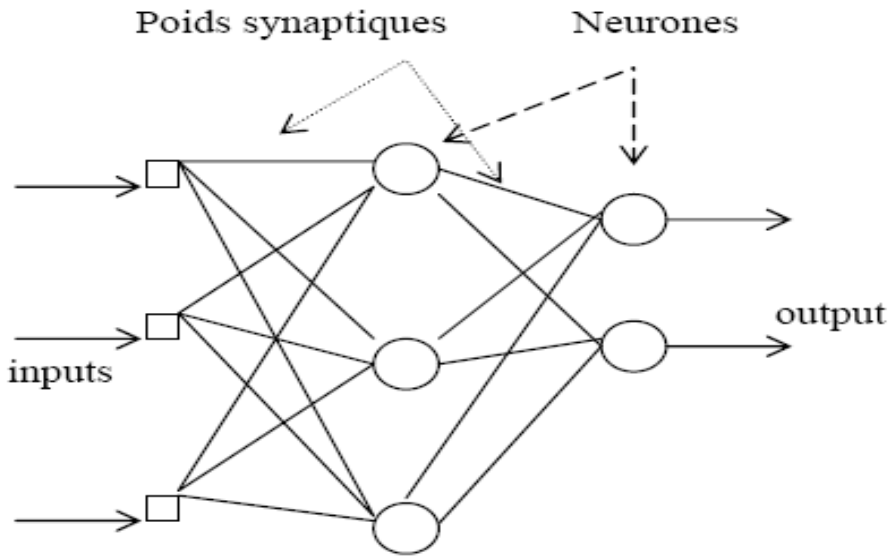
$$Pj= \sum_{i=1}^{n} wij\ xi$$

Et

$$Yj= \Phi(Pj)=\phi(\sum_{i=1}^{n} wij\ xi)$$

1.3Définition d'un réseau de neurones [22]

Un réseau de neurones est constitué par un certain nombre de neurones interconnectés. Chaque neurone peut recevoir plusieurs signaux intrant des autres neurones, les agrège en se basant sur une fonction d'entrée et génère, enfin, un signal sortant, basé sur une fonction de sortie, appelée fonction d'activation. Le signal sortant alimentera, ensuite, d'autres neurones, suivant la typologie du réseau. La valeur numérique d'un poids synaptique, relatif à une connexion entre deux nœuds, reflète l'ampleur de l'influence d'un neurone sur l'autre neurone et le sens de cette influence. La résolution d'un problème neuronal revient à déterminer les poids relatifs aux connexions. Dans sa forme typique, un réseau de neurones est composé de trois types de couches : une couche d'input, une couche d'output et des couches cachées. Les neurones de la couche d'input reçoivent les signaux de l'environnement, les neurones de la couche d'output émettent des signaux à l'environnement.
Néanmoins, les neurones des couches cachées n'ont aucune interaction directe avec l'environnement, d'où leur appellation. Les connexions entre les couches se font dans une seule direction, de la couche d'input vers la couche d'output. Chaque couche est formée au moins d'un neurone



Réseau de neurones : architecture "feedforward" MLP à une couche cachée

En effet, choisir un réseau de neurone, revient à sélectionner une architecture de réseau et, par conséquent, sélectionner la manière, dont les neurones sont structurés et connectés, ainsi que, le type d'algorithme d'apprentissage à utiliser. Trois éléments sont, donc, particulièrement importants quelque soit le réseau de neurones :

- la structure des neurones

- la topologie du réseau

-l'algorithme d'apprentissage qui permet de trouver les poids synaptiques du réseau

Les réseaux multicouches « feedforward » "multilayer feedforward networks" représentent une classe importante des réseaux de neurones. Pour un tel type de réseau, au sein d'une couche, chaque neurone agit indépendamment des autres, et en particulier, il ne reçoit aucune connexion en provenance des neurones de cette couche.

1.4 Propriété fondamentale des réseaux de neurones: l'approximation universelle parcimonieuse [35] (Théorème de Kolmogorov)

« Toute fonction, bornée, suffisamment régulière, f 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 ». Si on considère un nombre d'input égal à n

et un nombre d'output égal à p on aura à approcher une fonction de IR^n dans IR^p

Le théorème de l'approximation universelle de fonction s'énonce comme suit :

- soit φune fonction bornée, continue et strictement croissante,

- soit $I_n= [0,1]^n$ un intervalle de IR^n

- soit C(In) l'espece de fonctions continues sur In et soit ξ un réel positif alors, il existe un entier p et des suites de parametre αi bi et ωij (avec 1≤i≤p et 1≤j≤n) telque :

$$(x_1,x_2,\dots,x_n)=\sum_{i=1}^{p} \alpha i\ \varphi\left(\sum_{i=1}^{n} wij\ xi + bi\right)$$

soit une approximation de la fonction φ tel que :

$$\forall\ (x_1,x_2,\dots,x_n)\ ;|\ F\ (x_1,x_2,\dots,x_n) - f(x_1,x_2,\dots,x_n)|<\varepsilon$$

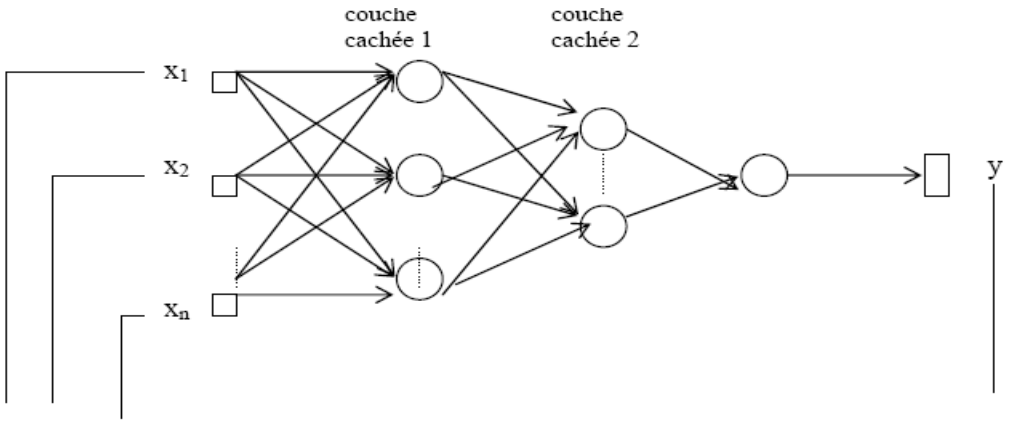
Cette propriété n'est pas spécifique aux réseaux de neurones, ils existent bien d'autres familles de fonctions paramétrées possédant cette propriété, c'est le cas, notamment, des polynômes, des séries de Fourier, etc …. La spécificité des réseaux de neurones /réside dans le caractère parcimonieux de l'approximation : A précision égale, les réseaux de neurones nécessitent moins de paramètres ajustables (poids synaptiques) que les approximateurs universels utilisés. Plus précisément, le nombre de poids synaptiques varie, linéairement, avec le nombre de variables de la fonction à approcher, alors, qu'il varie, exponentiellement, pour la plupart des autres approximateurs. En pratique, dès qu'un problème fait intervenir plus de deux variables, les réseaux de neurones sont, en général, préférables aux autres méthodes.

1.5 Perceptron multicouches

Le "MultiLayer Perceptron" est l'architecture la plus connue des réseaux de neurones du type « feedforward ». Il représente le modèle le plus courant et le plus simple de réseau non linéaire.
Pour doter le perceptron multicouches de la propriété de non-linéarité, il faut qu'il comporte, au mois, une couche cachée, et que les fonctions d'activation des neurones, qui le composent, soient non-linéaires.
Plus on introduit de neurones avec une fonction d'activation non linéaire, plus on dote le réseau d'une capacité pour résoudre des problèmes complexes et plus le découpage de l'espace des inputs obtenu se rapproche de celui des

données.

Dans la pratique, il est rare d'utiliser plus de deux couches cachées. L'architecture suivante est relative à un réseau multicouches « feedforward », avec deux couches cachées et un seul output**.[39]**



$(x_1, x_2, \ldots, x_n)_1$	1	y_1
$(x_1, x_2, \ldots, x_n)_2$	2	y_2
$(x_1, x_2, \ldots, x_n)_3$	3	y_3
⋮ ⋮ ⋮	⋮	⋮
$(x_1, x_2, \ldots, x_n)_N$	N	y_N
Vecteur input	Numéro de l'observation	Output désiré

Réseau de neurones MLP : les poids synaptiques sont estimés à travers un apprentissage sur une base de donnée comportant les arguments de la fonction $X=(x_1, x_2, \ldots, x_n)_i$ $(1 \le i \le N)$ et sa valeur $y=(y_1, y_2, \ldots, y_N)$.

Les connexions, entre les couches, se font dans une seule direction, de la couche d'input vers la couche d'output. Un neurone d'une couche donnée reçoit, de tous les neurones de la couche précédente, des signaux intrants pondérés par des poids synaptiques. Après agrégation de ces signaux, suivant une fonction d'entrée, il génère un signal sortant basé sur une fonction de sortie appelée fonction d'activation, lequel signal est envoyé avec pondération, avec des coefficients synaptiques, à tous les neurones de la couche suivante. Le signal d'input se propage à travers le réseau dans une direction « forward », de couche en couche. Le nombre de neurones constituant les couches d'input et d'output dépend du problème à traiter, alors que, le nombre de couches cachées, ainsi que, le nombre de neurones, sur chacune d'elles, sont fixés par le modélisateur. Un réseau sans couches cachées ne peut résoudre que des problèmes linéaires. La propriété de non-linéarité est acquise par l'introduction de la première couche cachée.

2. *Algorithme de rétropropagation du gradient*

2.1 L'algorithme [45]

L'apprentissage par l'algorithme de rètropropagation des gradients est une application d'une méthode statistique connue sous le nom « approximation stochastique » qui a été proposée par Robbins & Monro (1951). On considère un réseau de neurones, du type MLP à Q couches. Soit Nq le nombre de neurones sur la couche numéro q (avec $1 \leq q \leq Q$). A l'itération n, qui correspond à la n ^{eme} observation du fichier « training », si le neurone j est un neurone appartenant à la couche de sortie, le signal d'erreur à la sortie de ce neurone est :

$$e_j(n)=d_j(n) - y_j(n)$$

Les neurones de la couche de sortie sont les seuls pour lesquels le signal de sortie peut être calculé directement. L'erreur quadratique, relative à l'ensemble des neurones de sortie, est la somme des fonctions énergie des erreurs correspondantes à ces neurones, soit :

$$\xi(n) = \frac{1}{2}\sum_{i=1}^{N_Q} e_j^2 \ (n)$$

Pour un fichier de training comportant N observations. L'erreur quadratique moyenne, sur l'ensemble de ces N observations, et relative à l'itération n, est :

$$\xi_m = \frac{1}{N} \sum_{n=1}^{N} \xi\ (n)$$

L'objectif de l'apprentissage est d'ajuster les poids synaptiques du réseau, afin de, minimiser cette erreur ξ_m On considère un processus d'apprentissage pour lequel les poids synaptiques sont mis à jour pour chaque observation du fichier training. Une itération correspond au parcours de l'ensemble des observations du jeu d'apprentissage par l'algorithme d'apprentissage. Si on considère un neurone j, qui reçoit des signaux de tous les neurones de la couche précédente, le potentiel à l'entrée de ce neurone est, donc :

$$P_j(n) = \sum_{i=0}^{m} wij(n)\ yi(n)$$

Le signal de sortie correspondant est :

$$Y_j(n)=\varphi_j(p_j(n))$$

L'algorithme de rètropropagation du gradient consiste à corriger les poids synaptiques ω_{ji} selon la règle Delta, tel que ce poids devient :

$$\omega_{ji}(n+1)=\omega_{ji}(n)+\Delta\omega_{ji}(n)$$

où, si on désigne par η est le coefficient d'apprentissage, alors :

$$\Delta\omega_{ji}(n)=-\eta\frac{\partial\xi(n)}{\partial\omega ji(n)} \quad (1)$$

La dérivée partielle de la fonction d'erreur, par rapport au poids synaptique ω_{ij} (n) peut être décomposée comme suit :

$$\frac{\partial\xi(n)}{\partial\omega\, ji(n)}=\frac{\partial\xi(n)}{\partial ej(n)}\frac{\partial ej(n)}{\partial yj(n)}\frac{\partial yj(n)}{\partial pj(n)}\frac{\partial pj(n)}{\partial\omega ji(n)} \quad (2)$$

D'après la définition, même, de chacune de ces variables, on a :

$$\xi(n)=\frac{1}{2}\sum_{i=1}^{N_Q}e_j^2(n)\Rightarrow\frac{\partial\xi(n)}{\partial ej(n)}=e_j(n)$$

$$e_j(n)=d_j(n)-y_j(n)\Rightarrow\frac{\partial ej(n)}{\partial yj(n)}=-1$$

$$Y_j(n)=\varphi_j(p_j(n))\Rightarrow\frac{\partial yj(n)}{\partial pj(n)}=\varphi_j(p_j(n))$$

$$P_j(n)=\sum_{i=0}^{m}wij(n)\,yi(n)\Rightarrow\frac{\partial pj(n)}{\partial wji(n)}=y_j(n)$$

Soit en remplaçant, dans l'équation (2) on obtient :

$$\frac{\partial\xi(n)}{\partial\omega ji(n)}=-e_j(n)\,\varphi_j(p_j(n))\,y_i(n)$$

En remplaçant cette expression, dans l'équation (1) on a

$$\Delta\omega_{ji}=\eta\,e_j(n)\,\varphi_j(p_j(n))\,y_i(n)=\eta\,\Psi_j(n)\,y_i(n)$$

Avec

$$\Psi_j(n)=-\frac{\partial\xi(n)}{\partial pj(n)}=e_j(n)\,\varphi_j(p_j(n))$$

Ψ_j(n) est le gradient local de l'erreur relatif à la sortie du neurone j, le signale d'erreur à la sortie du neurone j, est une variable clé pour l'ajustement des poids synaptiques.

2.2 La fonction d'activation [22]

Le gradient local de l'erreur, pour chaque neurone, dépend de la dérivée de la

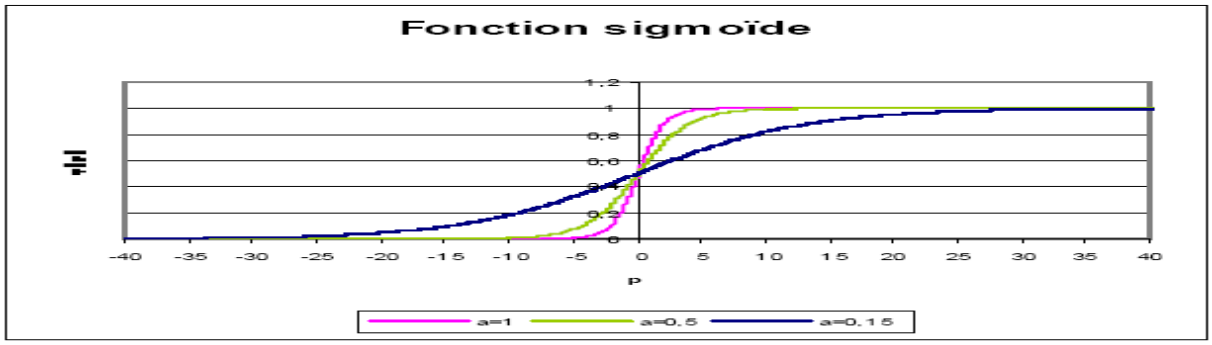
fonction d'activation. Ainsi, cette fonction est supposée dérivable, donc, continue. Les fonctions d'activation non linéaires et différentiables, couramment utilisées dans les perceptrons multicouches, sont la fonction sigmoïde et la fonction tangente hyperbolique.

2.2.1 La fonction sigmoïde

La forme générale de la fonction sigmoïde, appliquée à un neurone j, est définie par :

$$\varphi\,(p_j(n))=\frac{1}{1+e^{-apj(n)}}$$

Ou a est un réel strictement positive et $p_j(n)$ potentiel à l'entrée du neurone j, à l'itération n. cette fonction est dans l'intervalle [0.1]. soit $yj(n)=\varphi(pj(n))\in[0.1]$ la courbe de la fonction sigmoïde est représenté comme suit :



Graphique de la fonction d'activation sigmoïde pour a=0,15 ; a=0,5 et a=1

Sa fonction dérivée est :

$$\frac{\partial yj\,(n)}{\partial pj(n)}=\varphi'(pj(n))=\frac{a\,e^{-apj(n)}}{(1+e^{-apj(n)})^2}=a\,y_j\,(n)\,(1\text{-}y_j(n))$$

Il s'en suit que la variation des poids synaptiques, reliant un neurone j, à des neurones k, est d'autant plus importante, que le gradient local du neurone j est grand et, donc, que la valeur de la fonction signal à la sortie de ce neurone est d'autant plus proche de 0.5 et, donc, le potentiel à son entrée proche de 0. Elle est, d'autant plus faible, que le potentiel est grand en valeur absolue. D'après Rumelhart et al. (1986), c'est ce qui fait la stabilité de l'algorithme de rètropropagation du gradient.

2.2.2 La fonction Fonction tangente hyperbolique

La forme générale de la fonction tangente hyperbolique est définie par :

$$\varphi\big(pj(n)\big)=a\tan h\big(b.pj(n)\big)\;avec\;a>0\;et\;b>0$$

Cette fonction est, à valeur dans l'intervalle [-1,1], tel que :

$$y_j(n) = \varphi\big(pj(n)\big) \in [-1,1]$$

Sa fonction dérivée est :

$$\frac{\partial yj\ (n)}{\partial pj(n)} = \varphi'\big(pj(n)\big) = ab\ [1 - \Big(\tanh\big(bpj(n)\big)^2\Big] = \frac{b}{a}\ [a - yj(n)][a + yj(n)]$$

2.3 Le coefficient d'apprentissage

La correction des poids synaptiques est proportionnelle au coefficient d'apprentissage *n*. Plus ce coefficient est petit, plus les variations des poids synaptiques du réseau sont faibles, d'une itération à une autre, et vice versa. Un faible coefficient d'apprentissage rend la convergence de l'algorithme plus lente mais permet une précision meilleure et surtout permet d'éviter les minimas locaux qui faussent la solution du problème. Un coefficient d'apprentissage grand, permet d'accélérer le rythme de l'apprentissage avec des variations importantes des poids synaptiques, d'une itération à l'autre, en contrepartie le réseau peut devenir instable (oscillations des valeurs des poids synaptiques). En plus, le risque de tomber dans un minimum local est d'autant plus grand que le coefficient d'apprentissage est grand. Il y'a donc un arbitrage à faire pour converger, rapidement, vers une solution tout en améliorant la précision, éliminer le risque d'instabilité du réseau et le risque de tomber dans des minimas locaux. Le risque d'instabilité du réseau peut être évité, en utilisant la règle Delta généralisée, au lieu de la règle Delta, tout court, Rumelhart et al. (1986). Ceci, en incluant un coefficient α positif, tel que, la règle d'apprentissage devient :

$$\Delta\ \omega_{ji}(n) = \alpha\ \Delta\ \omega_{ij}(n-1) + \eta\ \Psi_j(n)\ y_i(n)$$

L'introduction du coefficient α à un effet stabilisateur qui permet d'éviter

les oscillations dans la détermination des coefficients synaptiques.

.

3. Convergence de l'apprentissage

L'étude de la convergence du processus d'apprentissage vers un minimum de la fonction d'erreur soulève au moins deux questions : - Vers quel minimum la convergence s'effectue-t-elle ? - A quelle vitesse l'algorithme converge-t-il ? Dans un contexte linéaire, le minimum vers lequel converge l'algorithme est unique. Dans un contexte non linéaire, les choses se compliquent, singulièrement. Il peut y avoir de nombreux minimas

locaux, plusieurs minimas globaux. Dans ces conditions, le minimum, vers lequel converge l'algorithme de rétropropagation, sera fonction de son point de départ (initialisation aléatoire des connexions). Dans ces conditions, rien ne garantit que deux apprentissages successifs conduiront à une même estimation des paramètres du réseau. La vitesse de convergence est fonction de plusieurs facteurs : la complexité du problème (le nombre de variables d'entrée, le nombre de variables de sortie, le nombre de données dans l'échantillon d'apprentissage), la complexité de la topologie retenue (plus l'espace des paramètres est de dimension importante, plus la recherche d'un minimum acceptable sera lente et difficile) et le coefficient d'apprentissage choisi. La convergence de l'algorithme de rètropropagation du gradient ne peut être démontrée. De plus, il n'y a pas de critère bien défini pour définir la convergence. Un critère d'ordre pratique correspond au minimum global de la surface d'erreur. On peut définir un critère comme suit (Kramer & Sangiovanni- Vincentelli, 1989) : « La convergence de l'algorithme de rètropropagation du gradient est considérée comme atteinte quand la norme Euclidienne du vecteur gradient de l'erreur atteint un niveau, suffisamment, faible ». Une autre propriété du minimum est que l'erreur quadratique moyenne devient stationnaire au voisinage du minimum. Ainsi, un autre critère peut-être défini : « La convergence de l'algorithme de rètropropagation du gradient est considérée comme atteinte quand le taux absolu de variation de l'erreur quadratique moyenne par itération atteint un niveau, suffisamment, faible ». Le taux de variation est considéré, suffisamment, petit, quand il est compris dans l'intervalle de 0,1% à 1% par itération. Un tel critère peut provoquer un arrêt prématuré du processus d'apprentissage. Un autre critère de convergence : après chaque itération relative à l'apprentissage, le réseau est testé pour sa performance de généralisation. Le processus d'apprentissage est arrêté quand cette performance a atteint un niveau maximum (Early Stopping).**[22]**

4. Contrainte de généralisation et technique de « Cross- validation »

Un réseau conçu, pour bien généraliser, génère une correspondance correcte, même si, les inputs sont, légèrement, différents de l'exemple qui a permis l'apprentissage. Quand l'apprentissage se fait sur un échantillon d'une taille importante, le réseau de neurones peut capter un bruit présent dans l'échantillon d'apprentissage et qui est, complètement, étranger à la fonction qu'on cherche à modéliser. C'est le phénomène de sur-paramètrage ou de sur-apprentissage.**[22]**Ce phénomène réduit, considérablement, la capacité de généralisation du réseau. A ce niveau deux questions se posent : - Comment organiser l'apprentissage de manière à doter le MLP retenu de la meilleure capacité de généralisation possible ? - Comment contrôler le problème de sur-paramétrage ? L'objectif est d'utiliser un minimum de paramètres pour atteindre un niveau d'erreur donné afin d'améliorer d'autant la capacité de

généralisation. D'une manière générale, un excès de neurones dans les couches cachées (autrement dit, un excès de paramètres) réduit la capacité de généralisation du réseau. La généralisation est influencée, par trois facteurs : - La taille de l'échantillon de training, - L'architecture du réseau de neurones, - La complexité du problème traité. En ce qui concerne les deux premiers facteurs, la généralisation peut être vue de deux manières : - L'architecture du réseau est fixée et on détermine la taille de l'échantillon de training requise pour une bonne généralisation. - La taille de l'échantillon est fixée et on détermine la meilleure architecture du réseau qui permet une bonne généralisation. En pratique, pour une bonne généralisation, la taille de l'échantillon du training doit satisfaire la condition : le nombre d'observations (items) N doit être du même ordre de grandeur que le rapport (W/e). (W étant le nombre total de paramètres (poids synaptiques et biais) et e est l'erreur de classification permise dans l'échantillon test). Par exemple, avec une erreur de test de 10%, le nombre d'observations dans l'échantillon de training doit être autour de 10 fois le nombre de poids synaptiques dans le réseau. La technique de "cross-validation" est un outil statistique standard (Stone, 1974, 1978). L'échantillon d'observations disponible (les datas) est subdivisé d'une manière aléatoire entre un échantillon d'apprentissage et un échantillon test. L'échantillon d'apprentissage est, à son tour, subdivisé en deux échantillons : échantillon pour l'estimation des paramètres et un échantillon pour tester le modèle relatif à ces paramètres. Le but est de valider le modèle sur des données autres que celles utilisées pour l'apprentissage.

Ainsi, un bon apprentissage ne suffit pas, il faut aussi avoir des bonnes performances au niveau du test, autrement dit, une erreur de test minimale et un pourcentage de bon classement maximal au niveau de l'échantillon de test. La technique de "cross-validation" est utilisée surtout quand on cherche une bonne généralisation.

5. Les vertus et les limites de l'algorithme de rétropropagation du gradient

L'algorithme de la rétropropagation du gradient est, sans doute, le plus populaire des algorithmes d'apprentissage pour le perceptron multicouche. Il présente la simplicité de calcul du signal fonction et du signal erreur, au niveau de chaque neurone. Il permet une descente stochastique du gradient dans l'espace des poids synaptiques (puisque ces poids sont mis à jour, d'une observation à une autre, et que les observations sont relatives à des variables elles-mêmes stochastiques). Cette propriété fait que l'algorithme converge

vers le minimum global de l'erreur en faisant des zigzags autour de la vraie direction à partir de l'erreur initiale. Ces « zigzags » sont à l'origine de la lenteur de la convergence de l'erreur vers le minimum global.
L'inconvénient le plus évoqué de l'algorithme de rétropropagation du gradient est que le temps nécessaire à l'apprentissage évolue, exponentiellement, avec le nombre de neurones, sans aucune assurance que le minimum global de l'erreur sera atteint. A ce niveau, l'algorithme d'apprentissage « cascade correlation » s'avère plus avantageux.**[39]**

Chapitre 4

Modèles De Régression Logistique

1. Données binaires

1.1. Introduction

Supposons que pour une population *P*, on dispose d'une variable *Y* qui ne peut prendre que 2 valeurs. Par convention on associera à ces deux valeurs les valeurs canoniques 0 et 1.**[17]**

On dispose d'un *n*-uplet de cette population (Y_1, …., Y_n) de variables indépendantes, c'est-à-dire d'un tirage au hasard, avec remise de *n* individus de la population *P*, dont en peut connaitre la réponse Y_i pour chaque individu *i*.

On peut alors écrire **[17]**

$$P(Y=1)=\pi \text{ ou } P(Y=0)=1-\pi$$

Ou π est la probabilité de « succès ».

Dans la plupart des cas, les observations *Y* sont associées avec des variable explicatives (ou exogènes) $(X^1,\ldots, X^P)$. On dispose donc en réalité de *n*-uplets

$$(X_1^{\,1},\ldots, X_1^{\,P}, Y_1)$$
$$\vdots$$
$$(X_n^{\,1},\ldots, X_n^{\,P}, Y_n)$$

Ou les variables $(X^1,\ldots, X^P)$ peuvent être numériques, ou catégorielles. Le principal objectif d'une analyse statistique est d'étudier la relation entre la probabilité de réponse πi et les variables explicatives $(X^1,\ldots, X^P)$.

1.2 Données explicatives catégorielles [30]

On suppose ici que les variables explicatives sont catégorielles $(X^1,\ldots, X^P)$, c'est-à-dire qu'elles ne peuvent prendre qu'un nombre fini de valeurs possibles. Il est possible alors d'estimé les $P(Y=1/(X^1,\ldots,X^p))$ de façon efficace et sans biais (c'est-à-dire de façon optimale).

Prenons un exemple, on suppose que $p=2$, et que les valeurs possibles pour X^1 et X^2 soient 0 et 1. Les combinaisons possibles pour les variables $(X^1,\ldots, X^P)$ sont alors forcément fini, il est facile de les énumérer :

$$(X^1=1, X^2=1) := C_1$$

$$(X^1=1, X^2=0) := C_2$$

$$(X^1=0, X^2=1) := C_3$$

$$(X^1=0, X^2=0) := C_4$$

Alors pour notre n-uplet

$$(X_1{}^1,\ldots, X_1{}^P, Y_1)$$
$$\vdots$$
$$(X_n{}^1,\ldots, X_n{}^P, Y_n)$$

Lorsque tous les C_i apparaissent dans l'échantillon plus d'une fois, on parle de données répétées. Dans ce cas particulier, on peut étudier le modèle saturé :

$$\hat{\pi}_t = \frac{\sum_j (x_{j,}^1\ x_j^2) = C_i{}^{y_j}}{\sum_j (x_{j,}^1\ x_j^2) = C_i{}^1}$$

Qui est la proportion conditionnelle au classe C_i des réponses Y. on note alors

$$P(Y = 1/(X^1, X^2) = C_i) = \pi_i \approx \hat{\pi}_t = \frac{\sum_j (x_{j,}^1\ x_j^2) = C_i{}^{y_j}}{\sum_j (x_{j,}^1\ x_j^2) = C_i{}^1}$$

Ce qui revient à dire que $\pi=(\pi_1, \pi_2, \pi_3, \pi_4)$ est une fonction de (X^1,X^2)que l'on pourra donc noter

$$\pi(X^1,X^2)= (\pi(X^1=1, X^2=1),\ \pi(X^1=1, X^2=0), \pi(X^1=0, X^2=1), \pi(X^1=0, X^2=0)=(\pi_1, \pi_2, \pi_3, \pi_4)$$

2. Modèle Logit, Probit et autres

2.1. Paramétrisation

On considère, à priori, que les données explicatives sont vectorielles. Si jamais elles sont qualitatives, elles seront forcément codées sur des simplexes (des vecteurs, avec que des 0 et un 1). Notons que si une variable qualitative à K états possibles, alors β_0+B^TX aura $K+1$ paramètres si X est codé sur le simplexe de R^K, il faudra donc, en réalité coder X sur le simplexe de R^{K-1} et lui permettre de prendre la valeur nulle. De même, si il y a plusieurs variables qualitatives il faudra imaginer que toutes ces variables et leurs interaction sont regroupées dans une seule variable de bonne dimension. Nous verrons ultérieurement (modèles log linéaires) comment on peut décomposer ces croisements de variables en interactions. Le principe est exactement le même que l'analyse de la variance**.[15]**

Dans la suite on suppose que l'on regroupe toutes les variables quantitatives dans un vecteur X_{quant}, et les modalités possibles des variables qualitatives dans la variable X_{qual}. Pour étudier la relation entre la réponse π et les variables explicatives (X_{quant},X_{qual}), il est pratique de construire un modèle formel de la fonction π (X_{quant},X_{qual}).

Les modèles linéaires ont un rôle prépondérant en économétrie et en statistique car leurs propriétés sont généralement bien connues et ils sont relativement facilement interprétables. On va donc supposer que la dépendance entre Y et (X_{quant},X_{qual}) intervient grâce à une combinaison linéaire de (X_{quant},X_{qual}) :

$$\eta=\beta_0+\beta^T_{quant}X_{quant}+\beta^T_{qual}X_{qual}$$

pour des coefficients β_0, β_{quant}, β_{qual} inconnus. On remarque que généralement on aura $-\infty<\eta<\infty$, donc pour exprimer π à l'aide de η il faut utiliser une transformation g de π qui va de]0,1[dans R et modéliser alors

$$g(\pi_i)=\eta_i=\beta_0+\beta^T_{quant}X_{quant}+\beta^T_{qual}X_{qual}$$

pour $i=1,\ldots,n$. un choix standard de fonction g est :

1. la fonction logistique :

$$g(\pi)=ln(\frac{\pi}{1-\pi})$$

2. l'inverse de la distribution N(0,1) :

$$g(\pi)=\Phi^{-1}(\pi)$$

3. la fonction log-log :

$$g(\pi)= ln\ (- ln(1-\pi))$$

et plus généralement n'importe quel inverse de fonction de distribution de variable aléatoire de support *R*.

2.2 Le modèle logit [26]

2.2.1 Définition

Lorsque la fonction *g* (link function) est la fonction logistique, on dit que l'on a un modèle logit.

Si, Par exemple, on étudie un modèle logistique avec deux variables explicatives scalaire X_1 et X_2, on aura le modèle :

$$ln\ \frac{\pi(X^1,X^2)}{1-\pi(X^1,X^2)} = \beta_0+\beta_1X^1+\beta_2X^2$$

d'où le rapport des chances conditionnellement à $X=(X^1,X^2)$ vaudra

$$\frac{\pi(X^1,X^2)}{1-\pi(X^1,X^2)} = exp\ (\beta_0+\beta_1X^1+\beta_2X^2)$$

Soit

$$\pi(X^1,X^2)=(1-\pi(X^1,X^2))\ exp(\beta_0+\beta_1X^1+\beta_2X^2)$$

d'où

$$\pi(X^1,X^2)=\frac{exp(\beta0+\beta_1X^1+\beta_2X^2)}{1+exp(\beta0+\beta_1X^1+\beta_2X^2)}$$

par définition cette fonction est donc l'inverse de $g(\pi)=ln\ (\frac{\pi}{1-\pi})$

2.2.2 Interprétation des paramètres

En supposant que X^1 et X^2 sont indépendants, on peut dire les choses suivantes :

- Si on augmente X^2 d'une unité, on ''augmente''le rapport des chances en le multipliant par un facteur $exp\ (\beta_2)$, il est alors important que X^1 reste constant (d'où l'hypothèse d'indépendance de X^1 et X^2)

- La dérivée de $\pi(X^1,X^2)$ par rapport à X^2 vaudra

$$\frac{\partial \pi}{\partial X^2} = \pi\ (1-\pi)\beta_2$$

Donc un petit changement de x^2 aura un effet d'autant plus grand que la probabilité π est proche de 0,5.

- La façon la plus simple de représenter les résultats est certainement de donner les graphiques de

$$\pi(\eta) = \frac{exp(\eta)}{1+exp(\eta)}$$

en faisant varié les X^i. L'avantage de cette méthode est qu'elle marche aussi pour les ''link fuction'' probit et log-log, car ces ''link fuction''sont facile à inverser.

3. Estimation des modèles binaires

Dans cette section, toutes les variables explicatives (qualitative, quantitative et la constante) sont regroupés dans une variable générique X**.[16]**

3.1 Estimation par maximum de vraisemblance

On note $\pi_i=\pi(X_i;\beta)$la probabilité conditionnelle de l'observation i, on aura alors

$$L(X_1Y_1,\ldots,X_nY_n\ ;\beta)=\prod_{i=1}^{n} \pi_i^{Yi}\ (1-\pi i)^{1-Yi}$$

D'où la log-vraisemblance

$$ln(L(X_1Y_1,\ldots,X_nY_n\ ;\beta))=\sum_{i=1}^{n}[yi\ ln\left(\frac{\pi i}{1-\pi i}\right)+(1-yi)\ ln(1-\pi i)$$

il est alors aisé de calculer la dérivée de cette fonction paramétrique par rapport à chaque composante de β.

On montre que si la fonction de lien est log-concave alors il existe généralement un unique maximum $\hat{\beta}$ à la log-vraisemblance. Si le modèle est régulier (la matrice de Ficher du modèle est inversible) alors on a les résultats asymptotiques classiques de statistiques paramétrique ;

- Consistance du paramètre : $\hat{\beta} \rightarrow \beta_0$
- Normalité asymptotique :$\sqrt{n}\ (\hat{\beta}-\beta_0) \rightarrow N(0,I^{-1})$
- Test du rapport de vraisemblance, de Wald…..

3.2 Approche graphique : le logit empirique

Si les variables exogènes, *X* sont uniquement qualitatives, notons *t* ses modalités possibles, n_t le nombre d'observations où $X=t$ et y_t la somme de la variable *Y* pour les t observations où $X=t$. En, la fonction Logit empirique est définie par :

$$exp\ (\alpha+\beta^T X)=\frac{\frac{y_t}{n_t}}{1-\frac{y_t}{n_t}} \text{ d'où } \alpha+\beta x_t \approx \ln\left(\frac{y_t+\frac{1}{2}}{nt_t-y_t+\frac{1}{2}}\right)$$

La constante ½ interdit au numérateur et au dénominateur des prendre la valeur 0. La représentation du nuage des points $(x_t,z_t := \ln\left(\frac{y_t+\frac{1}{2}}{nt_t-y_t+\frac{1}{2}}\right))$ donne une approche empirique de la modélisation. Si $x \in R$ et si le nuage $\{(x_t,z_t), t=1,\ldots,T\}$ est proche d'une droite, on optera pour le modèle Logit affine $\alpha+\beta^T x$, si le nuage a une forme parabolique on préféra le modèle $\ln\left(\frac{y_t+\frac{1}{2}}{nt_t-y_t+\frac{1}{2}}\right) \approx \alpha+\beta x_+\gamma x^2$

Validation du modèle [15]

3.3.1 Etude des résidus

Pour les données répétées, les résidus estimés réduits sont, pour $t=1,\ldots,T$:

$$\varepsilon_t=\varepsilon(\pi_t)=\frac{y_t-n_t\widehat{\pi_t}}{\sqrt{n_t\widehat{\pi}_t\,(1-\widehat{\pi_t})}}$$

Pour nt grand, $\varepsilon_t \rightarrow N(0,1)$ si le modèle est valide, puisque y_t suit une loi binomiale $B(n_t,\pi_t)$. De fortes déviations de ε_t par rapport à la loi $N(0,1)$ incitent à corriger le modèle.

3.3.2 Test d'adéquation

Comme le modèle permet de prévoir des probabilités $(\widehat{\pi_t})$, on peut les comparer avec les probabilités observées. C'est sur ce principe qu'est basé le test de Hosmer et Lemeshow qui est très peu puissant.

3.3.3 Tableau de classification

Une observation i est affectée à la classe Y=1 si $\widehat{\pi_t} \geq c$. le tableau de classification est obtenu pour le choix mathématiquement optimal $c=0.5$. On paurra alors définir :

- La sensibilité : capacité à diagnostiquer les positifs parmi les positifs.

- La spécificité : capacité à reconnaitre les non positifs parmi les non positifs.
- 1-spécificité : faux positifs.

Un bon modèle trouvera un compromis acceptable entre forte sensibilité et forte spécificité. On utilise aussi parfois le graphique ROC en posant :

y=sensibilité (c)

x=1-sensibilité (c)

l'aire sous la courbe ROC est une représentation graphique du pouvoir prédictif de la variable X.

Chapitre 5

Application

1. Description des données

N°	Nom des variables	Variables	Type des variables	Modalités
1	**Etat/Compte**	état du compte chèque (en DM)	Qualitatif	1 < 0
				2 : ∈ [0, 200[
				3 : ≥ 200 ou versement des salaires pendant au moins un an
				4 : pas de compte chèque
2	**Durée/Crédit**	durée en mois du crédit ∈ N	Quantitative	
3	**Hist**	historique des crédits	Qualitatif	0 : pas de crédit / tous remboursés
				1 : tous les crédits dans la banque remboursés
				2 : crédits en cours
				3 : retard de paiement dans le passé
				4 : compte critique / crédit existant dans d'autre banque
4	**But**	but du crédit	Qualitatif	0 : voiture neuve
				1 : voiture occasion
				2 : équipement / fourniture
				3 : radio / télévision
				4 : appareils ménagers
				5 : réparation
				6 : éducation
				7 : vacances
				8 : recyclage
				9 : professionnel
				10 : autre
5	**Montant/crédit**	montant du crédit (en DM) ∈ R	Quantitative	
6	**Montant/épargne**	montant de l'épargne (en DM)	Qualitatif	1 : < 100
				2 : ∈ [100, 500[
				3 : ∈ [500, 100[
				4 : ≥ 1000
				5 : inconnu
7	**Anc**	ancienneté dans le travail actuel (an)	Qualitatif	1 : sans emploi
				2 : < 1
				3 : ∈[1, 4[
				4 : ∈ [4, 7[
				5 : ≥ 7

8	**Taux/Apport**	taux d'apport ∈ R	Quantitative	
9	**S/F**	état marital	Qualitatif	1 : homme divorcé / séparé
				2 : femme divorcé / séparé / mariée
				3 : homme célibataire
				4 : homme marié / veuf
				5 : femme célibataire
10	**Garant**	autre demandeurs / garants	Qualitatif	1 : aucun
				2 : co-demandeur
				3 : garant
11	**Durée /Habitation**	durée d'habitation ∈ N dans la résidence actuelle (an)	Quantitative	
12	**Biens**	biens	Qualitatif	1 : immobilier
				2 : si pas
				1 : placement (assurance vie ou part dans la banque)
				3 : si pas
				1 et 2 : voiture ou autre, non compris dans la variable 6
				4 : inconnu
13	**Age**	âge (an) ∈ N	Quantitative	
14	**Autres/Crédits**	autre demande de crédits	Qualitatif	1 : banque
				2 : magasins
				3 : aucun
15	**Résid/Actu**	situation dans la résidence actuelle	Qualitatif	1 : locataire
				2 : propriétaire
				3 : occupant à titre gratuit
16	**Nbre/Crédit**	nombre de crédits dans la banque ∈ N	Quantitative	
17	**Emploi**	emploi	Qualitatif	1 : sans emploi / non qualifié – étranger
				2 : non qualifié - non étranger
				3 : emploi qualifié / fonctionnaire
				4 : gestion / indépendant / emploi hautement qualifié / haut fonctionnaire
18	**Nbre/Per/Rem**	nombre de personnes pouvant ∈ N rembourser le crédit	Quantitative	
19	**Téléphone**	téléphone	Qualitatif	1 : aucun
				2 : oui, enregistré au nom du client
20	**Etranger**	travailleur étranger	Qualitatif	1 : oui
				2 : non

le fichier "german.data" a 1000 observations (crédits accordée par la DeutshBank de Munich (LMU, 1994)) classées bon et mauvais client: 30% sont considérés comme mauvais client et 70% comme bon client.
Nous disposons de 20 variables (7 quantitatives, 13 qualitatives).

Une variable qui nous intéresse particulièrement, c'est la variable Réponse qui sert à distinguer les bons clients de la banque de ceux des qui ne le sont pas.

A- Les arbres de décision sous R avec rpart (Recursive partitionning)

Nous chargeons les données et faisons quelques vérifications :

> g.data<-read.table("germanca.txt")

> dim(g.data)

[1] 1000 21

> names(g.data)<-
c("etat_compte","durée_credit","hist_crédit","but_crédit","montant_crédit","montant_epargne","anc_travail","taux_apport","s_f","garant","durée_hab","biens","age","autres_crédit","résid_actu","nbre_crédit","emploi","nbre_per_rem","téléphone","etranger","réponse")

> names(g.data)

[1] "etat_compte" "durée_credit" "hist_crédit" "but_crédit"

[5] "montant_crédit" "montant_epargne" "anc_travail" "taux_apport"

[9] "s_f" "garant" "durée_hab" "biens"

[13] "age" "autres_crédit" "résid_actu" "nbre_crédit"

[17] "emploi" "nbre_per_rem" "téléphone" "etranger"

> levels(g.data$réponse)

[1] "bon" "mauv"

> table(g.data$réponse)

bon mauv

700 300

Nous chargeons le programme rpart et définisons l'ensemble d'apprentissage (60% des

données) et l'ensemble test (40% des données):

> **library(rpart)**

> **d=sort(sample(nrow(g.data),nrow(g.data)*.6))**

> **train<-g.data[d,]**

> **test<-g.data[-d,]**

> **dim(g.data)**

[1] 1000 21

> **dim(train)**

[1] 600 21

> **dim(test)**

[1] 400 21

> **data=train**

> **dim(data)**

[1] 600 21

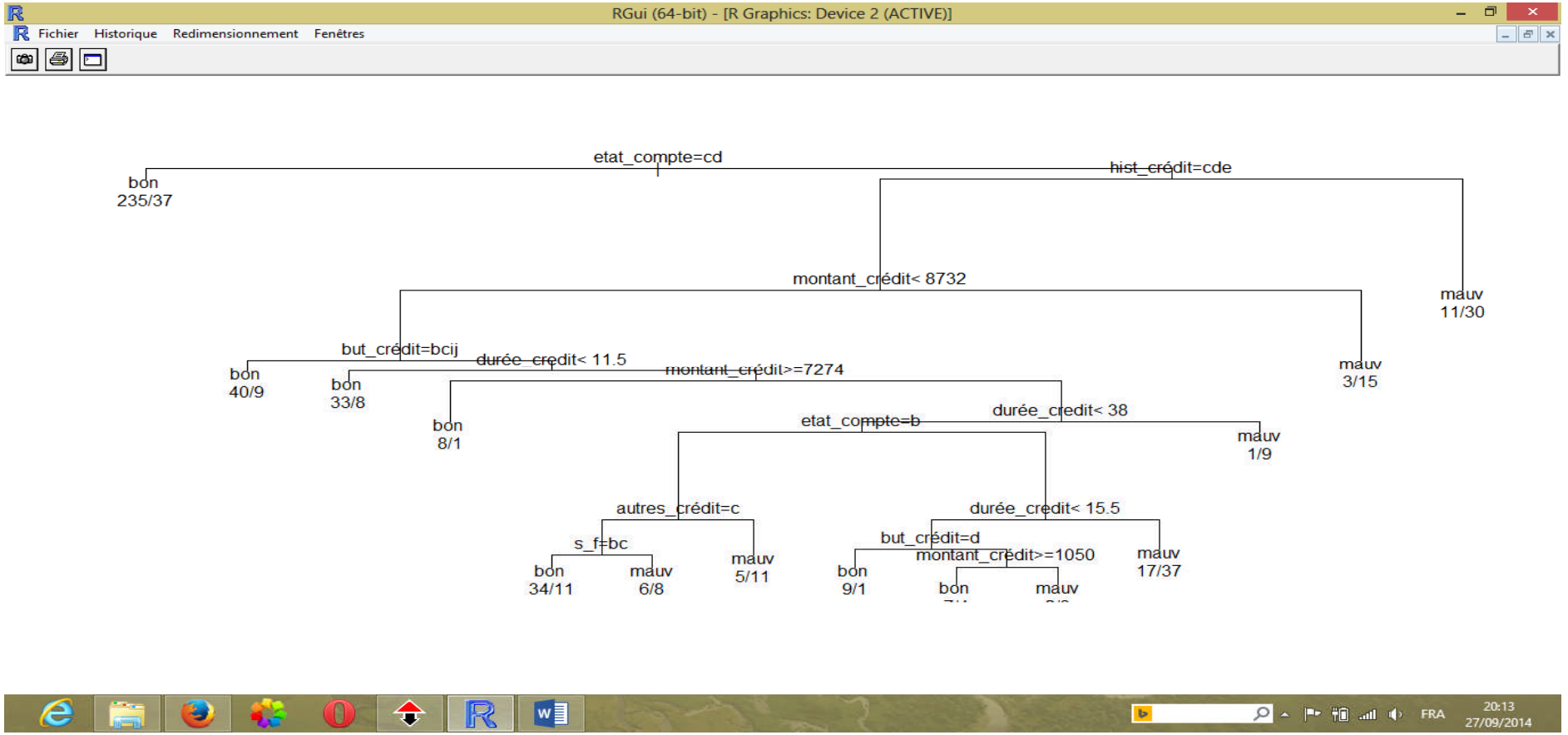
Nous construisons le 1 er arbre sur l'ensemble d'apprentissage :

> **arbre.data<-rpart(réponse~.,data,method="class")**

> **print(arbre.data)**

> **plot(arbre.data)**

> **text(arbre.data,use.n=T)**



A partir de cet arbre nous allons obtenir les règles de décisions :

Nous allons donner les résultats de cette instruction (règles données sous la forme de Rule number) et commenter l'une d'entre elle

Rule number: 207 [réponse=mauv cover=10 (2%) prob=0.90]

La feuille contient 10 clients et est labélisée « mauvais » avec 1 bon (1/10=0.1) et 9 mauvais clients (9/10=0.9)

A titre d'exemple de règles de décision , au niveau du nœud (durée du crédit<38) si un client possède les attributs suivants :

- **etat_compte=A11,A12** (l'état du compte est inferieur à 0 DM ou entre 0 et 200 DM)
- **hist_crédit=A32,A33,A34** (le client a des crédits en cours, retard de paiement dans le passé ou bien qu'il a un crédit dans une autre banque)
- **montant_crédit< 8732** (le montant du crédit est inferieur à 8732)
- **but_crédit=A40,A42,A43,A44,A45,A46** (le crédit est fait pour acheter une voiture neuve, équipement, radio, télévision, appareils ménagers, pour réparation ou éducation)
- **durée_credit>=11.5** (la durée du crédit est superieur ou égale à 11 mois et demi)
- **montant_crédit< 7274** (le montant du crédit et inferieur à 7274)
- **durée_credit>=38** (la durée du crédit est superieur ou égale à 38 mois)

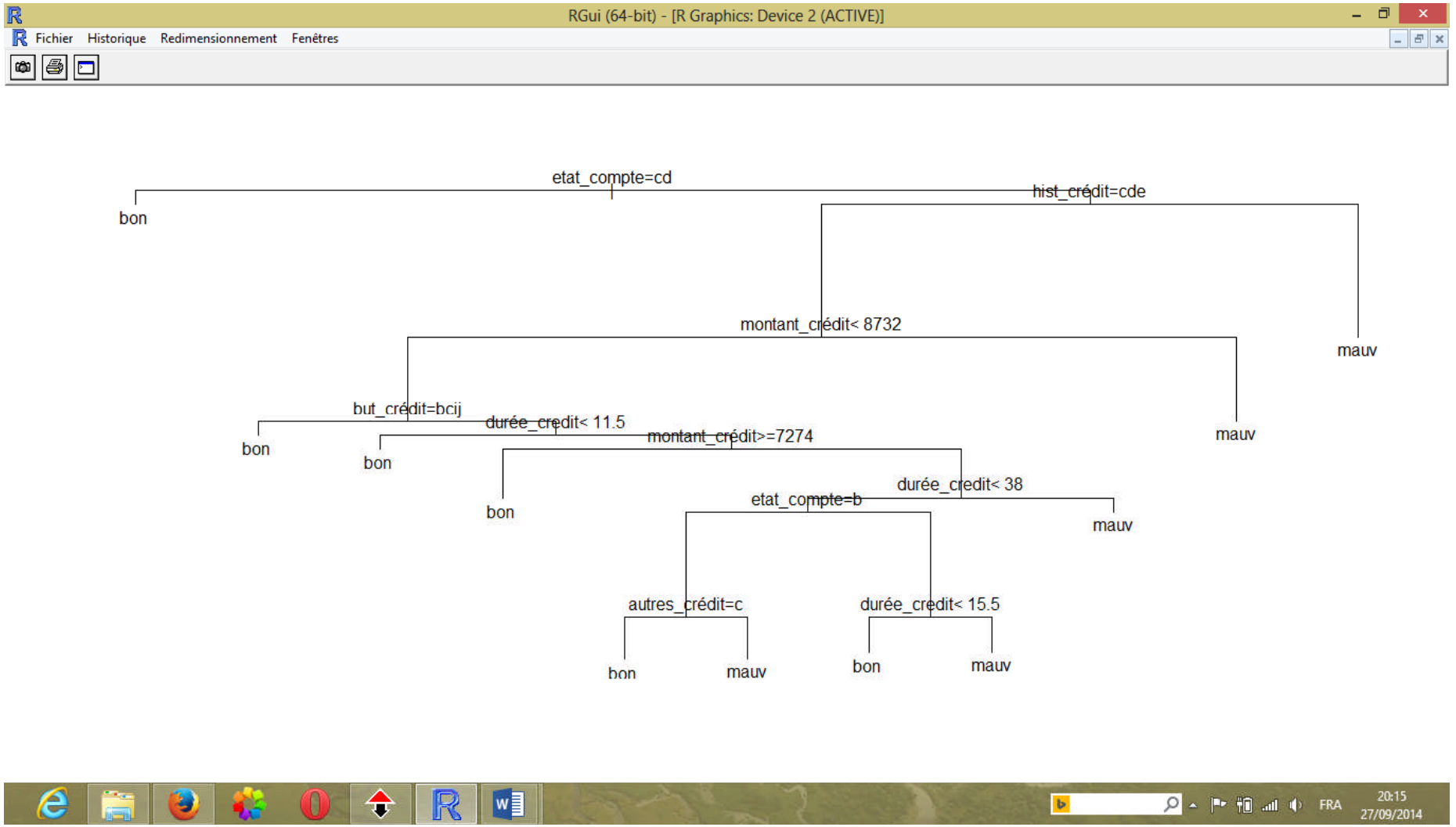
il est classé comme un mauvais client.

nous allons procéder à l'élagage de l'arbre avec la procédure prune –prune, arbre,cp coéfficient de compléxité qui peut prendre des valeurs inferieures a 0.1

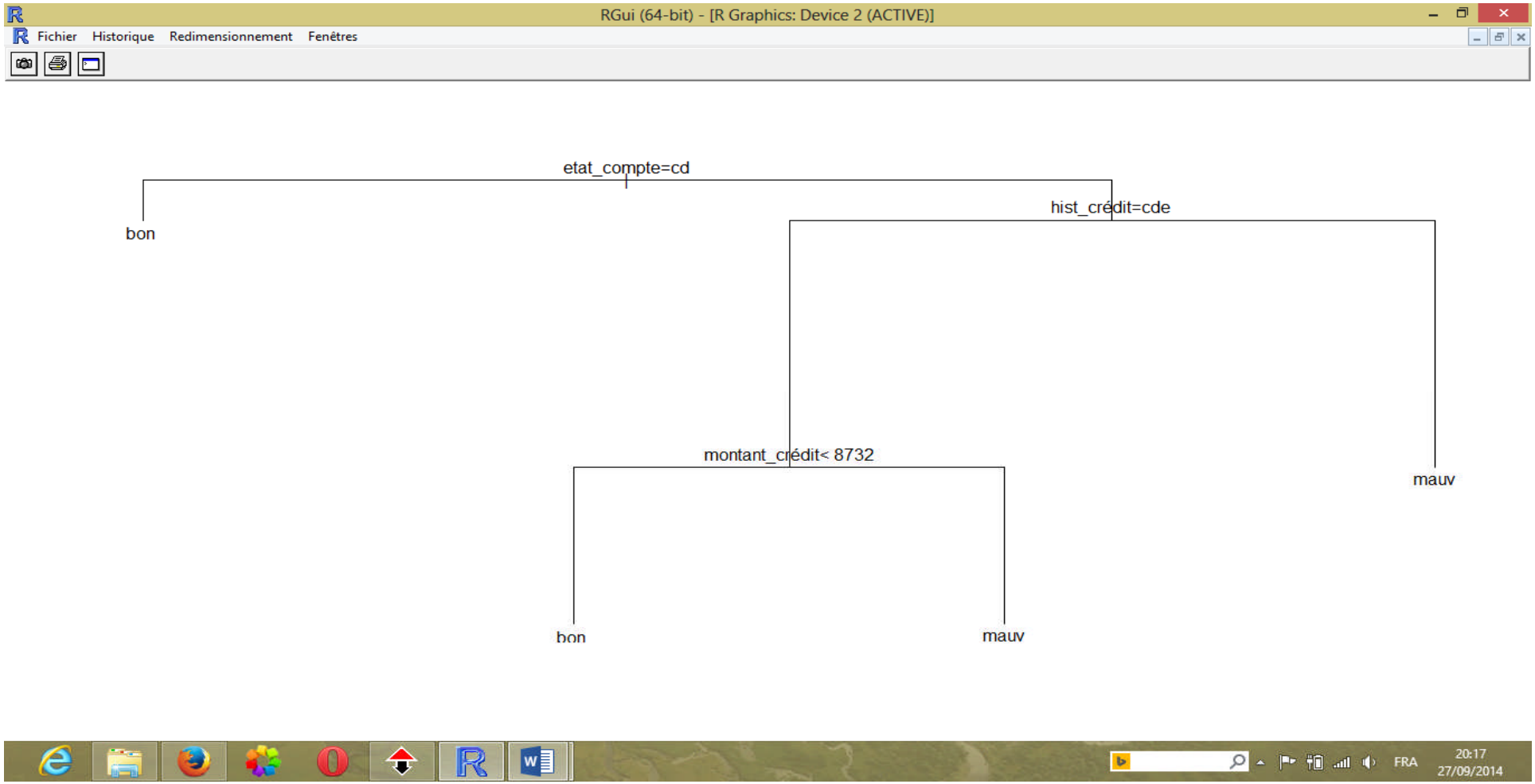
La sortie ne contient pas d'arbre et corréspond à l'ensemble d'apprentissage(racine)

En fait, pour cp variant de 0.01 à 0.07 on obtient toujours l'ensemble racine. L'arbre est élagué completement
Pour cp=0.06 on obtient un arbre élagué partiel
On fait varier la cp

> arbre2<-prune(arbre.data,cp=0.02)



> arbre3<-prune(arbre.data,cp=0.05)



Matrice de confusion pour le modèle Arbre de décision sur german.txt [test] (comptes) :

Réel	Prédit bon	Prédit mauv
bon	119	20
mauv	28	33

Matrice de confusion pour le modèle Arbre de décision sur german.txt [test] (%) :

Réel	Prédit bon	Prédit mauv
bon	60	10
mauv	14	16

Erreur générale : 0.24

Le modèle a correctement prévu les résultats pour 76 % des (bon classements) en revanche la prévision est fausse pour 24% d'entre eux.

B. Les réseaux de neurones :

Rappelons brièvement que l'objet de cette partie est d'explorer une nouvelle démarche basée sur les réseaux de neurones artificiels afin d'améliorer l'aide à la décision du banquier.

L'intérêt de cette méthode réside dans le fait qu'elle ne nécessite aucune hypothèses de départ ni de normalité ni de linéarité des variables étudiées. De plus, elle permet d'utiliser aussi bien des variables quantitatives que qualitatives en même temps.

Notre base de données est constituée de 1000 dossiers de crédit accordés par DeutshBank de Munich (LMU, 1994)

En sachant que les modèles des réseaux de neurones se construisent par apprentissage à partir d'un certain nombre d'exemples (les dossiers de crédit dans notre cas). Nous avons utilisé uniquement 600 dossiers pour l'échantillon d'apprentissage et nous avons laissé de côté 400 dossiers pour l'échantillon test afin de tester la capacité prédictive du réseau.
.

> library(nnet)

Le modèle avec 0 couche cachées :(perceptron)

Résumé du modèle Neurones net (construit avec 'nnet') :

A 48-0-1 network with 49 weights.
Inputs: etat_compteA12, etat_compteA13, etat_compteA14, duree_credit, hist_creditA31, hist_creditA32, hist_creditA33, hist_creditA34, but_creditA41, but_creditA410, but_creditA42, but_creditA43, but_creditA44, but_creditA45, but_creditA46, but_creditA48, but_creditA49, montant_credit, montant_epargneA62, montant_epargneA63, montant_epargneA64, montant_epargneA65, anc_travailA72, anc_travailA73, anc_travailA74, anc_travailA75, taux_apport, s_fA92, s_fA93, s_fA94, garantA102, garantA103, duree_hab, biensA122, biensA123, biensA124, age, autres_creditA142, autres_creditA143, resid_actuA152, resid_actuA153, nbre_credit, emploiA172, emploiA173, emploiA174, nbre_per_rem, telephoneA192, etrangerA202.
Output: as.factor(reponse).
Sum of Squares Residuals: 210.0000.

Options de construction du réseau de neurones : skip-layer connections; entropy fitting.
In the following table:
b represents the bias associated with a node

h1 represents hidden layer node 1
i1 represents input node 1 (i.e., input variable 1)
o represents the output node

Weights for node o:

b->o	i1->o	i2->o	i3->o	i4->o	i5->o	i6->o	i7->o	i8->o	i9->o	i10->o
-0.66	0.23	0.29	-0.31	-0.68	-0.36	0.27	0.23	-0.31	-0.18	0.31

i11->o	i12->o	i13->o	i14->o	i15->o	i16->o	i17->o	i18->o	i19->o	i20->o	i21->o
-0.02	0.29	-0.50	0.39	0.25	-0.16	-0.55	-0.52	0.25	-0.65	-0.15

i22->o	i23->o	i24->o	i25->o	i26->o	i27->o	i28->o	i29->o	i30->o	i31->o	i32->o
-0.03	-0.20	0.30	-0.16	-0.04	0.49	0.56	0.44	0.41	0.51	0.38

i33->o	i34->o	i35->o	i36->o	i37->o	i38->o	i39->o	i40->o	i41->o	i42->o	i43->o
0.22	0.47	-0.41	0.15	-0.22	0.46	-0.08	-0.41	0.33	-0.54	0.56

i44->o	i45->o	i46->o	i47->o	i48->o
0.59	0.64	0.13	-0.68	-0.51

Matrice de confusion pour le modèle Réseau de neurones sur german.txt [test] (comptes) :

	Prédit
Réel	bon
bon	100
mauv	50

Matrice de confusion pour le modèle Réseau de neurones sur german.txt [test] (%) :

	Prédit
Réel	bon
bon	67
mauv	33

Erreur générale : 0.33

Le modèle a correctement prévu les résultats pour 67 % des (bon classements) en revanche la prévision est fausse pour 33% d'entre eux.

C. la régression logistique

> m1<-glm(réponse~.,data=g.data.train,family=binomial(link=logit))

> summary(m1<-glm(réponse~.,data=g.data.train,family=binomial(link=logit)))

Call:

glm(formula = réponse ~ ., family = binomial(link = logit), data = g.data.train)

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.0534	-0.7119	-0.3497	0.6475	2.6029

Coefficients:

	Estimate	**Std. Error**	**z value**	**Pr(>|z|)**
etat_compteA12	**-3.713**	2.757	-1.347	0.178028
durée_credit	**4.190**	1.196	3.503	0.000460
hist_créditA31	**2.517**	6.596	0.382	0.702755
but_créditA410	**-1.420**	8.504	-1.669	0.095042 .

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 832.02 on 699 degrees of freedom

Residual deviance: 611.38 on 651 degrees of freedom

AIC: 709.38

Number of Fisher Scoring iterations: 5

Le modèle s’écrit sous la forme:

g(X)= -3.713 état _compte+4.190 durée_credit +2.517 historique _credit -1.42 but _crédit

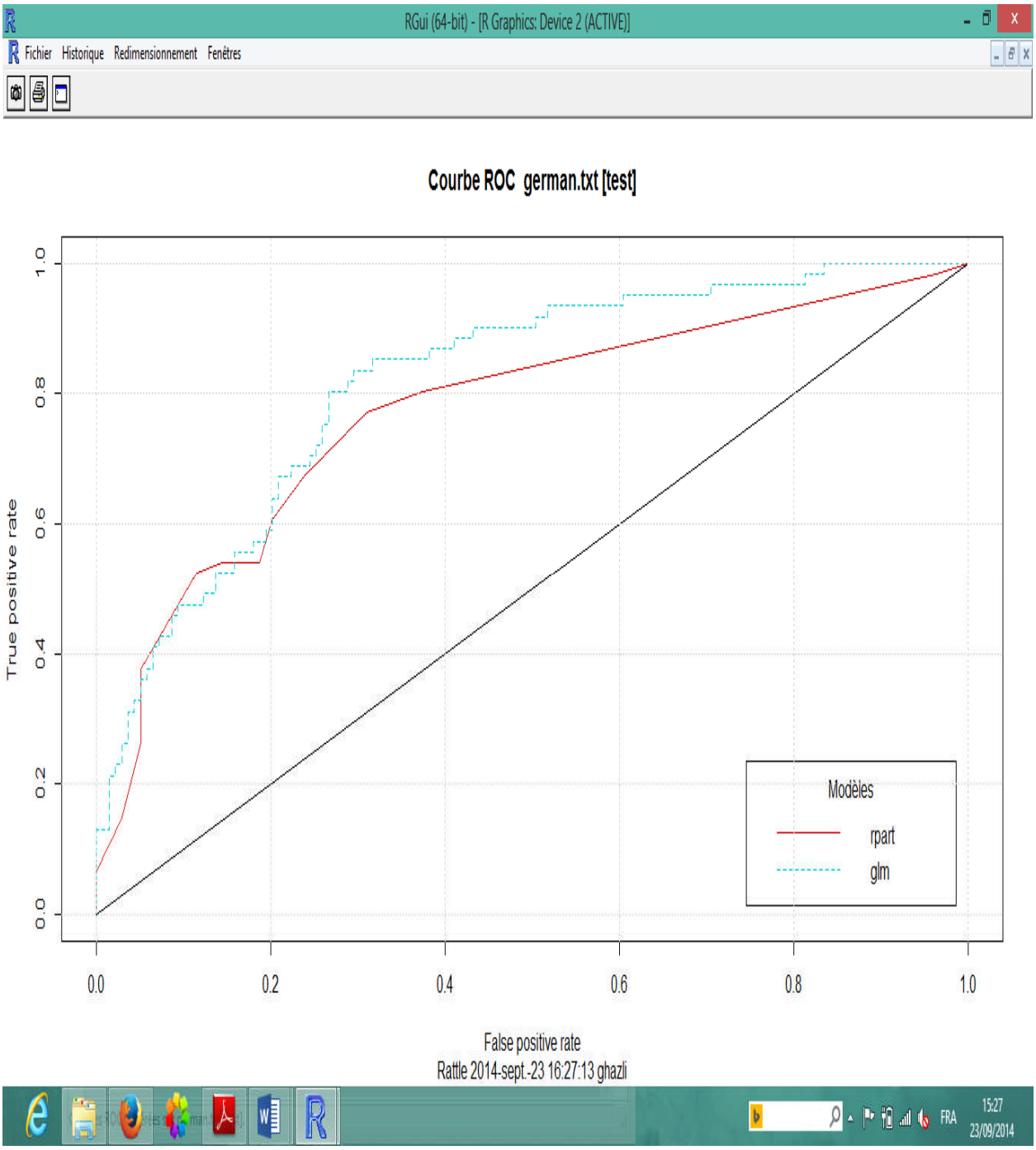
Matrice de confusion pour le modèle Linéaire sur german.txt [test] (comptes) :

	Prédit	
Réel	bon	mauv
bon	120	19
mauv	29	32

Matrice de confusion pour le modèle Linéaire sur german.txt [test] (%) :

	Prédit	
Réel	bon	mauv
bon	60	10
mauv	14	16

Erreur générale : 0.24


RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
Fichier Historique Redimensionnement Fenêtres
Courbe ROC german.txt [test]
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Modèles
rpart
glm
False positive rate
Rattle 2014-sept.-23 16:27:13 ghazli
FRA
15:27
23/09/2014

Conclusion Générale

La qualité d'information possédée par une banque sur les clients est cruciale dans l'évaluation des demandes de prêts. En effets les modèles de prédiction utilisés par la majorité des banques ne permettent pas de prédire le comportement d'un emprunteur (solvable/non solvable) provenant d'un échantillon de taille restreinte.

Dans le cadre de ce mémoire nous nous somme intéressé à l'évaluation du risque de crédit en présence des données d'une banque. Pour cela, nous avons considéré trois techniques de modélisation.

La première technique considérée dans ce travail est la technique des arbres de décisions. En se basant sur un jeu de données composé de 1000 observations représentant les crédits aux consommations accordées par la DeutshBank de Munich (LMU, 1994).

La deuxième technique est la technique de la régression logistique qui est l'une des techniques les plus utilisées en prédiction de risque lorsque la variable de réponse est dichotomique. L'utilisation de cette première technique nécessite la possession d'un échantillon d'apprentissage de taille suffisante.

Les résultats trouvés dans cette étude viennent pour confirmer que les arbres de décision et les réseaux de neurones ne sont pas des outils fiables en scoring.

Dans certains temps le modèle basé sur la régression logistique donne le meilleur résultat en présence d'une autre procédure comme les arbres de décisions qui est couteuse en termes de temps.

Bibliographie

1. Gestion bancaire : risque de non remboursement des crédits aux entreprises : une revue de la littérature Véronique Rongés Université Paris I Discipline centre de la recherche européenne en finance et gestion.
2. Développements récents de la méthode des scores de la banque de France –bordos (2001)- bullettin de la banque de France N° 9 Juin 2001
3. Quantitative méthods in credit management : Asurvey-operation research revue 42 , 1994 Rosenberd et Genit (1994)
4. Credit risk development over the last 20 years journal berleing and finance 1998. Wiley Altman.saunders, 1998
5. Credit risk new approaches to value at risk and other paradigms new York wiley A.saunders, 1999
6. Analyse discriminante –application au risqué et scoring financier Dunod (2001)
7. Scoring sur données d'entreprises : instrument de diagnostique individuel et outil d'analyse de portefeuille d'une clientèle Michelle bardos observation de la banque de France : revue madulad 2008 N° 38.
8. An introduction to multivariante statistical analysis Wiley (1984)
9. Comparing decisions trees with logistic regression for crédit risk analysis S.S Satchidunanda International Institute M Information Technology Bangecore India Et Jay B.Simha Abissa Systemes Bengecore India
10. Prévision des risques des credits de gestion par des modèles basés sur les réseaux de neurones artificiels Matoussi Hamadi Professeur ISCAE Tunisie et Krichene Abdelmopula Aida FSEG-Sfax-Tunisie
11. Analyse bancaire de l'entreprise –Paris économie 5 eme édition (collection economica institut technologique de banque EMANCHOUN (2001)
12. Credit scoring and its applications (monographic on mathematical modeling and computation) L.C.Tomas D.B Edelman I.N.Crook-Siammonographie
13. What is the point of credit scoring review I.J MESTER (1997) Bussenis Federas Reserve Bank of Zhile Delphic Sept/Oct 1997
14. Financial ratios as predictors of failure W.H.Beaver Journal of

accounting researd 1966 vol 4
15. Régression logistique Gilles Gasso INSA Rouen -Département ASI Laboratoire LITIS
16. Merbouha et A. Mkhadri : Méthodes de scoring non-paramètriques. Revue de Statistique Appliquée, 56(1):5,26, 2006.
17. G. Saporta : Credit scoring, statistique et apprentissage. In EGC, pages 3et 4, 2006.
18. Evaluation statistique du risque de credit par la technique du scoring : Cas de Afriland First Bank TENE Georges Colince.
19. Un modele de “credit scoring” pour une institution de micro-finance africaine: le cas de nyesigiso au mali boubacar diallo laboratoire d’economie d’orléans (leo), université d’orléans mai 2006.
20. Evolution des options et gestion des risques financiers par les réseaux de neurones et par les volatilités stochastiques, thèse de doctorat Yacine JERBI université de Sfax.
21. Les réseaux de neurones G. DREYFUS École Supérieure de Physique et de Chimie Industrielles de la Ville de Paris (ESPCI).
22. Guide to Credit Scoring in R.
23. Introduction à la régression logistique Maryse Raffestin – mai 2010.
24. Scoring sur données d’entreprises : instrument de diagnostic individuel et outil d’analyse de portefeuille d’une clientèle Mireille Bardos.
25. Fouille des données Notes de cours Ph. PREUX Université de Lille 3.
26. Assadi Réza & Khattar Karim.(1mai 2002).L’utilisation d’un réseau de neurones pour optimiser la gestion d’un firewall. Ecole polytechnique de Montréal.Québec,Canada.
27. Bakiri.F & Benmiloude .M.(1991).Maladie des Glandes Endocrines. INESSM .O.P.U. Alger
28. Benahmed Nadia. (Mars 2002) . Optimisation de réseau de neurons pour la reconnaissance de chiffres manuscrits isoles:selection et pondération des primitivites par l’algoritmes Génétique.Ecole de technologie supérieure.Université de Québec.Canada.
29. Bishop Christopher M.(1995).Neural networks for pattern recognition.Birmingham:Clarendon press Oxford.
30. Claude TOUZET.(Sptembre 1998).Réseaux de neurons artificiels a la robotique cooperative. Université de Droit,d’Economie et des Sciences d’Aix-Marseille III Faculté des Sciences et Techniquess de Saint-Jérôme.(France)
31. Culloch W.S.Mc et Pitts W.(1943) A logical calculs of the idea immanent in Nervous activity.Bulletin of Mathmatical Biophysics,5,p 115-133 al.

32. Crucianu Mihail.(Avril 1997).Algorithmes d'évlolution pour les réseaux de neurons.Laboratoire d'informatique;Ecole d'ingénieurs en Informatique pour l'industrie (France)
33. C.Berge.(1973).Graques et hypergraphes,Bordas,deuxiéme edition
34. Dreyfus G.,Martinez J.M.,Samueliese M. (2004).Réseaux de neurons méthodologie et application. Edition Eyrolles.
35. Duchesne Thierry(2004).département de mathémétique et de statistique Université Laval. Canada.
36. F-C.chen,(1990).Back-propagation neural network for nonlinear self-tuning adaptative control,IEEE control system Magazine,pp 44-48.
37. Guyon L.,Poujaud I.,Personnaz L.,Dreyfus G.,Denker J., & le Cun Y.(1989). Comparing different neural networks architectures for classifying handwritten digits2. Int .J. Conf.on Neural Networks,2,127-132.Wachington,USA
38. Hoie.J,collaborateurs.(1998).Distant Metastases in papillary Thyoïd Cancer.Cancer 61:1-6
39. Jodouin Jean-François(1994). Les réseaux de neurons:principes et definitions. Edition Hermes.
40. Kary FRÄMLING.(Juillet 1992).Lesréseaux de neurons comme outils d'aodé à la decision floue Ecole Nationale Supérieure des Mines de Saint-Etienne.France
41. Krranowsky.w.j.(25-27.April 1995)discrimination and Classification Using Both binary tree hypothesis.Proc Of the ECML 95,8th European Conference on ML,Heraklion,Grete,Greece.
42. L.L.,Alin Morineau,M.P.(juilet 1995).Statistique exploratoire multimensionnelle.Paris.
43. Lerman .I.C & Ph.Peter.(juillet1985) Elaboration et logiciel d'un indice de similarité entre objets d'un type quelconque.Application au probléme du consensus en classification. Publication Interne Irisa n.262,Rennes.
44. Medsker Larry,Liebowitz Jay(1994).Design and development of Expert Systems and Neural Networks.Edition MacMillan Publishing Company.
45. Morgan.J.N&J.A.Sonquist.(1963).Problems in analysis of survy data and a prosal.J.Am.Statist.Assoc,58,pp.415-434.
46. O.Kinouchi.M.H.R(1996).Tragtenberg, «Modeling neurons by simple maps» international journal of bifurcation and chos,vol.N12 A,pp.150-1560
47. Richard O.Duda,Peter E .Hart, & David G.Stork. (2001).Pattern

Classification.(Second edition).New York:Wiley-Interscience.
48. Rival Isabelle.LES Réseaux de Neurones Formels pour pilotage de Robots Mobiles, Laboratoire d'Electronique de l'ESPCI (Ecole Supérieur de Physique et de Chimie Industrielles).
49. Rivals I.,Personnaz L.,Dreyfus G.,Ploix J.-L.(1995).Modélisation classification et commande par réseaux de neurons:principes fondamentaux. méthodologie de conception,et illustrations industrielles,in Récents progress en genie des proceeds 9,Lavoisier technique et documentation.Paris.
50. Rosenblatt,R.(1958).Principles of Neurodynamics.Spartan Books.New-York
51. Zurada J.M.(1992).Introduction to artificial neural systems.Minnesota:West