REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MENTOURI CONSTANTINE
FACULTE DES SCIENCES EXACTES
DEPARTEMENT DE MATHEMATIQUES

N° d'ordre:
Série:

MEMOIRE
EN VUE DE L'OBTENTION DU DIPLOME DE
MAGISTER EN MATHEMATIQUES

OPTION : MODELISATION ET ANALYSE STATISTIQUE

THEME

L'ANALYSE DISCRIMINANTE ET LE PERCEPTRON MULTICOUCHE

PRESENTE PAR:

OTMANI Imene

Soutenu le: 02\06\2011

Devant le jury:

Président	**S. BOUGHABA**	**M.C.**	**à l'Université Mentouri**
Rapporteur	**F. L. RAH MANI**	**M.C**	**à l'Université Mentour**
Examinateur	**Z. MOHDEB**	**Prof.**	**à l'Université Mentouri**
Examinateur	**Z. GHERIBI**	**M.C.**	**à l'Université Mentouri**

Table des matières

Introduction générale

La modélisation constitue le parachèvement d'un travail statistique quand il arrive à sa pleine maturité. En effet, établir le modèle qui gère un phénomène signifie que nous avons compris et maîtrisé tous les tenants et les aboutissants de ce phénomène. Le modèle met en lumière les facteurs intervenants, le degré de leurs influences et les relations exactes qui les lient. La puissance et l'intérêt de la statistique prennent toute leur ampleur dans la modélisation qui permet un fait capital : l'extrapolation à la population entière de l'information récoltée dans l'échantillon d'étude. Le modèle doit rester valable en dehors des données qui ont servi pour son élaboration.

C'est ce qui fait que la régression soit un domaine central de la statistique et qui revêt une grande importance. Beaucoup de domaines s'adossent sur cette discipline et beaucoup d'autres y prennent leurs racines et leurs fondements tels que, entre autres, les séries chronologiques et l'économétrie.

La régression est un domaine vaste qui a fait l'objet de beaucoup d'études. Certains de ses aspects sont complètement maîtrisés ; d'autres présentent encore des difficultés.

La part la plus importante des travaux a été consacrée à la régression linéaire, qui est la plus largement utilisée. Elle est devenue un outil compris et consacré, ne présentant pas de problèmes sauf lorsque les variables explicatives sont fortement corrélées. Mais même pour cette question, qui fait que l'estimateur des moindres carrés soit instable, des estimateurs concurrents ont été proposés et ont permis de dépasser l'obstacle.

La vraie grande question reste celle lorsqu'il y a non linéarité du modèle. Les difficultés sont essentiellement d'ordre technique. Il est beaucoup plus complexe de mener une régression non linéaire qu'une régression linéaire. Des solutions partielles sont proposées pour traiter des familles particulières de fonctions (régression logistique, régression polynômiale, ...) mais le problème de fond persiste ; l'inexistence d'une méthode unifiée et générale conçue pour effectuer toute sorte de régression non linéaire.

C'est une discipline, d'origine multidisciplinaire, mais qui est aujourd'hui parfaitement assimilé à une extension des méthodes statistiques, qui depuis quelques décades, vient apporter les outils pour traiter toutes les questions qui sont restées en suspens. Ce sont les réseaux de neurones artificiels. Ils permettent de reproduire tout aussi bien les méthodes standards de la statistique comme ils permettent de surmonter les problèmes relevant du non linéaire. Ils sont aptes à apporter des solutions nouvelles et à juguler des problèmes traditionnellement classés difficiles.

Dans notre travail, nous avons essayé de mener une étude comparative opposant les techniques conventionnelles de régression et de discrimination à

ce qui, dans les réseaux de neurones, se présentent comme équivalents.

Les deux premiers chapitres exposent brièvement les notions de régression puis les notions de discrimination qui, sous un certain angle, n'en sont que des corollaires.

Le chapitre trois introduit les réseaux de neurones, en focalisant sur les paramètres qui permettent de construire les outils appropriés pour effectuer les tâches de classification (ou de discrimination) et les tâches de modélisation.

Ce chapitre s'achève avec la présentation du perceptron multicouche qui permet le traitement efficace des problèmes non linéaires, que cela soit en termes de régression ou en termes de classification.

Le chapitre quatre réunit les éléments préalables et expose l'algorithme de la rétro-propagation du gradient, qui est l'algorithme d'apprentissage le plus couramment utilisé dans les réseaux multicouche à propagation avant.

Les problèmes spécifiques des réseaux de neurones sont ceux de la stabilité et de la convergence : l'apprentissage étant itératif, les réseaux peuvent dans certains cas osciller autour de la solution sans l'atteindre, ils peuvent dans d'autres cas ne pas converger vers la solution. Le chapitre cinq expose les algorithmes à utiliser dans pareils cas.

Chapitre 1

La régression statistique

Dans ce chapitre, nous tenterons de voir des concepts traditionnels à propos de la régression, qui est un ensemble de techniques analytiques des données. Le but de la régression est de déterminer la relation suivante :

$$y = f(x) + \varepsilon \tag{1.1}$$

où$f(.)$ est une fonction déterministe, et ε est une variable aléatoire. Pour cela, il y a deux difficultés propres qui sont :

∘ La fonction de régression $f(.)$ (déterministe) peut avoir une forme analytique quelconque, mais est de toute façon inconnue.

∘ ε est une variable aléatoire de moyenne nulle, mais de distribution inconnue. (dépend possiblement de x (hétéroscédasticité)).

D'après la relation entre les variables, on a distingué deux catégories : la régression paramétrique et la régression non paramétrique.

1.1 La régression paramétrique

La régression paramétrique présente la relation entre les variables explicatives et les variables expliquées, en terme de forme fonctionnelle spécifique qui contient un certain nombre de paramètres réglables.

1.1.1 La régression linéaire

La régression linéaire simple

La régression linéaire simple est une partie importante dans la régression paramétrique définie comme suit :

Si on a un échantillon aléatoire (X_i, Y_i), $i = 1, ..., n$. Un modèle de régression linéaire simple suppose la relation affine entre X_i et Y_i suivante :

$$Y_i = aX_i + b + \varepsilon_i \tag{1.2}$$

Les ε_i sont indépendants de distribution normale : $\varepsilon_i \backsim N(0,s^2)$.

La régression linéaire consiste à déterminer des estimateurs des valeurs a et b. pour quantifier la validité de cette relation, nous utilisons le coefficient de corrélation linéaire.

Empiriquement, à partir de l'observation (y_i, x_i) $i = 1, ..., n$, on a représenté un graphe dont l'ensemble de points représentent des mesures d'une grandeur y_i en fonction d'une autre x_i ; par exemple la taille y_i des enfants en fonction de leur âge x_i (voir figure1.1).

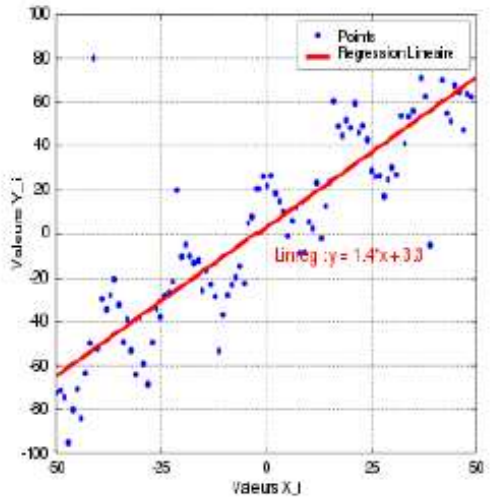

FIG. 1.1 – Un exemple graphique de régression linéaire simple

Les points paraissent alignés .On peut alors proposer un modèle linéaire, c'est-à-dire chercher la droite dont l'équation est $y_i = ax_i + b$ et qui passe

aussi par des points du graphe selon la méthode des moindres carrés et qui rend minimale la somme des carrés des écarts des points à la droite S donnée par :

$$S = \sum_{i\ \cdot 1}^{n} (y_i - ax_i - b)^2 \tag{1.3}$$

Où S représente le carré de la distance verticale des points expérimentaux $(x_i\ , y_{i)}$ à la droite considérée comme la meilleure. Cela revient donc à déterminer les valeurs des paramètres a et b (la pente de la droite et son ordonnée à l'origine) qui minimisent la somme S.

La notation vectorielle

Dans l'espace IR^n, muni de produit scolaire canonique, on considère le vecteur X de cordonnées $(x_1, x_2, ..., x_n)$; le vecteur Y de cordonnées $(y_1, y_2, ...\ , y_n)$;et le vecteur U de cordonnées (1,1,.... ,1).

On note alors $\overline{X}$ le vecteur $\overline{x}_i$, $\overline{Y}$ le vecteur $\overline{y}_i$ et le vecteur Z de cordonnées $(ax_1 + b, ax_2 + b, ..., ax_n + b)$ appartient à l'espace vectoriel engendré par X et U.

La somme$\sum (y_{i-} ax_i - b)^2$ représente le carré de la norme du vecteur $Y - Z$. Cette norme est minimale si et seulement si Z est le projeté orthogonale de Y alors :

$$(Z - Y)\, U = 0 \tag{1.4}$$

et

$$(Z - Y)(X - \overline{X}) = 0 \tag{1.5}$$

La généralisation à p variables explicatives de ce modèle s'appelle la régression linéaire multiple.

La régression linéaire multiple

Pour faire une régression linéaire multiple, il faut avoir n observations de Y_i $(i = 1, 2, ..., n)$ variables expliquées, et pour chaque Y_i on a p variables explicatives X_j où $(j = 1, 2, ..., p)$ et la relation entre Y_i et $X_{i1}, X_{i2}, ..., X_{ip}$ est formulé comme modèle linéaire.

$$Y_i = a_0 + a_1 X_{i1} + a_2 X_{i2} + ... + a_p X_{ip} + \varepsilon_i \tag{1.6}$$

Ou les constantes $a_0, a_1, ..., a_p$ sont appelées les coefficients partiels de régression .sur chaque a_j il s'agit de l'accroissement de y_i correspondant à

l'accroissement d'une unité x_{ij} quand toutes les autres variables sont maintenues constantes[1].

Les erreurs suivent une loi normale multidimensionnelle $\varepsilon \backsim N(0, \sigma^2 I_n)$

Pour estimer ces paramètres, nous utilisons comme dans ce qui précéde la méthode de moindre carré qui consiste à minimiser la somme des carrés des résidus S

$$S = \sum (y_i - a_1 x_{i1} - a_2 x_{i2} - .... - a_p x_{ip})^2 \tag{1.7}$$

(voir figure 1.2)

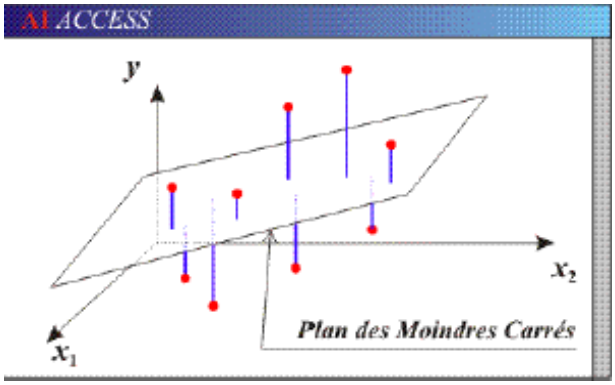

FIG. 1.2 – Un plan des moindres carrés

Généralisation (le cas matriciel)

Lorsqu'on dispose de plusieurs variables explicatives dans une régression linéaire, il est souhaitable d'avoir recours aux notations matricielles.

Si l'on dispose d'un jeu de n données $y_i, \quad i = 1, 2, ..., n$ que l'on souhaite expliquer par p variables explicatives $\quad (1, x_{1i}, x_{2i}, ..., x_{p-1i}) \quad , i = 1, 2, ..., n$ on peut poser :

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix} \tag{1.8}$$

et

$$X = \begin{bmatrix} 1 \;\; x_{11}....x_{p-11} \\ 1 \; x_{12}....x_{p-12} \\ .... \\ .... \\ 1 \; x_{1n}....x_{p-1n} \end{bmatrix} \tag{1.9}$$

La régression multiple linéaire s'exprime sous la forme matricielle suivante :

$$Y = XB$$

Alors, la question est d'estimer le vecteur de coefficient $B \in M_{P,1}$ employant la méthode de moindre carrés. On note l'estimateur par $\widehat{B}$ qui est donnée par :

$$\widehat{B} = (X^T X)^{-1} X^T Y \tag{1.10}$$

La lettre T dénote le transposé de la matrice.

S'il n'y a pas une colinéarité entre les variables explicatives, alors la matrice $X^T X$ est régulière $(\det (X^T X) \neq 0)$ et $(X^T X)^{-1}$ existe (c'est équivalent à rang (X) = rang $(X^T X)$ = p + 1).

Les limitations de la régression linéaire multiple 1/Complexité des calculs

Bien que similaires à ceux de la régression linéaire simple, les calculs et les résultats sont plus complexes en raison de la présence de plusieurs variables explicatives. Les calculs reposant sur des équations "ordinaires" deviennent lourdes à manipuler, et le recours à des équations matricielles devient alors une nécessité pratique.

2/Sélection des variables

Ce deuxième point est plus important pour le praticien : la régression linéaire multiple est probablement le premier exemple qu'il rencontrera de « compromis biais-variance » et dont nous résumons maintenant les aspects principaux :

∘Un modèle (qu'il soit prédictif ou descriptif) construit sur la base d'un échantillon ne doit pas contenir " trop " de paramètres (ici $\widehat{a}_0, \widehat{a}_1, ..., \widehat{a}_p$).

∘Ses performances s'améliorent sur l'échantillon, mais se dégradent sur des données nouvelles en raison d'une variance exagérée des estimations des valeurs prédites.

∘Les valeurs de ses paramètres prend toute signification, là encore en raison de leur grande variance.

∘Ces questions sont d'une grande important pratique et vont forcer l'analyste à fournir un effort important pour sélectionner, parmi les $\{x_i\}$ disponibles (souvent très nombreuses), celles qui seront retenues dans le modèle final.

3/ Colinéarité des variables explicatives :

S'il y a une colinéarité entre les variables explicatives, X^TX n' est pas régulière (det $(X^TX) = 0$) alors $(X^TX)^{-1}$ n' existe plus ,dans ce cas, nous ne pouvons pas déterminer l'estimateur du vecteur de coefficient B par la méthode de moindre carré ($\widehat{B}$ n' existe pas), alors la régression multiple est encore plus complexe[24].

Remarque 1 *Si les hypothèses initiales sont respectées, les estimateurs des MCO (Moindres Carrés Ordinaires) possèderont d' excellentes propriétés (sans biais, convergent en probabilité).*

1.1.2 La régression non linéaire

La régression non linéaire est une prolongation de la régression linéaire pour adopter des fonctions (non linéaires) générales de la forme :

$$y = f(x_1, x_2, ..., x_n, a_1, a_2, ..., a_n) \tag{1.11}$$

Voici des exemples des fonctions qui peuvent être modelées en utilisant la régression non linéaire :

$$y = a_0 + a_1 \exp(x_1) \tag{1.12}$$

$$y = a_0 + a_1 \sin(x_1) \tag{1.13}$$

$$y = a_0 \left[1 - \exp(-a_1 x_1)\right] \tag{1.14}$$

$$y_i = x_{1i}^{\lambda}(a_0 + a_1 x_{2i}) \tag{1.15}$$

$$y_i = a_0 x_{1i}^{a_1} \tag{1.16}$$

On peut estimer les différentes paramètres a_0 et a_1 de chaque modèle précédent avec des techniques differentes[24].

Remarque 2 *La forme non linéaire est non linéaire dans les paramètres, par exemple, le polynôme est non linéaire par rapport aux variables explicatives mais il est linéaire dans les paramètres donnés par :*

$$y = a_0 + a_1 {x_1}^2 + a_2 x_2^2 + ... + a_n {x_n}^2 \tag{1.17}$$

On peut poser une autre variable $z_i = {x_i}^2$ alors l'équation (1.17) devient

$$y = a0 + a1z1 + a2z2 + ... + a_n z_n$$

L'équation précédente a une forme linéaire.

Les problèmes de la régression non linéaire

1°/ Quand nous utilisons l'analyse de la régression non linéaire, la forme (modèle) de la fonction doit être indiquée. Pour la technologie et les problèmes scientifiques, le modèle peut être dicté par la théorie, mais pour les autres problèmes tels que les problèmes médicaux, il peut être difficile de développer des modèles non linéaires appropriés.

2°/ Les problèmes strictement numériques sont obnubilés par la nécessité d'estimer des paramètres avec des méthodes peu classiques, par exemple le

calcul de dérivé de S par rapport a a_0 et a_1 puis le fait de résoudre le système à deux équations qui en résulte, n'est pas facile.

3°/ Quelque fois un ajustement polynômiale peut être utilisé, mais il peut suffire d'éstimer la valeur d'un point intermédiaire de la courbe pour lequel aucune valeur ne peut être faite. Souvent, il faudra choisir un degré élevé pour le polynôme, alors qu'un modèle non linéaire fera intervenir un nombre plus faible de paramètres pour une qualité équivalente de l'ajustement. Ici la forme analytique plus complexe du modèle non linéaire compense, en quelque sort, le modèle plus restreint de paramètres.

4°/ La pratique peut être acceptable avec deux paramètres, mais elle deviendrait inutilisable des que ce nombre de parametres augmente.

1.2 La régression non paramétrique

Définition 3

la régression non paramétrique est une forme d'analyse de régression ou le prédicteur ne prend pas une forme prédéterminée , mais il est construit selon les informations dérivées à partir de la forme des données. Alors les variables expliquées sont présentées par une fonction des variables explicatives de forme non paramétrique comme suit :

$$Y_i = g(X_i) \qquad i = 1, 2, ..., n \tag{1.18}$$

Où g est une fonction non paramétrique .

La régression non paramétrique exige une plus grande taille de l'échantillon qu'une régression basée sur les modèles paramétriques parce que les données doivent bien rendre la structure du modèle estimé.

Nous avons vu que la régression doit également faire face, comme toute modélisation, à la très importante question du choix des variables à incorporer dans le modèle. Nous ne rappellerons jamais assez que l'ajout des variables augmente la quantité d'information disponible pour prédire les données et augmenter le nombre de paramètres du modèle (donc sa souplesse), Ce qui lui permet de mieux rendre compte des données disponibles. Mais que le prix à payer pour ces capacités accrues est d'une plus grande instabilité du modèle, Il existe donc un "juste milieu", difficile à trouver (sauf dans les cas les plus simples comme la régression linéaire simple). En plus, il y a deux difficultés propres à la régression, la première est que la fonction de régression $f(.)$ peut avoir une forme analytique quelconque, et de toute façon inconnue. L'analyste devra donc faire le choix de la forme fonctionnelle du modèle de régression même si la distribution des données s'écarte de la linéarité. Et la

deuxième difficulté est que ε_x est une variable aléatoire de moyenne nulle, mais de distribution inconnue, et dépendant possiblement de x (hétéroscédasticité). Le but de ce travail est de rendre ces difficultés minimales en utilisant les réseaux de neurone. Dans la régression, y (la variable expliquée) est une variable continue. On peut concentrer les discussions à un cas particulier où y a des valeurs discrètes (par exemple 0, 1,... etc). Cette méthode statistique s'appelle l'analyse discriminante. Dans le deuxième chapitre, nous allons définir l'analyse discriminante et quelque principes qui permettent d'utiliser les réseaux de neurones .

Chapitre 2

L'analyse discriminante

Dans le premier chapitre, nous avons fait un rappel de l'une des questions centrales de la modélisation statistique de données qui est la régression. Nous avons vu que cette méthode a plusieurs limitations. Pour faire une discussion à but d'amélioration, nous proposons le cas particulier ou la variable expliqué est discrète (qualitative). Nous appelons ce cas l'analyse discriminante.

Dans ce chapitre, nous allons définir les notions et discuter les propriétés de l'analyse discriminante et sa robustesse en utilisant l'exemple suivant :

"Lorsque y n'est pas une variable quantitative mais qualitative á deux niveaux, il est encore possible de faire « formellement » une régression, Par exemple, dans le domaine médical, on peut étudier les soins possibles pour une maladie : intervention chirurgicale(groupe 1, $y = 0$) , ou chimiothérapie (groupe 2, $y = 1$) en fonction de certaines données biologiques représentant les régresseurs. La régression linéaire multiple fournit les mêmes résultats que l'analyse discriminante linéaire á deux populations. Si dans les n observations n_1 valeurs de y sont égales á 0 et n_2 égales á 1,la moyenne $\overline{y} = n_2/n, (n = n_1 + n_2)$. Si on fait une régression sur le y centré, les observations correspondants á une intervention chirurgicale sont toutes négatives (et égales á $-\ n_1/n$) celles correspondant á la chimiothérapie sont toutes positives (et égale á n_2/n). L'estimation de y permet de choisir le type de soins auquel il faut soumettre un nouveau malade sur lequel ont été observées les mêmes données biologiques : intervention chirurgicale si l´estimation de y est négative, chimiothérapie si elle est positive. Cette estimation est un outil d'aide á la discussion pour le praticien, d'autant plus sur que la valeur est différente de 0"[25].

2.1 La discrimination

D'après l'exemple précédent, nous pouvons représenter les résultats de discrimination en termes de variable y, où

$$\begin{cases} y = 1 \text{ si le malade } \in C_1 \\ y = 0 \text{ si non il} \in C_2 \end{cases} \tag{2.1}$$

ou nous pouvons utiliser un seuil S,dans l'exemple précédent $S = 0$ telque

$$\begin{cases} \overline{y} \geqslant 0 \text{ le malade } \in C_1 \\ \overline{y} \lessdot 0 \text{ le malade } \in C_2 \end{cases} \tag{2.2}$$

Dans les problèmes plus complexe, il peut y avoir plusieurs classes (C), pas seulement C_1 et C_2, on associe les classes avec plusieurs variables y_k où $k = 1, ...c$

On peut rassembler les variables x_1 et x_2 dans un vecteur de dimension 2, x tel que $x = (x_1, x_2)$, meme pour les variables y_k telle que $y = (y_1, ..., y_c)$ de dimension c.

Alors nous pouvons modeler cette opération par une fonction mathématique qui contient un certain nombre des paramètres réglables w, dont les valeurs sont déterminées avec l'aide des données : $y_k = y_k(x, w)$ telles que $w = (w_1, ..., w_d)^T$.

2.2 Le théorème de Bayes et la classification

Nous revenons à l'exemple donné dans l'introduction de ce chapitre. Nous supposons que nous souhaitons classifier un nouveau malade, mais jusqu'ici nous n'avons fait aucune mesure sur lui. Le but est de classifier ce malade de façon de réduire au minimum la probabilité de classification fausse.

Nous avons rassemblé un grand nombre d'exemples des malades pour déterminer la probabilité a priori $p(c_k)$ quun malade appartient à chacune des classes c_1 ou c_2.

On suppose qu'on a

$$p(c_1) = 0.75 \text{ et } p(c_2) = 0.25 \tag{2.3}$$

alors nous assignons le malade précédent à la classe qui a la probabilité a priori plus élevé, (c.à.d. le malade appartient à c_1 si $p(c_1) \succ p(c_2)$, et à la classe c_2 autrement). Et ceci signifie que nous classifierons toujours un

nouveau malade qui correspondra au groupe des malades de l'intervention chirurgicale.

Nous pouvons utiliser le théorème de Bayes pour réaliser notre but, mais présentons d'abord la formulation générale de ce théorème.

Définition 4 *Soit A et B deux évènements quelconques d'un ensemble fondamental Ω muni d'une loi de probabilité P. La probabilité conditionnelle de A sachant que l'évènement B est réalisé, est notée par $P(A/B)$ et définie par la relation suivante*

$$P(A/B) = \frac{P(A \cap B)}{P(B)}$$

Théorème de Bayes

Aussi connu sous le nom de "Règle de Bayes" ou "Formule de Bayes"

Théorème 5 *Théorème de Bayes*

La formule simple

Soit A et B deux évènements quelconques d'un ensemble fondamental Ω muni d'une loi de probabilité P.La formule de Bayes est donnée par

$$P(B/A) = \frac{P(A/B).P(B)}{P(A)} \tag{2.4}$$

La formule générale

Elle s'énonce ainsi :

* Si A est un évènement quelconque,

* Et si $\{B_1, B_2, ..., B_n\}$ est une partition de Ω

Alors pour tout i :

$$P(B_i/A) = \frac{P(A/B_i).P(B_i)}{\sum_{j=1}^{n} P(A/B_j) P(B_j)} \tag{2.5}$$

Présentons le théorème de Bayes dans le contexte de la classification :

Le cas discret

Supposons qu'il faille affecter des observations à des classes sur la base de mesures faites sur l'unique attribut x, supposé prendre des valeurs discrètes $x^{(l)}$.

►Nous définissons *la probabilité a priori* de la classe c_k comme la proportion des observations qui appartiennent à cette classe, quelle que soit la valeur prise par x. Nous notons cette probabilité $P(c_k)$.

►De même, nous définissons la *probabilité inconditionnelle* $P\left(x^{(l)}\right)$comme la proportion des observations pour lesquelles x prend la valeur $x^{(l)}$, quelle que soit leur classe.

► De même, *la probabilité conditionnelle* de classe $P\left(x^{(l)}/c_k\right)$ est la proportion des observations dans la classe c_k pour lesquelles $x = x^{(l)}$. Pour une classe donnée, c'est une fonction de l.

►Nous ne considérons maintenant que les observations pour lesquelles $x = x^{(l)}$. *La probabilité a posteriori* de la classe c_k pour la valeur $x^{(l)}$ est la proportion des observations qui appartiennent à cette classe. Elle est notée $P\left(c_k/x^{(l)}\right)$,pour une valeur donnée, c'est donc une fonction de k.

$P(c_k)$ ést la probabilité pour qu'une observation, avant que la mesure sur x soit effectuée, appartienne à la classe k. Mais cette mesure a apporté de l'information supplémentaire, ce qui nous permet d'affiner nos probabilités de $P(c_k)$ à $P\left(c_k/x^{(l)}\right)$. Ceci explique les termes "*a priori*" et "*a posteriori*".

Le théorème de Bayes permet de relier ces quatre quantités. Il s'exprime par la formule :

$$P\left(c_k/x^{(l)}\right) = \frac{P\left(x^{(l)}/c_k\right) P(c_k)}{P\left(x^{(l)}\right)} \tag{2.6}$$

Donc ce que dit le théorème de Bayes est :

" Vous voulez savoir à quelle classe appartient une nouvelle observation en mesurant x ? Vous ne le saurez jamais avec certitude, mais, pour chacune des classes c_k, vous pouvez améliorer votre estimation initiale $P(c_k)$ de la probabilité d'appartenance à la classe en la transformant en $P\left(c_k/x^{(l)}\right)$, qui tient compte du résultat de la mesure.

Le dénominateur $P\left(x^{(l)}\right)$ joue le rôle d'un facteur de normalisation. Pour s'en convaincre, on additionne toutes les probabilités a posteriori pour obtenir :

$$\sum_j P\left(c_k/x^{(l)}\right) = 1 \tag{2.7}$$

En effet, une observation appartient forcément à une classe et à une seule, et la somme des probabilités d'appartenance aux classes est donc égale à 1.

Remplaçons chaque terme de la somme par son expression donnée par la formule de Bayes. Nous obtenons :

$$P\left(x^{(l)}\right) = \sum_j P\left(x^{(l)}/c_j\right) P\left(c_j\right) \tag{2.8}$$

La formule de Bayes s'écrit maintenant :

$$P\left(c_k/x^{(l)}\right) = \frac{P\left(x^{(l)}/c_k\right) P\left(c_k\right)}{\sum_j P\left(x^{(l)}/c_j\right) P\left(c_j\right)} \tag{2.9}$$

Où le dénominateur apparaît clairement comme un facteur de normalisation.

Le cas continu

La formule de Bayes n'est que légèrement modifiée si x est une variable continue. La définition des probabilités a priori est bien sûr inchangées. Mais :

* Les probabilités inconditionnelles $P\left(x^{(l)}\right)$ doivent être remplacées par la densité (de probabilité) inconditionnelle $f\left(x\right)$, qui est la fonction de densité de probabilité de x quand les étiquettes de classe ne sont pas prises en compte.

* Les probabilités conditionnelles de classe $P\left(x^{(l)}/c_k\right)$doivent être remplacées par les densités (de probabilité) conditionnelles de classes $f\left(x/c_k\right)$. Chacune de ces k densités est la densité de probabilité de x pour la population d'une classe quand les autres classes sont ignorées.

La formule de Bayes devient alors :

$$f\left(c_k/x\right) = \frac{f\left(x/c_k\right) P\left(c_k\right)}{f\left(x\right)} \tag{2.10}$$

Il est important de distinguer deux étapes séparées dans le procédé de la classification, elles sont : l'inférence et la décision

∘ *L'inférence*

L'inférence des données est employée pour déterminer des valeurs ($P\left(c_k\right)$, $P\left(x^{(l)}/c_k\right)$,$P\left(x^{(l)}\right)$) afin de calculer la probabilité a posteriori pour chaque classe comme suit :

* Prenez la probabilité a priori $P\left(c_k\right)$de la classe,

* Multipliez cette probabilité par $P\left(x^{(l)}/c_k\right)$, la probabilité conditionnelle de classe pour la valeur $x^{(l)}$ de l'attribut,

* Divisez le résultat par $P\left(x^{(l)}\right)$, la probabilité inconditionnelle de la valeur de l'attribut. Ce nombre est le même pour toutes les classes, et ne dépend que de l.

◦ *La décision*

Dans ce cas, ces probabilités sont employées pour faire des décisions qu'un nouveau point de donnée appartient à la classe pour laquelle la probabilité a posteriori est la plus grande.

2.3 La frontière de décision

Jusqu'ici, dans une perspective de minimiser la probabilité de classification fausse, nous avons vu que la théorie de la décision bayésienne est une stratégie qui consiste à affecter toute observation à la classe ayant la plus grande probabilité a posteriori, alors x sera affecté à la classe k si

$$p(c_k/x) > p(c_j/x) \text{ pour tout } \quad j \neq k \tag{2.11}$$

Nous remarquons que l'espace de donnée étant divisé en k classes disjointes. Les frontières entre ces régions sont appelées *les frontières de décision.*

Si l'espace des données est un espace multidimensionnel, ces frontières sont des hyperplans. Voir figure (2.1).

Dans la figure (2.1), l'espace des données est de trois dimensions alors la frontière de décision est un plan qui discrimine la classe c_1 et la classe c_2.

2.4 La fonction discriminante

On peut reformuler le traitement de classification en terme des fonctions $y_1(x), ...y_c(x)$ où, pour tout $k = 1, 2, ..., c$:

$$y_k(x) = p(c_k/x) \tag{2.12}$$

D'après les expressions (2.11) et (2.12), le vecteur de données x est affecté à c_k si

$$y_k(x) > y_j(x) \text{ pour tout } \quad j \neq k \tag{2.13}$$

x est affecté à la classe c_k si

$$\frac{P\left(x/c_k\right) P\left(c_k\right)}{P\left(x\right)} \succ \frac{P\left(x/c_j\right) P\left(c_j\right)}{P\left(x\right)} \text{ pour tout } \quad j \neq k \tag{2.14}$$

Donc

$$P\left(x/c_k\right) P\left(c_k\right) \succ P\left(x/c_j\right) P\left(c_j\right) \quad \text{pour tout } \quad j \neq k \tag{2.15}$$

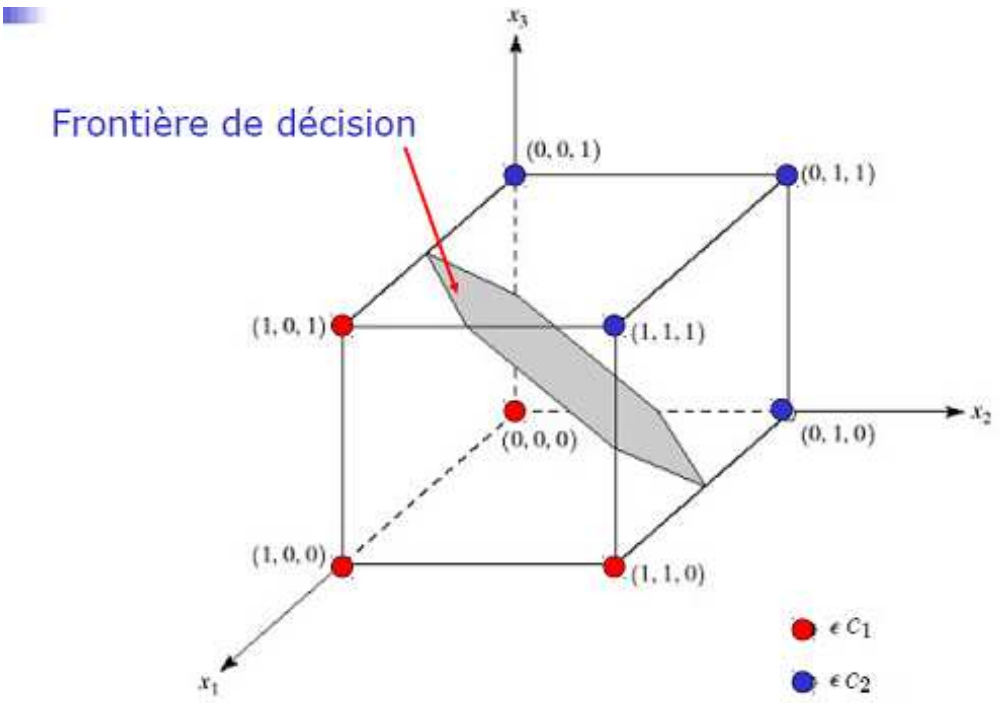

FIG. 2.1 – Un exemple du frontière de décision dans un espace trois dimensionnelle[12]

D'après (2.15) ,on peut définir $y_k(x)$ par la formule suivante :

$$y_k(x) = P\left(x/c_k\right) P\left(c_k\right) \tag{2.16}$$

Les $y_k(x)$ sont appelées *les fonctions discriminantes*

Remarque 6 *Plusieurs fonctions discriminantes peuvent être définies, et la plus utilisée est donnée par $g(y_k(x))$, où, $g(.)$ est une fonction monotone croissante et $y_k(x)$ est une autre fonction discriminante [10].*

Remarque 7 *On a remarqué que les frontières de décision sont définies par l'ensemble des équations suivantes*

$$y_k(x) = y_j(x) \quad pour\ tout \quad j \neq k \tag{2.17}$$

Remarque 8 *Lorsqu'il n'y a que deux classes c_1 et c_2, on peut donner la fonction discriminante sous une forme légèrement différente.On note*

$$y(x) = y_1(x) - y_2(x)$$

Alors

$$\begin{cases} si\ y(x) \geq 0\ alors & x \in c_1 \\ siy(x) < 0\ alors & x \in c_2 \end{cases} \tag{2.18}$$

L'avantage de cette notation est d'employer une seule fonction discriminante au lieu d'employer deux fonctions $y_1(x)$ et $y_2(x)$.

2.4.1 Les types de fonctions discriminantes

Nous avons annoncé précédemment qu'il y a une étape inférencielle dans le procédé de classification. Cet aspect descriptif vise à trouver une représentation qui permet d'interpréter les groupes grâce aux variables explicatives. Cette tâche est rendue difficile quand le nombre de variables explicatives est plus grand que trois. Il est possible d'utiliser la technique de l'analyse discriminante qui est la plus populaire des techniques de classification.

L'analyse discriminante est une technique paramétrique car elle fait l'hypothèse que les densités des classes ont une forme fonctionnelle particulière. Plus précisément, elle suppose que les classes sont multinormales. Cette hypothèse très forte permet alors de déduire des estimations précises de probabilité a posteriori de chacune des classes.“ L'utilisation de cette hypothèse dans le cadre de l'analyse a d'ailleurs donné naissance aux deux méthodes de discrimination les plus populaires : L'analyse discriminante linéaire et l'analyse discriminante quadratique. Et pour cela, il existe deux types de fonctions discriminantes : l'une est linéaire et l'autre est quadratique"[7].

La fonction discriminante quadratique

Dans l'analyse discriminante, nous supposons que les classes sont de distribution multinormale. Si l'espace est d-dimensionnelle, la forme de la fonction de densité de probabilité multinormale donnée par

$$f(x/c_k) = \frac{1}{(2\pi)^{\frac{d}{2}} |D|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T D^{-1}(x-\mu_k)\right\} \tag{2.19}$$

Où la moyenne μ est un vecteur de d-dimension, D la matrice de covariance de dimension $d \times d$, et $|D|$ le déterminant de D.

D'après la remarque (6), nous supposons que la fonction discriminante a une forme particulière $g(f(x))$ où $g(x) = \ln x$ et $f(x)$ une fonction discriminante représentée dans (2.16). Alors

$$y_k(x) = \ln f\left(x/c_k\right) + \ln P\left(c_k\right) \tag{2.20}$$

Nous substituons (2.19)dans (2.20) et nous omettons les termes constants, l'équation (2.20) devient :

$$y_k(x) = -\frac{1}{2}(x-\mu_k)^T D_k^{-1}(x-\mu_k) - \frac{1}{2}\ln|D_k| + \ln P\left(c_k\right) \tag{2.21}$$

La forme donnée dans l'équation (2.21) est la forme générale d'une fonction discriminante quadratique dans l'espace d-dimension. Pour cela, la frontière de décision est quadratique.

La fonction discriminante linéaire

La fonction discriminante linéaire est une régularisation d'une fonction discriminante quadratique car nous avons ajouté une hypothèse supplémentaire d'égalité des matrices de covariance (c-à-d : $\forall\ k = 1, 2, ..., c\ : D_k = D$) Alors

$$y_k(x) = -\frac{1}{2}(x-\mu_k)^T D^{-1}(x-\mu_k) + \ln P\left(c_k\right) \tag{2.22}$$

Où nous avons omis le terme $\ln|D|$ et le terme $x^T D^{-1} x$ car ils ne dépendent pas de la classe k car D est une matrice symétrique, alors $x^T D^{-1}\mu_k = \mu_k^T D^{-1} x$.

On peut écrire l'équation (2.22) sous la forme suivante

$$y_k(x) = w_k^t x + w_{k0} \tag{2.23}$$

Où

$$w_k^t = \mu_k^T D^{-1} \tag{2.24}$$

Et

$$w_{k0} = -\frac{1}{2}\mu_k^T D^{-1}\mu_k + \ln P(c_k) \tag{2.25}$$

Nous remarquons que l'équation (2.23) est linéaire en x, alors $y_k(x)$ est *une fonction discriminante linéaire*[18]. Les frontières de décision correspondantes à l'équation $y_k(x) = y_j(x)$ sont alors un hyperplan. Voir figure (2.2).

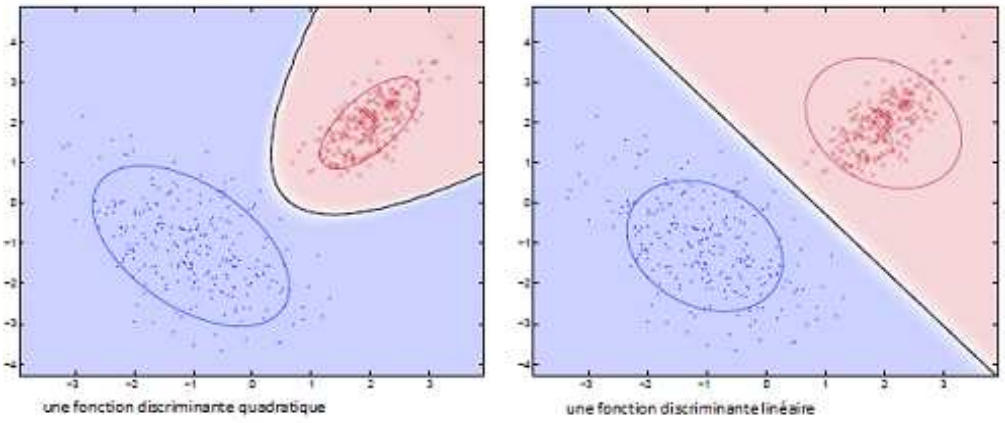

FIG. 2.2 – Frontières de décision de l'Analyse Discriminante Quadratique et de l'Analyse Discriminante Linéaire sur un même jeu de données en dimension 2 [7].

2.5 La séparabilité linéaire

Jusqu'ici dans ce chapitre, nous avons discuté des fonctions discriminantes ayant une frontière de décision qui est linéaire ou plus généralement hyperplan de dimension plus élevée .

Considérons pour le moment le problème de classifier un ensemble de données exactement , où le vecteur des points a été marqué comme appartenant à l'une de deux classes c_1 et c_2. Si tous les points peuvent étre classifiés correctement par un frontière de décision linéaire (hyperplan dans des grandes dimensions) , alors les points seront " *linéairement séparables*"

Définition 9 *"Un ensemble de vecteurs de données, s'appellent linéairement séparables s'ils peuvent être séparée par un ensemble d'hyperplans comme frontières de décision dans l'espace de données[15].*

L'Analyse Discriminante est une méthode simple, bien comprise sur le plan théorique, et raisonnablement efficace sur la plupart des problèmes ordinaires. Elle perd son efficacité lorsque les distributions des classes s'écartent sensiblement de la normalité, et dans le monde réel, les classes ne sont jamais parfaitement multinormales.

Une autre faiblesse de l'Analyse Discriminante est que sa version complète nécessite l'estimation d'autant de matrices de covariance . Ceci conduit rapidement à des modèles contenant des dizaines, voire des centaines de paramètres, un nombre important au regard des volumes limités de données habituellement disponibles.

En conséquence, l'Analyse Discriminante complète tend à être instable (modèle fortement dépendant des données), alors que ses versions "restreintes", avec moins de paramètres, gagnent en stabilité mais au prix d'un biais accru. Il existe alors un recours qui est l'utilisation des réseaux de neurones. Dans le troisième chapitre, nous allons confirmer que *Les Réseaux de Neurones artificiels* font un pas de plus dans la généralisation en supprimant toute hypothèse sur les densités conditionnelles.

Chapitre 3

Les réseaux de neurone et la statistique conventionnelle

Les développements récents dans l´Intelligence Artificielle (I A) ont permis la construction des Systèmes Experts, en particulier dans le domaine de la régression et dans l'analyse des données statistiques.

Les Réseaux de Neurones Artificiel (RNA) sont les plus utilisés parmi ces systèmes. Ils sont des champs d'évaluation avec des origines en neurobiologie. A l'image de nos cerveaux qui peuvent exécuter les tâches les complexes, les réseaux de neurones modèles ont également été trouvés utiles en résolvant des problèmes complexes.

Une plus large définition d'un réseau de neurone pratique est une collection de neurones reliés apprennant incrementalement de leur environnement (données) pour capturer les tendances linéaires et non linéaires essentielles dans les données multidimensionnelles , de sorte qu'il fournisse des prédictions fiables pour une nouvelle situation de même information bruyante et partielle. Les neurones sont les unités de bases de calcul qui exécutent le traitement local de données à l'intérieure d'un réseau. Ces neurones forment en parallèle les réseaux, dont la fonction de sortie est déterminée par la structure de réseau, et les forces de raccordement entre les neurones parlesquels le traitement est exécuté par une fonction d'activation. . Un réseau de neurones ressemble au cerveau à deux égards

1. La connaissance est acquise par un apprentissage.
2. Les forces d'intercommunication entre les neurones, connues sous le nom de poids synaptique où les poids sont employés pour stoker la connaissance" [14].

Comme déjà indiqué précédemment, les réseaux de neurones exécutent une variété de tâches, y compris la prédiction (régression) ou l'approximation de fonction, et la classification de modèles comme il est représenté sur le

schéma [3.1].

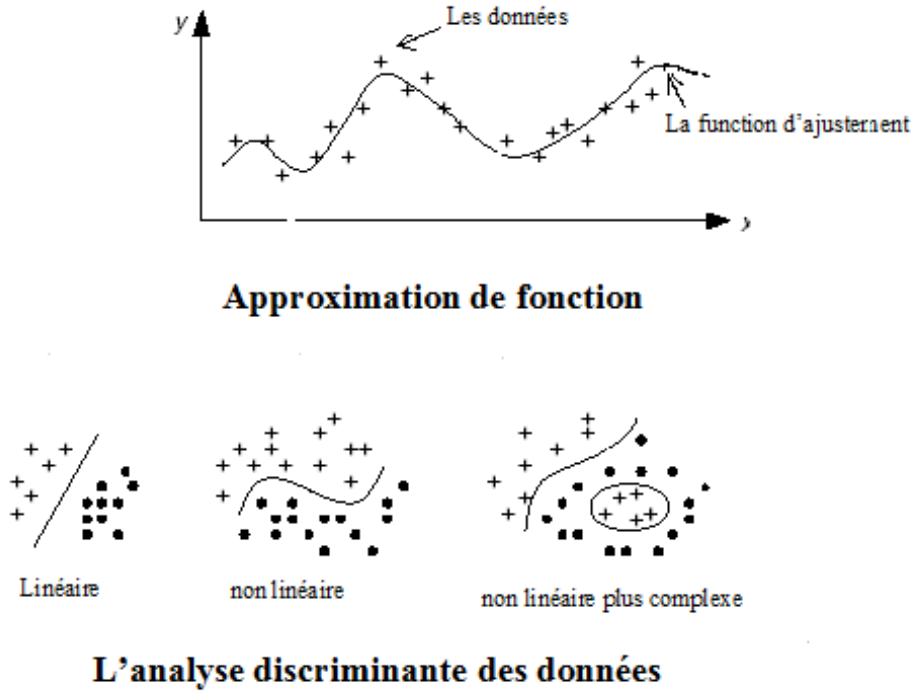

FIG. 3.1 – Approximation de fonction et l'analyse discriminante par un réseau de neurone

Les réseaux de neurones sont très puissants en adaptant les modèles aux données. Ils peuvent adapter arbitrairement les modèles non linéaires complexes aux données multidimensionnelles à toute exactitude désirée. D'un point de vue fonctionnelle, ils peuvent être considérés comme un prolongement de certaines techniques multivariable, telle que la régression linéaire multiple, la régression non linéaire, les tâches de classification comportant arbitrairement les frontières de décision non linéaires complexes. Nous pouvons discuter ces problèmes dans ce chapitre.

D'après la définition de réseaux de neurones, toutes ses différentes capacités varient en fonction de la fonction d'activation, la structure de réseau et le type d'apprentissage utilisé :

3.1 La fonction d'activation

Définition biologique

La fonction d'activation est une abstraction représentant le taux de potentiel d'action mise à feu dans la cellule. Sous sa forme plus simple, cette fonction est binaire c'est-à-dire, l'un ou l'autre neurone est la mise à feu ou pas.

Définition artificiel

La fonction d'activation(ou fonction de seuillage, ou encore fonction de transfert) d'un neurone artificiel définit le rendement de ce neurone donné à partir d'une entrée ou un ensemble d'entrées."Différentes fonctions de transfert pouvant être utilisées comme fonction d'activation du neurone sont énumérées dansla figure (3.2). Les trois les plus utilisées sont les fonctions «*seuil*» (en anglais «hard limit»), «*linéaire*» et «*sigmoïde*»."[21]

Nom de la fonction	Relation d'entrée/sortie	Icône	Nom Matlab
seuil	$a = 0$ si $n < 0$ $a = 1$ si $n \geq 0$		hardlim
seuil symétrique	$a = -1$ si $n < 0$ $a = 1$ si $n \geq 0$		hardlims
linéaire	$a = n$		purelin
linéaire saturée	$a = 0$ si $n < 0$ $a = n$ si $0 \leq n \leq 1$ $a = 1$ si $n > 1$		satlin
linéaire saturée symétrique	$a = -1$ si $n < -1$ $a = n$ si $-1 \leq n \leq 1$ $a = 1$ si $n > 1$		satlins
linéaire positive	$a = 0$ si $n < 0$ $a = n$ si $n \geq 0$		poslin
sigmoïde	$a = \frac{1}{1+\exp^{-n}}$		logsig
tangente hyperbolique	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
compétitive	$a = 1$ si n maximum $a = 0$ autrement		compet

FIG. 3.2 – Les fonctions d'activation $a = f(n)$ [21]

Dorénavant, nous remplaçons la terminologie statistique conventionnelle par une terminologie qui est utilisée dans les réseaux de neurone, tel que l'échantillon (le vecteur de variables $(x_1, x_2, ..., x_d)$ est remplacé par les entrés de neurone, la fonction de transfert non linéaire g correspond à la fonction d'activation dans le neurone qui transfère le mélange linéaire des entrées et les poids à une valeur de sortie qui est une fonction des entrées .

3.2 Réseau de Neurone Artificiel (RNA)

3.2.1 Le neurone formel

Définition 10 *Le neurone formel est conçu comme un automate doté d'une fonction de transfert qui transforme ses entrées en sortie selon des règles précises.*

3.2.2 Les types des réseaux de neurones

Comme déjà indiqué dans la définition, on peut distinguer les réseaux de neurones d'après la structure de réseau.

Les réseaux de neurones bouclés (cyclique) ou (connexions récurrentes) :

Dans le réseau bouclé, il est possible de trouver au moins un chemin (connexion) qui revient à son point de départ (ramenant l'information en arrière) alors le réseau bouclé contient une ou plusieurs boucles de rétroaction.réseaux bouclés.

Les réseaux de neurone non bouclés :

Un réseau de neurone non bouclé réalise une (ou plusieurs) fonction de ces entrées, par composition des fonctions réalisées par chaque neurone. Les neurones sont rangés par couches, il n'y a pas de connexion entre les neurones d'une même couche, et les connexions ne sont faites qu' avec les neurones des couches avales (propagation avant).

Le diagramme de ces réseaux ne contient aucune boucle de rétroaction ; ceci assure que les sorties de réseau peuvent être calculées en tant que *fonctions explicatives* des entrées et des poids.

On peut distinguer dans cette catégorie deux structures différentes dépendant du nombre de couches dans le réseau.

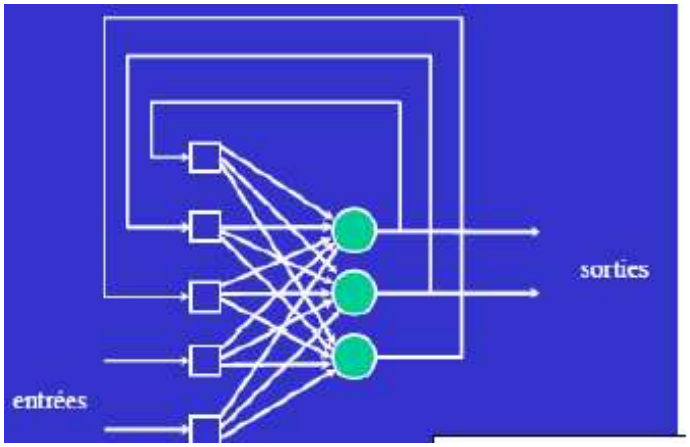

FIG. 3.3 – Un exemple de RNA cyclique [2].

Réseau a une seule couche : Dans ces réseaux, il y a plusieurs perceptrons rangés dans une seule couche. Un exemple de réseau avec une seule couche de trois perceptrons est schématisé dans la figure (3.4)

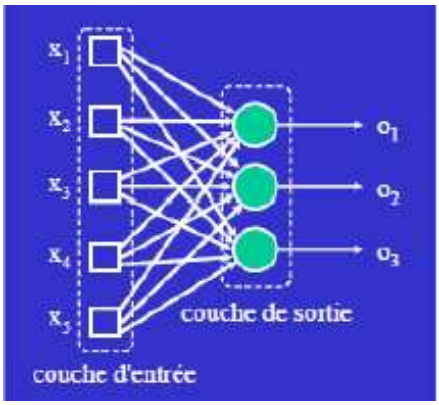

FIG. 3.4 – Un exemple de RNA avec une seule couche [2]

Remarque 11 *: seule couche indique la couche des sorties car dans la couche d' entrée il n'ya pas le traitement d'activation de la connexion (des fonctions d'activations).*

Réseau multicouche : Ce réseau a plusieurs couches de neurones (des poids adaptatifs). Habituellement, chaque neurone d'une couche est connecté

à tous les neurones de la couche suivante, et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (l'activation) au sein d'un réseau.

Les couches intermédiaires entre la couche d'entrée et la couche de sortie n'ayant aucun contact avec l'extérieur sont appelées couches cachées (dans l'exécution informatique). (voir figure3.5)

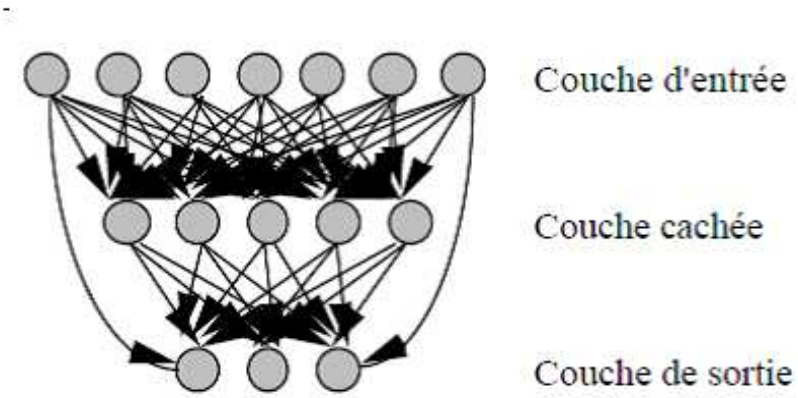

FIG. 3.5 – Un exemple d'un RNA multicouche [25]

Dans la suite, si nous discutons le réseau multicouche, nous illustrerons l'étude par le cas d'un réseau de deux couches (une seule couche cachée) car nous remuons des réseaux avec juste deux couches de poids adaptatif capables de rapprocher une fonction non linéaire continue.

On peut dire que n'importe quelle topologie (diagramme) de réseau, si elle est à propagation avant, peut être traduite en fonctions correspondantes. Alors, l'importance des réseaux de neurone est d'offrir un cadre très puissant et très général pour représenter les fonctions (linéaire ou non linéaire) de plusieurs variables d'entrées dans plusieurs variables de sortie, où les formes de ces fonctions sont régies par un certain nombre de paramètres réglables. Le processus qui fait l'ajustement des ces paramètres sur la base d'ensemble s'appelle l'apprentissage ; et pour cette raison l'ensemble de données s'appelle l'ensemble d'apprentissage.

3.3 L'apprentissage

L'apprentissage est vraisemblablement la popriété la plus intéressante des réseaux de neurones. Cependant, elle ne concerne pas tous les modèles, Mais les plus utilisés.

Définition 12 *l'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré.*

"C'est-à-dire un changement dans la valeur des poids qui relient les neurones d'une couche à l'autre"[7]. Soit le poids w_{ij} reliant le neurone i à son entrée j. Au temps τ, un changement $\triangle w_{ij}^{(\tau)}$ de poids peut s'exprimer simplement de la façon suivante :

$$\triangle w_{ij}^{(\tau)} = w_{ij}^{(\tau+1)} - w_{ij}^{(\tau)} \tag{3.1}$$

Par conséquent, $w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \triangle w_{ij}^{(\tau)}$, avec $w_{ij}^{(\tau+1)}$ et $w_{ij}^{(\tau)}$ représentent respectivement les nouvelles valeurs et les anciennes aussi du poids w_{ij}.

3.3.1 Les types d'apprentissage

Un ensemble fixé de règles bien définies pour la solution d'un problème d'apprentissage est appelé "un algorithme d'apprentissage ". Au niveau de ces algorithmes d'apprentissage ; il a été défini deux grands types d'apprentissage : supervisé et non supervisé.

L'apprentissage supervisé

Les données utilisées pour l'apprentissage supervisé sont dites "complètes" car elles contiennent à la fois les valeurs x_1, ..., x_c prises par les p variables explicatives et leur appartenance aux c classes t_1, ...,t_c. Les données complètes sont donc l'ensemble des couples (observation, cible), i.e. $\{(x_1, t_1)$, ..., $(x_c, t_c)\}$.On peut dire qu'" un professeur" fournit aux exemples de ce que celui-ci doit faire pour tenir compte de l'erreur observée en sortie.

L'apprentissage non supervisé

Les données utilisées pour l'apprentissage non supervisé ne sont pas "complètes" car elles ne contiennent que les valeurs x_1, ..., x_n prises par les p variables explicatives, alors il n'y a pas de " professeur".

3.3.2 Les règles d'apprentissage

Il y a différentes règles pouvant guider l'apprentissage d'un réseau de neurone

Par correction d'erreur (la règle delta)

Correction d'erreur désigne la correction de l'erreur observée en sortie. L'apprentissage par correction des erreurs consiste à minimiser un indice de performance E basé sur les signaux d'erreur $e_i^{(\tau)}$ qui est l'erreur entre ce qu'on obtient $y_k\,(x)$et ce qu'on voudrait obtenir t_k, dans le but de faire converger les sorties du réseau avec ce qu'on voudrait qu'elles soient. Un critère très usuel est la somme des erreurs quadratiques E (nous discuterons le choix de ce critère dans le chapitre quatre). Il faut changer les poids de réseau dans une direction qui diminue E, alors dans le sens opposé au gradient. On parle alors d'une direction de «descente» donnée par

$$\Delta w^{(\tau)} = -\eta \nabla E^{(\tau)} \tag{3.2}$$

Où η est appelée le taux d'apprentissage et $\nabla E^{(\tau)}$ désigne le gradient de E par rapport à ces paramètres libres (les poids w) au temps τ, la règle (3.2) dite de «*descente du gradient*». Nous ferons une discussion plus détaillée sur cette règle dans le cinquième chapitre.

Par la règle de Hebb

Dans cette section, nous abordons une règle qui s'inspire des travaux du neurophysiologiste Donald Hebb. Dans un contexte neurobiologique, Hebb cherchait à établir une forme d'un apprentissage associatif au niveau cellulaire. Dans le contexte des réseaux artificiels, on peut reformuler l'énoncé de Hebb sous la forme d'une règle d'apprentissage donnée par

$$\Delta w_j^{(\tau-1)} = \eta p_j^{(\tau)} a^{(\tau)} \tag{3.3}$$

Où η est une constante positive qui détermine la vitesse de l'apprentissage, $p_j^{(\tau)}$ correspond à l'activité pré-synaptique (l'entrée j du neurone) au temps τ, et $a^{(\tau)}$à l'activité post-synaptique (sortie du neurone) à ce même temps τ. Cette formule fait ressortir explicitement la corrélation entre le signal qui entre et celui qui sort.

La règle d'apprentissage compétitif

"Comme son nom l'indique, l'apprentissage compétitif consiste à faire compétitioner les neurones d'un réseau pour déterminer celui qui sera actif à un instant donné. Contrairement aux autres types d'apprentissage où, généralement, tous les neurones peuvent apprendre simultanément et de la même manière, l'apprentissage compétitif produit un «*vainqueur*» ainsi que, parfois, un ensemble de neurones «voisins» du vainqueur. Seul ce vainqueur et, potentiellement, son voisinage bénéficient d'une adaptation de leur poids. On dit alors que l'apprentissage est local car il est limité à un sous-ensemble des neurones du réseau"[21]

Donc, on peut écrire la règle d'apprentissage compétitif comme suit

$$\Delta w = \begin{cases} \eta(x-w) & \text{si le neurone est vainqueur} \\ 0 & \text{autrement} \end{cases} \tag{3.4}$$

Où $0 < \eta < 1$ correspond à un taux d'apprentissage, x est le vecteur d'entrée et w est le vecteur de poids.

Dans les paragraphes suivants de ce chapitre , nous insisterons sur les réseaux de neurones qui traitent des tâches de régression (approximation de fonction) et l'analyse discriminante statistique. On distingue trois catégories d'après les fonctions d'activation qui sont utilisées dans les neurones (fonction seuil , linéaire , et non linéaire)

3.4 Les réseaux de neurones à seuil

On commence par un seul neurone simple avec une ou plusieurs entrées présentées sur la figure [3.6] qui s'appelle le perceptron

3.4.1 Le perceptron

Le perceptron est historiquement le premier modèle. Il est développé comme approximation simple des neurones biologiques par Mc Clloch- Pitts en 1940[20]. C'est un seul neurone artificiel avec d valeurs d'entrées résumées dans un vecteur de dimension d , $X = (x_1, x_2, ..., x_d)^T$,une unité de transfert se compose d'un additionneur (pour sommer les signaux d'entrées) ,et une fonction d'activation à seuil g, tel que l'unité de transfert (le traitement est relié avec les entrées par des poids synaptiques caractérisés par w_i ou $i = 1, ..., d$, et une seule sortie y. "Ce neurone permet de séparer ou « partitionner » l'espace des données à l'entrée en deux parties séparées par un hyperplan selon le résultat de classification de l'entrée en 1 ou 0"[3].

Alors la tâche discutée dan ce paragraphe consiste à classifier correctement les vecteurs d'entrée en deux groupes (1ou0). Pour simplifier les études, nous pouvons considérer un perceptron avec seulement deux entrées schématisées dans la figure suivante. voir figure (3.6)

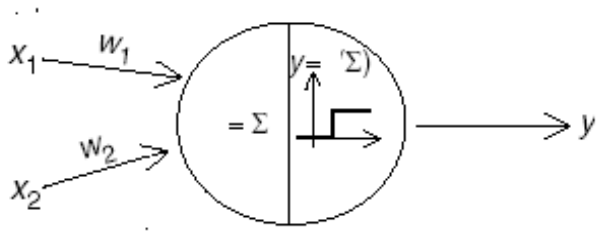

FIG. 3.6 – Un perceptron avec deux entrées [23].

Le neurone reçoit les deux entées multipliées par les poids w_1 et w_2 respectivement qui donnent la valeur d'entrée nette a où $a = w_1x_1 + w_2x_2$ puis l'activation de fonction à seuil de l'entrée nette a donnée comme suit :

$$f(a) = \begin{cases} 1 \ si \ a \geq 0 \\ 0 \ si \ a \leq 0 \end{cases} \tag{3.5}$$

Dans ce cas, le seuil est 0, Si $a = 0$ alors $w_1x_1 + w_2x_2 = 0$
Donc,

$$x_1 = \left(\frac{-w_2}{w_1}\right) x_2 \tag{3.6}$$

Cette équation présente la frontière de décision. On remarque qu'elle est une frontière linéaire.

3.4.2 Comparaison de perceptron avec l'analyse discriminante linéaire

Nous avons vu dans le chapitre (02) que l'analyse discriminante est une méthode statistique multivariable employée pour analyser simultanément les frontières entre les catégories en termes de plusieurs variables numériques indépendantes. Elle peut être employée également comme un classificateur dans lequel un ensemble de variables d'entrées est affecté à une classe cible. Par conséquent, l'analyse discriminante linéaire peut être employée pour classifier des données d'entrée comme nous avons déjà vu dans le chapitre (02).

Comme classificateur, le perceptron et l'analyse discriminante linéaire sont équivalents.

3.4.3 L'apprentissage de perceptron

La sortie de perceptron peut être exprimée par la forme suivante

$$g(a) = \begin{cases} -1 & \text{si} \quad a < 0 \\ 1 & \text{si} \quad a > 0 \end{cases} \tag{3.7}$$

Et

$$y(x) = g(\sum_{j=0}^{N} w_j x_j) \tag{3.8}$$

On peut écrire

$$y(x) = g(w^t x) \tag{3.9}$$

Où $\cdot\ g$ est défini dans (3.7) .

$\cdot w^t$ est le transposé du vecteur de poids w

$\cdot\ x$ est le vecteur d'entrée

Notre but est de trouver une phase d'utilisation de perceptron pour classer un vecteur d'entrée dans la classe c_1 ou c_2 (discrimination en deux classes). Supposons que nous associons à chaque vecteur d'entrée x_n une valeur cible t_n telle que la sortie cible de réseau est

$$\begin{cases} t_n = 1 \text{ si } x_n \in c_1 \\ t_n = -1 \text{ si } x_n \in c_2 \end{cases} \tag{3.10}$$

D'après les expressions (3.7) et (3.9) on a

$$\begin{cases} et & w^t x > 0 \quad \text{pour } x \in C_1 \\ & w^t x < 0 \quad \text{pour } x \in C_2 \end{cases} \tag{3.11}$$

Et d'après (3.10) et (3.11) on a

$$w^t x_n t_n > 0 \qquad \text{pour tout les vecteurs } x_n \tag{3.12}$$

Ceci suggère que nous essayons de réduire au minimum la fonction d'erreur suivante ; connue sous le nom de critère de perceptron :

$$E^{perc}(w) = -\sum_{x^n \in M} w^t x_n t_n \tag{3.13}$$

Où M est l'ensemble des vecteurs x_n qui sont mal classifiés

Si nous appliquons la règle delta donnée dans le paragraphe (3.3.2) à la fonction erreur présentée par (3.9)

$$\frac{\partial E^{perc}}{\partial w_{kj}} = x_n t_n \tag{3.14}$$

Alors nous obtenons :

$$w_{kj}^{(\tau+1)} = w_j^{(\tau)} + \eta \; x_n t_n \tag{3.15}$$

Ceci correspond à un algorithme d'apprentissage très simple. Il est facile de voir que ce procédé tend de réduire la fonction erreur comme suit :

Comme

$$\|x_n t_n\|^{\tau} \rangle 0 \quad \text{et} \quad \eta \rangle 0 \tag{3.16}$$

D'après (3.16) nous remarquons que

$$E^{perc}\left(w^{(\tau+1)}\right) \quad \langle \quad E^{perc}\left(w^{(\tau)}\right) \tag{3.17}$$

Alors, la fonction erreur est diminuée.

Remarque 13 *Pour le cas particulier de cette fonction d'erreur (le critère de perceptron $E^{perc}(w)$), nous voyons que la valeur de η est en fait sans importance puisqu'un changement de poids équivalent à une regraduation de poids et de biais.*

Remarque 14 *Nous pouvons prendre $\eta = 1$, cette propriété ne se tient pas, pour la plupart d'autres formes de fonction d'erreur.*

3.4.4 Théorème de convergence de perceptron

Il y'a un résultat intéressant énoncé dans le théoreme de convergence de perceptron.

Théorème 15 *Téorème de convergence de perceptron*

Pour n'importe quel ensemble de données linéairement séparables, la règle d'apprentissaformesge du perceptron est garantie pour trouver une solution en un nombre fini d'étapes[17].

Preuve. Puisque nous considérons un ensemble d'apprentissage qui est linéairement séparable

Nous avons vu qu'il existe au moins un vecteur de poids $\widehat{w}$ pour lequel tous les vecteurs x_n sont correctement classifiés ; de sorte que :

$$\widehat{w}^T x_n \; t_n \;\; \rangle \quad 0 \;\; \text{pour tout} \;\; n \tag{3.18}$$

L'apprentissage commence par un certain nombre de vecteurs arbitraires de poids. Sans perdre la généralité, nous pouvons supposer que le vecteur est égal à zéro.

A chaque étape de l'algorithme ; le vecteur de poids employé est mis à jour .

$$w^{(\tau+1)} = w^{(\tau)} + x_n t_n \tag{3.19}$$

Où x_n est un vecteur d'entrée qui est mal classifié par le perceptron, on a

$$\begin{aligned} w^0 &= 0 \\ w^1 &= x_n t_n \\ w^2 &= 2\,(x_n t_n) \\ &\vdots \end{aligned}$$

Supposons qu'après répétition de l'algorithme pendant un certain nombre d'etapes, le nombre de fois où chaque vecteur x_n présenté est mal classifé est τ^n.

Alors, le vecteur de poids à ce moment sera donné par :

$$w^{(\tau)} = \tau^n x_n t_n \tag{3.20}$$

Et le vecteur de poids pour tous les points d'apprentissage est donné par

$$w = \sum_n \tau^n x_n t_n \tag{3.21}$$

Nous prenons maintenant le produit scalaire de cette équation, nous obtenons :

$$\widehat{w}^T w = \sum_n \tau^n \; \widehat{w}^T x_n t_n \quad \geq \tau \min_n \left(\widehat{w}^T x_n t_n\right) \tag{3.22}$$

Où $\tau = \sum_n \tau^n$ est le nombre total d'étapes où le poids en mis à jour et l'inégalité (3.22) vérifiée alors,

Le résultat est obtenu en remplaçant chaque vecteur mis à jour par le plus petit des vecteurs mis à jour,. à partir de (3.14) et (**??**), nous remarquons que $\widehat{w}^T w$ est majorant d' une fonction qui se développe linéairement avec τ alors :

$$\begin{aligned} F(\tau) &\leq \widehat{w}^T w \\ \text{où } F(\tau) &\longrightarrow +\infty \\ \text{si } \tau &\longrightarrow +\infty \end{aligned} \tag{3.23}$$

Nous considérons maintenant la grandeur des vecteur de poids w :
D'après(3.19). on a :

$$\begin{aligned} \left\|w^{(\tau+1)}\right\|^2 &= \left\|w^{(\tau)} + x_n t_n\right\|^2 \\ &= \left\|w^{(\tau)}\right\|^2 + \|x_n\| \left(t_n\right)^2 + 2w^{(2)T} x_n t_n \end{aligned} \tag{3.24}$$

Et on a si x_n est un vecteur mal classifié , alors :

$$w^{(\tau)T} x_n t_n \langle 0 \tag{3.25}$$

Et en utilisant (3.25) et (3.24), nous obtenons :

$$\left\|w^{(\tau+1)}\right\|^2 \leq \left\|w^{(\tau)}\right\|^2 + \|x^n\|^2 \left(t_n\right)^2 \tag{3.26}$$

Nous avons $\left(t_n\right)^2 = 1$ car $t_n = \mp 1$ et $\|x_n\|^2 \leq \|x\|^2_{\max}$ où $\|x\|_{\max}$ est la longueur du plus long vecteur d'entrée , alors (3.26) se réduit à :

$$\Delta \|w\|^2 = \left\|w^{(\tau+1)}\right\|^2 - \left\|w^{(n)}\right\|^2 \leq \|x\|^2_{\max} \tag{3.27}$$

après τ étapes, le vecteur de poids mis à jour, nous avons :

$$\|w\|^2 \leq \|x^*\|^2_{\max} \tag{3.28}$$

Alors , nous voyons que pour une valeur de τ suffisamment grande ($\tau \longrightarrow +\infty$) les deux résultats (3.23) et (3.28) deviennent incompatibles. Nous pouvons conclure alors que l'algorithme doit converger dans un nombre fini d'étapes.
■

3.4.5 Un exemple pratique

Un problème simple mais réaliste sera résolu en utilisant le perceptron. Dans cet exemple, le but est de classer un ensemble de 59 individus en deux classes $c_1$et c_2, telle que la classe c_1 représente les individus diabétiques et ceux non diabétiques sont présentés par la classe c_2 selon le taux de glycémie veineuse x_1 et le taux de glycémie post prandiale x_2 (le tableau de données provient du centre de prélèvement sanguin du CHU Constantine. Il est représenté à l'annexe A). Cette section étudiera à quel point le perceptron classifie les patients. Puisqu'on a deux variables (problème bidimensionnelle), un perceptron avec deux entrées doit être formé. Seulement un neurone de sortie représente les deux classes avec un résultat 0 associe la classe c_1, et 1 pour l'autre classec_2.

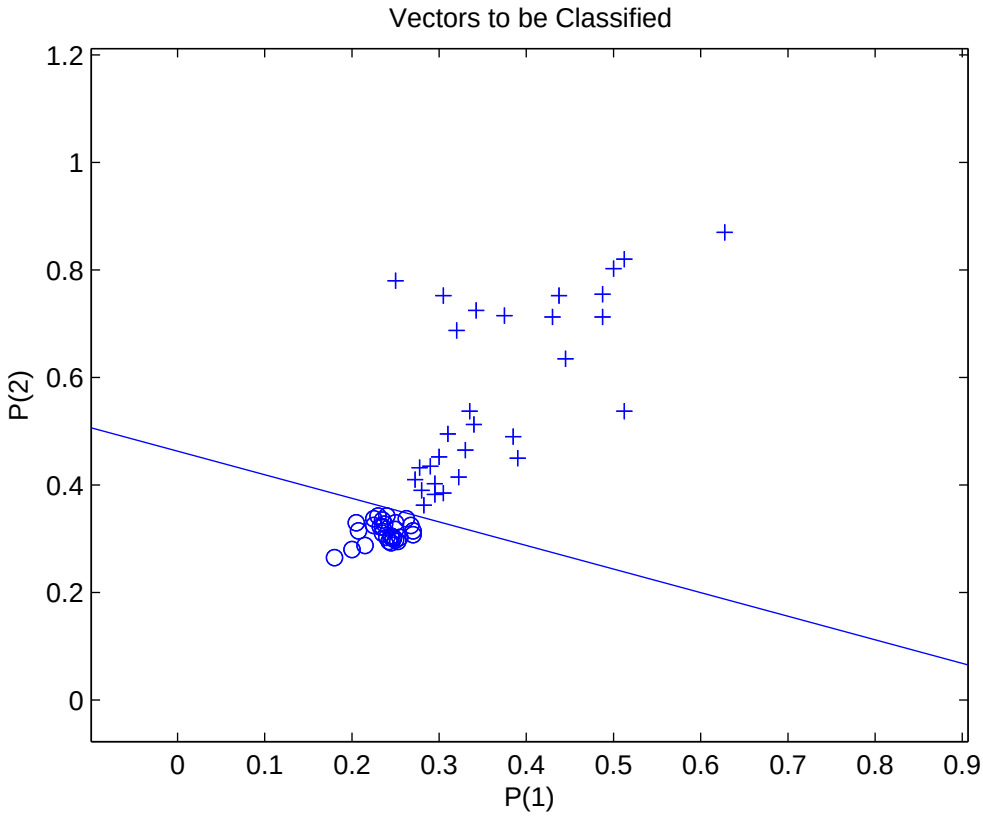

FIG. 3.7 – La frontière de décision finale superposée aux données

Le traitement de ces données avec le logiciel MATLAB donne la figure suivante ou on note les individus de classe c_1 par des cercles et l'autre classe par des plus(+).

Dans la figure (3.7), la frontière de décision finale est superposée aux données.ceci prouve que le perceptron a trouvé la meilleure frontière de décision linéaire pour ce problème. Nous remarquons qu'il n'existe pas de points de classification fausse alors la performance est de 100 pour cent.

Remarque 16 *Si le problème de classification implique plus de deux classes,alors, il peut être résolu par un perceptron à plusieurs sorties.*

Remarque 17 *Le perceptron et le réseau de perceptrons peuvent être prolongés à la classification multidimensionnelle (2 classes, multi classes) quand les dimensions du modèle d'entrée sont plus grandes que deux.*

3.5 Réseaux de neurones linéaires

Dans la classification linéaire, le perceptron et le réseau de perceptrons ont seulement deux sorties (0ou1). Dans la suite, nous pouvons généraliser les études pour le cas où la sortie est une quantité continue c'est –à-dire, elle peut prendre plusieurs valeurs. Dans ce cas, nous utilisons le neurone et les réseaux linéaires. Dans les paragraphes précédents, nous avons défini le neurone, la fonction d'activation linéaire et la dimensionnalité (les entrées) , alors on peut dire que le neurone linéaire est un seul neurone avec une fonction d'activation linéaire. Ceci est décrit dans le modèle de neurone dans la figure [3.8]

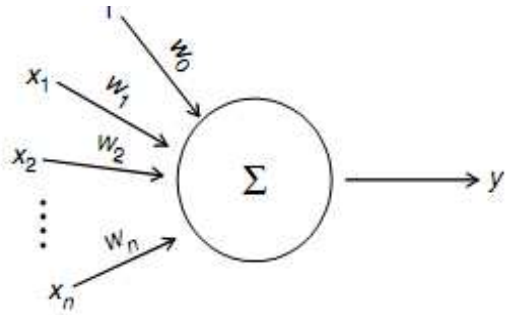

FIG. 3.8 – Le modèle de neurone linéaire [23].

Où il y a plusieurs entrées et une entrée de biais qui est égale à 1.

Widrow et Hoff (1960) ont développé le premier modèle de neurone linéaire adaptatif (ADALINE), et pour la première fois, il mise en application l'apprentissage supervisé par correction d'erreur (la règle delta présenté dans le paragraphe 3.3.2).

Dans ce paragraphe, nous allons explorer les larges possibilités de classification et de prédiction du neurone linéaire générale et nous allons examiner comment il est entraîné par la règle delta.

3.5.1 Le neurone linéaire comme classificateur

Ce paragraphe va examiner comment un neurone linéaire peut être entraîné comme classificateur en utilisant la règle delta.

Il est possible de la correction du neurone linéaire comme un classificateur en passant la sortie à trouver une fonction seuil, telle qu'elle est montrée dans la figure[3.9]

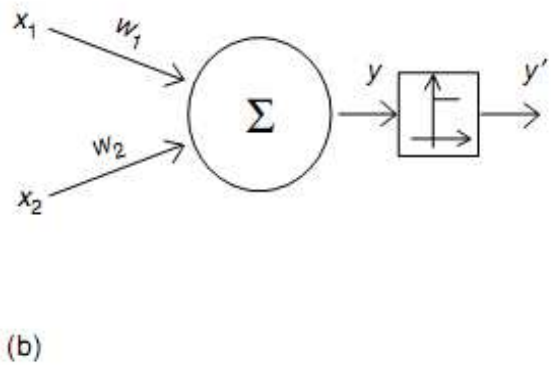

FIG. 3.9 – le neurone linéaire classificateur avec deux entrées [23].

Cependant, l'erreur est calculée en se basant sur la sortie linéaire qui est continue, la sortie du classificateur est donnée comme suit :

$$y^{'} = \begin{cases} 1 \ si \ y \geq 0 \\ 0 \ si \ y \leq 0 \end{cases} \tag{3.29}$$

En effet, il y a deux variables d'entrées, il est nécessaire d'utiliser un neurone linéaire à deux entrées et une sortie, tel qu'on le voit dans la figure [3.8]

Pour les entrées x_1, x_2 , ayant les poids correspondants w_1, w_2 , sans l'entrée du biais , il faut calculer d'abord l'entré nette a :

$$a = w_1 x_1 + w_2 x_2 \tag{3.30}$$

La transformation linéaire donne (dans notre cas ici c'est l'identité) :

$$g(a) = a \text{ alors } y = a \tag{3.31}$$

Ainsi mathématiquement, le neurone linéaire produit un modèle linéaire dont l'équation pour la sortie est simplement :

$$y = a = w_1 x_1 + w_2 x_2 \tag{3.32}$$

L'apprentissage avec la règle delta (deux entrées)

Pour commencer l'apprentissage par correction d'erreur, il faut qu'on ait les valeurs de vecteurs cibles t (vecteur bidimensionnelle)

On a :

$$\begin{aligned} a &= w_1 x_1 + w_2 x_2 \\ y &= a \\ E &= t - y \\ &= t - w_1 x_1 - w_2 x_2 \end{aligned} \tag{3.33}$$

D'après la règle delta, où

$$\begin{aligned} \triangle w &= \eta \frac{\partial E}{\partial w} \\ &= \eta E x \end{aligned} \tag{3.34}$$

η : est le taux d'apprentissage.

Posons les poids initiaux donnés comme $w_1^{(0)}$, $w_2^{(0)}$, et fixons le taux d'apprentissage η . Alors après l'$i$ème itération on a :

$$\begin{aligned} w_1^{(i+1)} &= w_1^{(i)} + \eta \; x_1 E \\ w_2^{(i+1)} &= w_2^{(i)} + \eta \; x_2 E \end{aligned} \tag{3.35}$$

Pour chaque entrée, le réseau donne la sortie y.

La fonction à seuil décide de la valeur de $y^{'}$d'après le signe de y .

Si $y^{'} \neq t$ alors la classification est incorrecte, alors il y a une erreur E donnée par : $E = t - y$

Donc, il y a une variation des poids de réseau qui sont présentée dans ce qui suit :

$$\begin{aligned} \triangle w_1^{(0)} &= \eta\ E\ x_1 \\ \triangle w_2^{(0)} &= \eta\ E\ x_2 \end{aligned} \tag{3.36}$$

Les nouveaux poids après la première incrémentation sont :

$$\begin{aligned} w_1^{(1)} &= w_1^{(0)} + \triangle w_1^{(0)} \\ w_2^{(1)} &= w_2^{(0)} + \triangle w_2^{(0)} \end{aligned} \tag{3.37}$$

Ensuite, nous entraînons le réseau avec un autre vecteur d'entrée avec le nouveau vecteur de poids $w\ \left(w_1^{(1)}, w_2^{(1)}\right)$

Après cette opération le réseau décide : si la classification est correcte, les poids ne seront pas modifiés, sinon le réseau répètera la même procédure que précédemment jusqu'à obtenir $E = 0$

A la fin, le neurone linéaire a classé correctement toutes les observations données.

La frontière de classification pour le neurone linéaire entraîné est donnée par $y = 0$

Comme $y = a$ alors

$$y = a = w_1 x_1 + w_2 x_2 = 0 \tag{3.38}$$

Donc

$$x_2 = -\left(\frac{w_1}{w_2}\right) x_1 \tag{3.39}$$

Remarque 18 *Avec des données nombreuses le neurone linéaire et le perceptron produiront un résultat identique.*

Remarque 19 *Dans les problèmes de grandes dimensions les classes sont séparées par un hyperplan ; le concept d'apprentissage s'applique sans changement.*

3.5.2 Les propriétés de classification du neurone linéaire comme capacités prédictives

La classification est seulement l'une des capacités du neurone linéaire. Nous avons vu précédemment que nous pouvons entraîner le neurone linéaire comme classificateur. Mais il est capable de donner beaucoup plus que celle. Il peut aussi être un prédicteur. Pour classifier cela, il est utile de regarder la

forme générale de la relation entre entrées et sorties établies par le neurone linéaire pour voir que la classification est une forme de prédiction quand l'angle de vue est restreint.

La sortie du neurone avec deux entrées comme nous avons vu précédemment est $y = w_1x_1 + w_2x_2$,où y est un plan appelé des plan solutions et il est montré dans la figure [3.10]

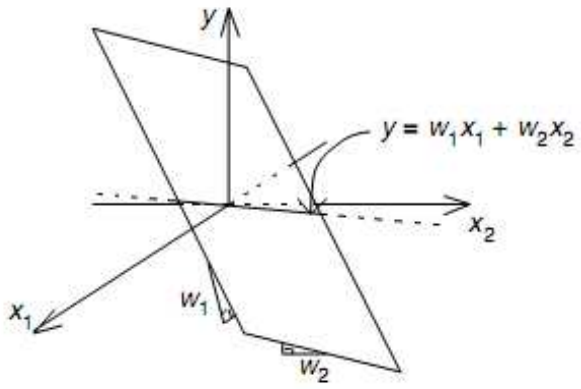

FIG. 3.10 – La frontière de décision de neurone linéaire avec le plan des solutions [23].

Ici w_1 est la pente suivant l'axe de x_1 , et w_2 et la pente suivant l'axe x_2.Chaque vecteur d'entrée (x_1, x_2) induit une seule image dans le plan des solutions

Classification

Dans le plan des solutions une droite sert à la classification. C'est la droite où ce plan est coupé par le deuxième plan (x_1, x_2) (c'est –à-dire $y = 0 : 0$ est le seuil).Elle est la frontière de décision. Les valeurs positives $(y \gtrdot 0)$ classent l'observation dans une catégorie et les valeurs négatives $(y \lessdot 0)$ classent l'observation dans l'autre catégorie $(y \lessdot 0)$ correspond au demi-plan en dessus du plan $(x_{1,}x_2)$ et $(y \lessdot 0)$ correspond au demi-plan en dessous du plan $(x_{1,}x_2)$.

Remarque 20 *Le neurone linéaire produit aussi les résultats comparables à celle de l'analyse discriminante linéaire de la statistique.*

Prédiction

Le plan entier sert à la prédiction. Ce qui vu ici est que les ajustements des poids altèrent la pente du plan des solutions jusqu'au l'erreur entre la cible et la sortie soit minimisée. Ce plan là n'est pas produit par le perceptron. Le percetron ne peut que nous fournir la droite frontière dans le plan.

Nous pouvons voir que le neurone linéaire est capable de produire une application continue des entrées vers les sorties, c'est tout qui est requis dans la prédiction ou dans l'approximation des fonctions. Nous allons voir en détail les possibilités prédictives du neurone linéaire.

3.5.3 Neurone linéaire comme prédicteur

Nous avons vu que le neurone linéaire produit un modèle linéaire dont l'équation pour la sortie est simplement $y = w_1x_1 + w_2x_2 + ... + w_nx_n + w_0$. En terminologie statistique , on dit que la sortie y régresse sur les entrées $x_1, x_2...x_n$, donc le modèle du neurone linéaire est analogue aux modèles de la régression linéaire multiple en statistique (nous discuterons cela dans les paragraphes suivants). Dans l'approximation de fonction, c'est la gradeur de la sortie y qui entre en considération et non pas la catégorie ou la frontière de décision, cela veut dire que durant l'entraînement , c'est la fonction linéaire qui minimise l'erreur qui est cachée. Pour la simplicité, nous allons commencer par le cas d'un neurone linéaire à une seule entrée x et sans biais comme il est montré dans la figure [3.11]

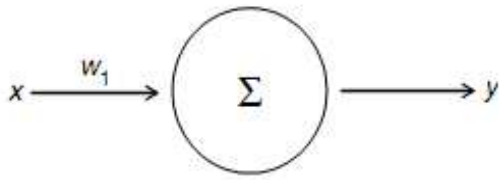

FIG. 3.11 – Le neurone linéaire avec une seul entrée [23].

Il y a un seul poids w et la sortie du neurone linéaire prend la forme $y = wx$. Dans ce cas, le poids définit la pente de la droite et lorsque nous donnons au poids une valeur aléatoire, elle devrait par la suite s'installer sur la valeur exacte. Alors, l'apprentissage revient à trouver la valeur exacte du poids qui fixera la pente de la droite.

L'apprentissage avec la règle delta (en cas d'un seul neurone linéaire) La sortie de neurone précédent est donnée par $y = wx$ dont la valeur cible est t, alors,

$$\begin{aligned} e &= t - y \\ &= t - wx \end{aligned} \tag{3.40}$$

Donc,

$$\begin{aligned} \Delta w &= \eta \frac{\partial E}{\partial w} \\ &= \eta\ e\ x \end{aligned} \tag{3.41}$$

Où

$$E = \frac{1}{2}e^2 = \frac{1}{2}\left(t - w_1 x\right)^2$$

Le poids de neurone à la $\tau^{ème}$ itération peut être exprimé comme suit

$$\begin{aligned} w^{(\tau+1)} &= w^{(\tau)} + \Delta w^{(\tau)} \\ &= w^{(\tau)} + \eta\ e^{(\tau)}\ x \end{aligned} \tag{3.42}$$

3.5.4 Comparaison du modèle de neurone linéaire avec la régression linéaire

En tant que prédicateur, le neurone linéaire est fonctionnellement équivalent à la régression linéaire simple si nous utilisons un neurone linéaire avec juste une seule entrée et un biais 1 qui correspond aux valeurs de poids w_0 alors la sortie de neurone est donnée par

$$y = w_0 + w_1 x_1 \tag{3.43}$$

L'équation (3.43) est similaire à la régression linéaire simple. S'il y a plusieurs variables alors ceci rend l'utilisation d'un neurone linéaire avec plusieurs entrées (et un biais) est nécessaire.Dans un neurone avec entrées multiple, sa sortie est exprimée par l'expression suivante

$$y = w_1 x_1 + w_2 x_2 + ... + w_n x_n \tag{3.44}$$

L'équation (3.44) montre qu'un neurone linéaire à plusieurs entrées est équivalent à la régression linéaire multiple. Nous avons vu dans le premier chapitre que dans la régression linéaire multiple, les coefficients (interception et pentes) d'une relation entre une variable dépendante et plusieurs variables indépendantes sont cherchés tels que la somme de moindre carrés de l'ensemble des données est réduite au minimum.

Le neurone linéaire ne fait aucune hypothèse au sujet de la distribution des données tandis que la régression linéaire (simple ou multiple) suppose que les variables ε_i sont normalement distribuées et que la variance de la variable ε_i est constante à travers la gamme des variables explicatives x_i (homoxédasticité).

Remarque 21 *Beaucoup d'unités linéaire de neurone peuvent être collées pour former un réseau à une seule couche avec plusieurs sorties linéaires comme illustré par le schéma [3.15]. Dans la classification, chaque neurone représente une classe (équivalent au réseau de perceptrons multiple). Alors, le classificateur linéaire à plusieurs sorties est équivalent à un classificateur de fonction discriminante linéaire multicatégoriel.*

Remarque 22 *Pour montrer l'équivalence de résolution de ces tâches par les réseaux de neurones et par les méthodes statistiques, nous pouvons utiliser les deux méthodes pour les mêmes données et comparer les deux résultats corresponds*

3.5.5 Un exemple pratique

Dans le paragraphe précédent, nous avons discuté la capacité de discrimination de neurone linéaire. Pour confirmer cette propriété, nous avons mesuré le taux de cholestérol x_1 et la tension artérielle moyenne x_2 ($x_2 = \frac{tension\ systolique + tension\ diastolique}{2}$) afin de séparer les groupes des patients victimes d'AVC (le premier groupe c_1) ou non (le deuxième groupe c_2) parmi les 59 patients. Le traitement de ces données avec le logiciel de MATLAB représenté dans la figure (3.12), ou on note les individus de classe c_1 par des plus (+) et les individus de classe c_2 par des cercle ($\circ$). (le tableau de données provient du centre de prélèvement sanguin du CHU Constantine .Il est donné dans l'annexe B).

Dans la figure (3.12), la ligne de frontière de décision de classificateur linéaire est superposée aux données. Pour la classe c_2, il n'y a pas des points mal classés alors la performance par rapport à c_2 est de cent pour cent, mais pour la classe c_1, il y a des points de mal classification qui sont 8 parmi les 22 points dans c_1, alors la performance par rapport à c_1 est égale à 63.64 pour cent. Les deux résultats précédents prouvent que la performance globale est égale 81.82 pour cent. Alors la performance de ce traitement n'est pas cent pour cent car les points ne sont pas linéairement séparables

Dans le traitement des données par le logiciel MATLAB, nous remarquons que le neurone linéaire trouve le meilleur poids en employant la règle delta plus rapidement que le perceptron.

3.6 Limitation d'un réseau à une seule couche

Nous avons remarqué dans le deuxième exemple pratique que la performance de classificateur linéaire n'est pas de cent pour cent car les données

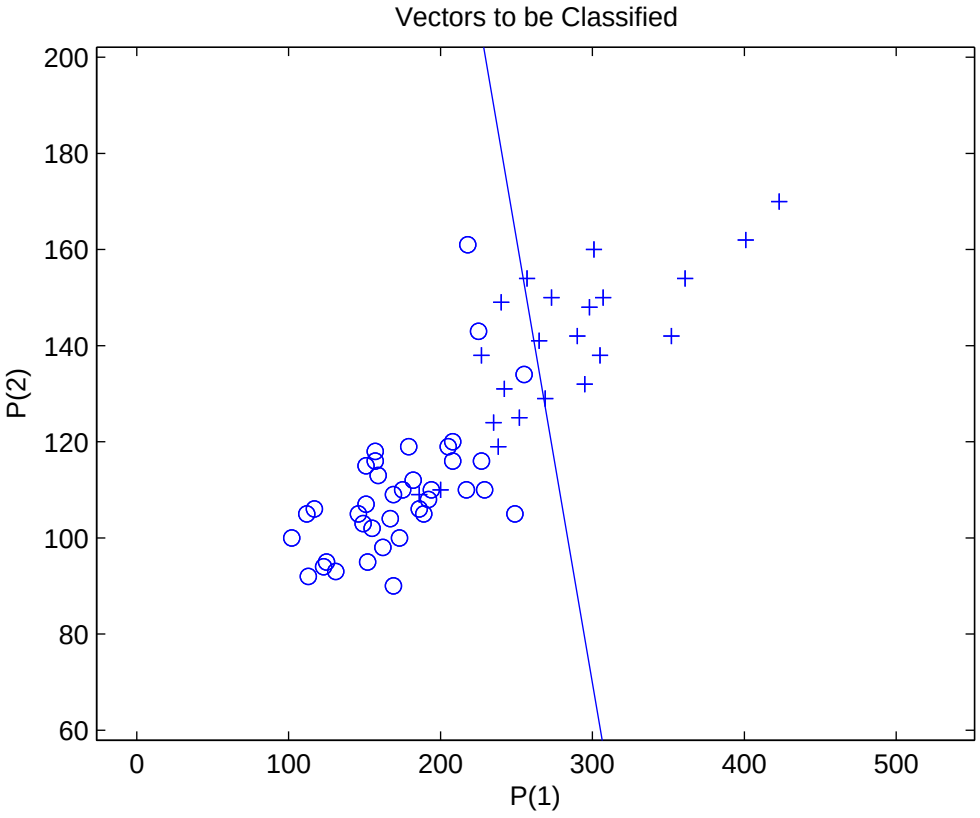

FIG. 3.12 – La frontière de décision de classificateur linéaire superposée aux données.

ne sont pas linéairement séparables. Si on applique les mêmes donnés au perceptron, alors l'algorithme d'apprentissage ne se terminera jamais. De même pour un réseau à une seule couche, il peut seulement classifier les points qui sont linéairement séparables, car ce type de réseau correspond aux fonctions discriminantes ayant une frontière de décision linéaire, ou plus généralement, hyperplans dans des dimensions plus élevées, alors il est une classe de réseau très étroite de fonction discriminante possible, et dans beaucoup de situations pratiques, il ne peut pas représenter un choix optimal. Dans la littérature sur le calcul neuronale, beaucoup d'attention est souvent accordée à l'incapacité des réseaux à une seule couche pour résoudre des problèmes simples tels que le problème XOR résumé dans la figure (3.13) Ceci fournit la motivation principale pour l'usage de réseaux multicouches.

entrée x1	entrée x2	sortie
0	0	0
0	1	1
1	0	1
1	1	0

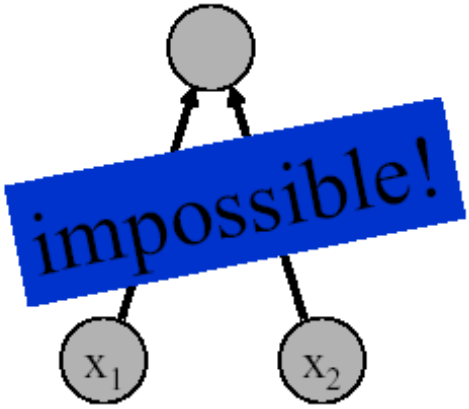

FIG. 3.13 – Le problème XOR [3].

Dans le paragraphe suivant de ce chapitre, nous prolongerons la discussion de l'analyse linéaire au sujet de l'analyse non linéaire en utilisant les réseaux de neurones. Les réseaux de perceptron multicouches présentés dans le paragraphe (3.2.2) sont présentés en détail afin de mettre en exergue l'in-

teret du traitement non linéaire dans les réseaux de neurones. La puissance de ces réseaux vient de la couche cachée des neurones. Car si on ajoute une couche de neurones, le nombre de paramètres augmentera et pour un modèle quelconque, l'augmentation de nombre du paramètres (le degré de liberté) rend ce modèle plus flexible. Pour cela, le perceptron multicouches est très flexible et peut être formé pour assumer la forme des modèles de données, indépendamment de la complexité de ces modèles.

Pour des problèmes fortement non linéaires (complexes) un perceptron multicouche est impliqué. Ce réseau peut être nécessaire pour rapprocher correctement la relation entre les entrées et les variables cibles.

3.7 Perceptron Multicouche (PMC)

La disposition d'un réseau de (PMC) avec une couche cachée est indiquée dans le schéma [3.14].

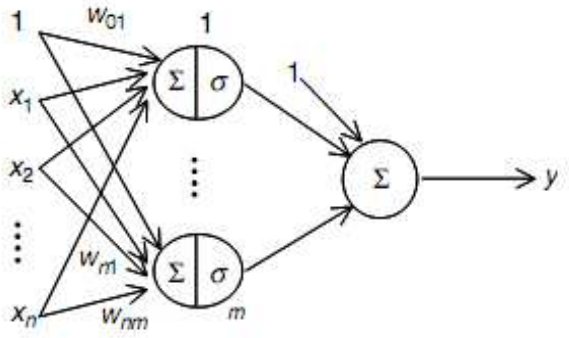

FIG. 3.14 – Le modèle de PMC [23].

Sur le schéma [3.14], les x_1,...,x_n sont des variables d'entrées comportant la couche d'entrée. Ce type de réseau peut rapprocher n'importe quelle relation fonctionnelle complexe multidimensionnelle (de dimension n).

3.7.1 Le perceptron multicouche à deux entrées

La compréhension gagnée avec deux entrées peut être prolongé à trois entrées ou plus. en regardant deux entrées, il est possible d'établir une base pleine pour la compréhension des réseaux avec plusieurs d'entrées, parce que

la connaissance que nous avons gagné de deux entrées (problèmes bidimensionnels) est généralisée à beaucoup d'entrées (problèmes multidimensionnelles). C'est possible, par ce que l'information de principe fondamental de PMC peut être extraite à partir de ces exemples.

Un réseau avec deux entrées peut rapprocher n'importe quelle sortie qui dépend de deux variables indépendantes. Par conséquent, elle peut résoudre n'importe quel problème de prédiction bidimensionnelle ou problème de classification.

La structure d'un réseau à deux entrées est représentée sur le schéma [3.15], dans lequel il y a deux entrées, un ou plusieurs neurones cachés, et une sortie. Pour des problèmes de classification impliquant plus de deux classes, il est nécessaire d'employer un neurone de sortie pour chaque classe. Cependant pour la plupart des problèmes de prédiction, seulement un neurone est nécessaire.

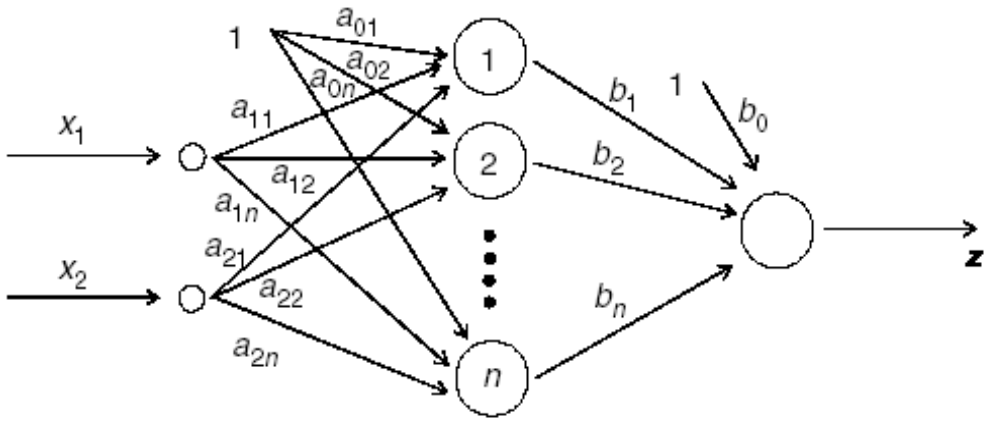

FIG. 3.15 – Le PMC avec deux entrées.[23]

La prédiction avec le PMC à deux entrées

Traitement des entrées bidimensionnelles par le neurone caché La discussion de ce réseau est traitée en commençant par le traitement d'un neurone caché isolé comme il est représenté dans le schéma [3.16]

On assume que toutes les fonctions d'activation sont logistiques. Chaque neurone caché reçoit deux entrées et le biais qui sont multipliés par les poids correspondants et additionnés pendant la première étape de calcul. La somme pondérée a est donnée par

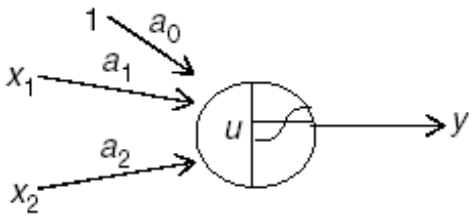

FIG. 3.16 – Un seul neurone caché non linéaire avec deux entrées [23].

$$a = w_1 + w_1 x_1 + w_2 x_2 \tag{3.45}$$

Cette équation est un plan dans l' espace bidimensionnel (x_1, x_2). Comme c'est représenté dans le schéma [3.17].

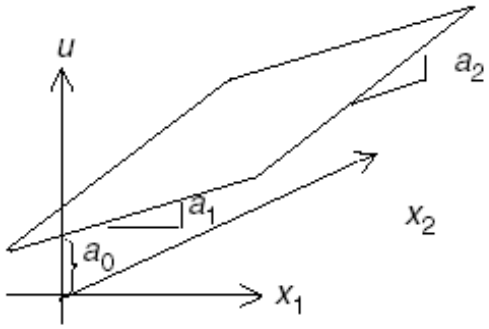

FIG. 3.17 – L'équation de somme nette présente un plan dans l'espace bidimensionnel [23].

Par conséquent, l'effet d'apprentissage doit tracer les entrées x_1 et x_2 à un plan bidimensionnel et pour commander complètement la position et l'orientation de plan dans l' espace bidimensionnel par w_0, w_1, w_2

L'entrée nette a est passée par la fonction logistique pour obtenir la sortie de neurone caché z comme suit :

$$z = \frac{1}{1 + \exp^{-a}} \tag{3.46}$$

Où z est une fonction logistique. En substituant a dans(3.46) on trouve :

$$z = \frac{1}{1 + \exp^{-(w_1 + w_1 x_1 + w_2 x_2)}} \tag{3.47}$$

Maintenant z et a seront explorés pour plusieurs cas d'après les valeurs : w_0, w_1,et w_2.

1- $w_0 = 0, w_1 = 1, w_2 = 0$

La courbe de z en fonction de x_1 et x_2 pour ce cas est montrée dans la figure [3.18]qui dépend d'une fonction logistique dans l'espace bidimensionnel. La valeur de poids $w_1$commande la pente de la fonction par rapport à l'axe x_1.

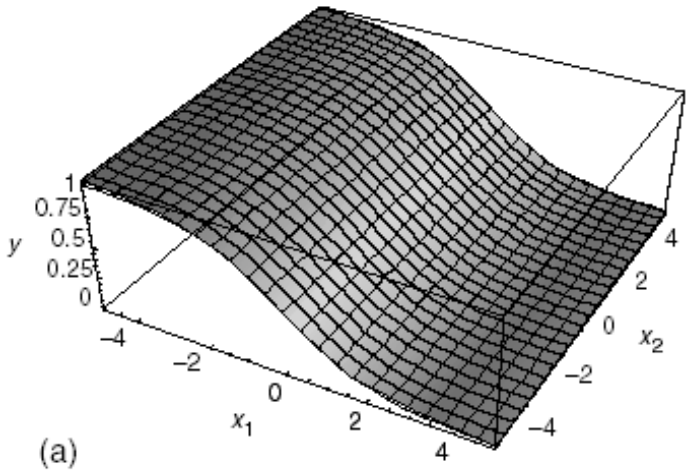

FIG. 3.18 – La fonction bidimensionnelle pour $w_0 = 0$, $w_1 = 1$, $w_2 = 0$

Puisque le poids $w_2 = 0$ la pente par rapport à l'axe x_2 est zéro et $w_0 = 0$, la fonction est centrée à $x_1 = 0$ et $x_2 = 0$.

2- $w_0 = 0, w_1 = 0, w_2 = 1$

Ce cas est représenté dans la figure [3.19] qui démontre que w_2 commande la pente de la fonction par rapport x_1 et la pente par rapport x_2 est zéro et la fonction étant centrée sur $x_1 = 0$ et $x_2 = 0$

.

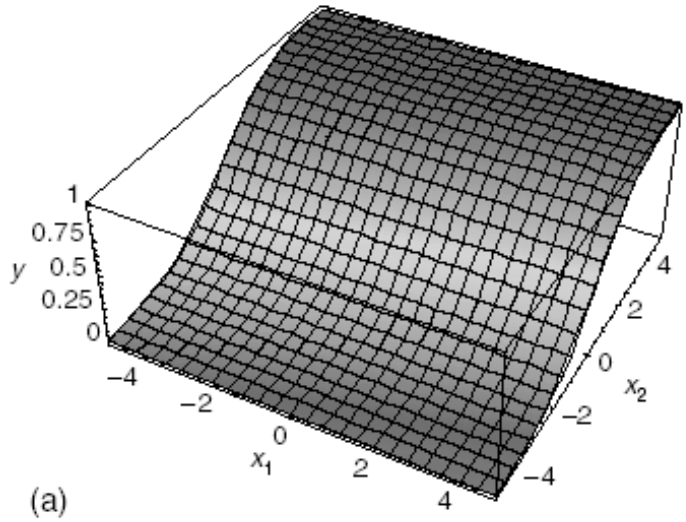

FIG. 3.19 – La fonction bidimensionnelle pour $w_0 = 0$, $w_1 = 0$, $w_2 = 1$ [23]

3-$w_0 = 0, w_1 = 1, w_2 = 2$

Dans ce cas , où $w_1$et w_2, qui commandent les pentes, sont non nulles, alors une fonction logistique plus complexe est produite, comme il est représenté dans la figure [3.20].

4-$w_0 = -0,5, w_1 = 1, w_2 = -1$

Dans ce cas , la pente par rapport x_1 est positive, et celle par rapport x_2 est négative. Comme le démontre la figure[3.21].

Les illustrations graphiques ci-dessus prouvent que deux entrées sont representées par une fonction : logistique bidimensionnelle de z dont les pentes sont commandées par les poids w_1 et w_2, le poids w_0 décale la région de l'activation la plus élevée de la fonction logistique.

Il est possible de visualiser comment plusieurs neurones peuvent agir ensemble pour rapprocher une fonction bidimensionnelle ou un modele de prédiction des résultats de deux variables indépendantes fondamentalement, chaque neurone donne une fonction sigmoïde bidimensionnelle, dont la forme et la position sont commandées par ses poids w_0 ,w_1 et w_2 selon la nature non linéaire de la fonction rapprochée. De cette façon, plusieurs neurones ajoutent la puissance et une grande flexibilité au réseau de neurones pour l'identifica-

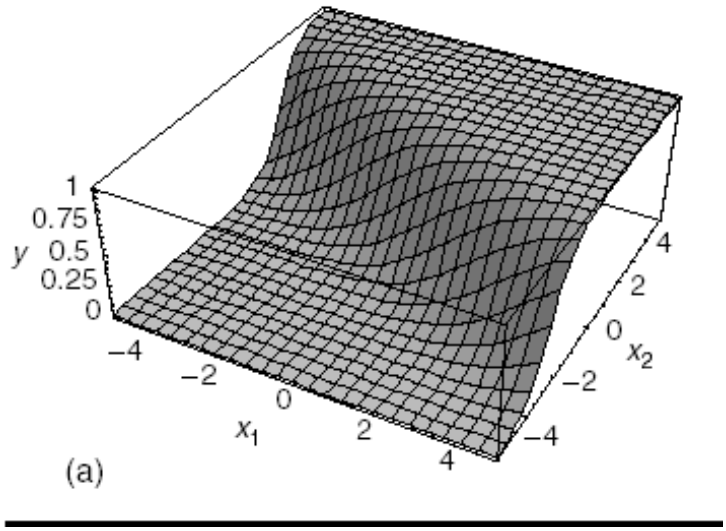

FIG. 3.20 – La fonction bidimensionnelle pour $w_0 = 0$, $w_1 = 1$, $w_2 = 2$[23]

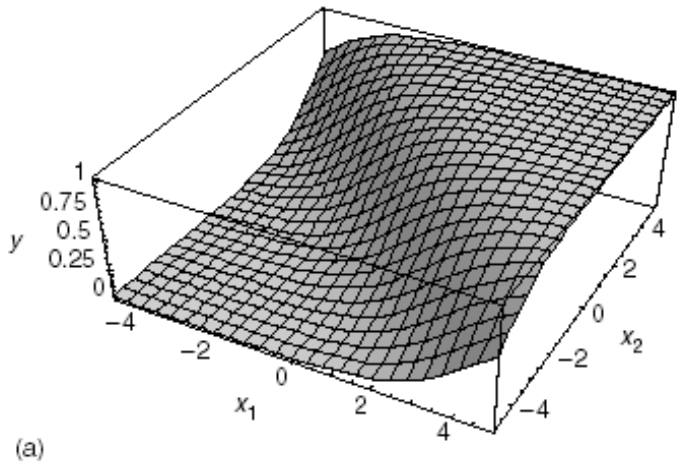

FIG. 3.21 – La fonction bidimensionnelle pour $w_0 = 0$, $w_1 = 1$, $w_2 = 2$ [23].

tion de modèle non linéaire et cela lui permet d'approcher n'importe quelle fonction bidimensionnelle pour la prédiction.

La sortie de réseau La dernière étape du traitement synthétise les sorties de neurone caché en calculant leur somme pondérée puis cette somme est traitée par la fonction d'activation du neurone de sortie. Cette somme produit la forme désirée de la valeur cible t. La valeur t dans le cas de prédiction peut être une surface arbitrairement complexe et non linéaire.

La classification avec le PMC à deux entrées

La prédiction et la classification sont fondamentalement le même problème en exceptant l'ajustement final. Alors la classification est un sous ensemble de problème de prédiction. La frontière finale de classification induit un plan à travers le modèle qui produit la surface horizontalemen. Celle-ci a la valeur de l'activation de sortie, et cette frontière peut être arbitrairement complexe et non linéaire divisant l'espace d'entrées dans des classes d'une façon complexe.

Nous avons considéré le même réseau PMC avec deux entrées, alors la somme pondérée est donnée par(3.46)

$$a = w_1 + w_1 x_1 + w_2 x_2 \tag{3.48}$$

Puis, a est passé par une fonction logistique (fonction d'activation d'un neurone caché) pour obtenir une sortie de ce neurone z donnée par(3.47)

$$z = \frac{1}{1 + \exp^{-a}} \tag{3.49}$$

La frontière est définie par $a = 0$, alors :

$$\begin{aligned} a &= w_1 + w_1 x_1 + w_2 x_2 \\ &= 0 \end{aligned} \tag{3.50}$$

Donc, nous remarquons que la frontière de décision est une droite. Dans ce cas aussi, nous examinons plusieurs cas de valeurs de w_0, w_1 et w_2 .

Nous prenons les mêmes cas précédents.

1-$w_0 = 0, w_1 = 1, w_2 = 0$

L'équation de la ligne de frontière passe par un plan horizontal à travers le milieu de la fonction logistique. C'est une droite verticale qui peut être obtenue par les solutions de $a = 0$ et est présentée dans la figure [3.22]

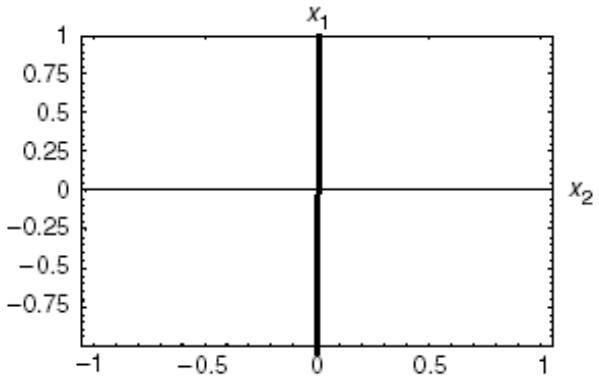

FIG. 3.22 – La ligne de frontière dans le cas $w_0 = 0, w_1 = 1, w_2 = 0$

2-$w_0 = 0, w_1 = 0, w_2 = 1$.

La ligne de frontière est dans ce cas une droite horizontale comme représentée dans la figure [3.23]

3-$w_0 = 0, w_1 = 1, w_2 = 2$

Dans ce cas , la figure de frontière est une ligne diagonale, elle divise symétriquement l'espace d'entrée comme cela est représentée dans la figure[3.24].

L'activité de neurone est plus grande en haut de la ligne et moins en dessous .

4-$w_0 = -0,5, w_1 = 1, w_2 = -1$

L'effet de la valeur de w_0 doit excentrer la ligne de frontière, dont il décale essentiellement la région de l'activité la plus élevée vers le centre, comme c'est représenté dans la figure[3.25]

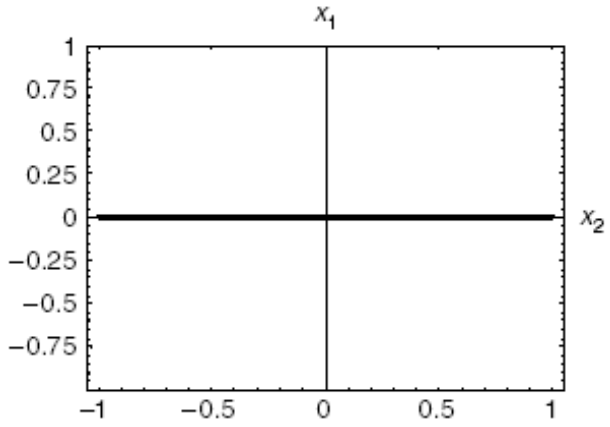

FIG. 3.23 – Dans ce cas la frontière de décision est une droite horizontale [22].

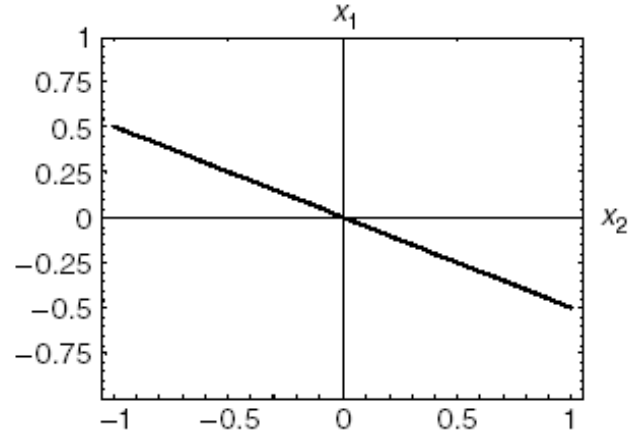

FIG. 3.24 – La ligne frontière dans le cas $w_0 = 0$, $w_1 = 1$, $w_2 = 2$ [23]

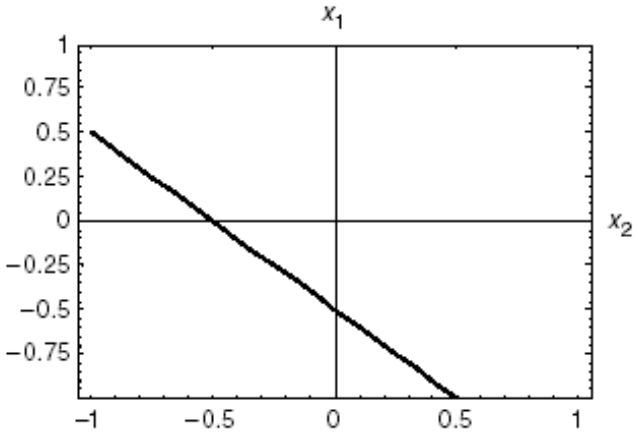

FIG. 3.25 – La ligne frontière en cas de $w_0 = -0.5$, $w_1 = 1$, $w_2 = -1$ [23]

Avec le même principe que précédement, le réseau PMC avec deux entrées approche n'importe quelle fonction bidimensionnelle (frontière de décision pour la classification).

La sortie de réseau Dans le cas de classification, les valeurs de y en dessus de 0.5 (ou autre seuil à définir par l'utilisateur) sont ajustées sur 1 classifiées en tant qu'une classe, et ceux en dessous d'elles sont ajustés sur 0 et classifiées en une autre classe.

Alors le PMC avec deux entrées est capable de classifier des individus dans l´espace bidimensionnel, et les concepts étudiés dans ce cas peuvent être prolongés á un PMC avec plusieurs entrées (des problèmes de classification multidimensionnels). Le problème est que les fonctions logistiques deviennent multidimensionnelles et ne peuvent pas donc être visualisées graphiquement. Cependant, il est possible de comprendre intuitivement tout le processus du traitement de l'information même dans ces derniers réseaux de neurones, basé sur la compréhension du processus du traitement des données des réseaux bidimensionnels. Et maintenant que le modèle de la formulation est clair, il est possible d'écrire les équations avec une facilité relative, comme il le sera montré dans la suite.

3.7.2 Le PMC avec des données multidimensionnelles

Un PMC, sous sa forme plus générale, peut avoir plusieurs neurones d'entrées et plusieurs neurones de sorties. Dans le cas de la prévision, il y a habituellement un seul neurone de sorties ; la classification multiclasse exige plus d'un. Il peut y avoir une ou plusieurs couches cachées et plusieurs neurones cachés dans chaque couche. Dans le cas général, là où il y a n entrées, M neurones cachés, et c neurones de sorties. Des étapes intermédiaires du traitement dans un PMC peuvent être construites comme suit

L'entrée de neurone caché a_j et la sortie de j'éme neurone z_j sont :

$$a_j = w_{0j} + \sum_{i=1}^{n} w_{ij} x_i \tag{3.51}$$

$$z_j = g\left(a_j\right) \tag{3.52}$$

Là où x_i est l'$i\acute{e}me$ entrée ,w_{ij} est le poids associé à l'entrée i et le neurone j,w_{oj} est le poids biais du neurone caché j et le $g\left(a_j\right)$ peut être la fonction d'activation qui transforme a_j en la sortie de neurone caché z_j

La somme nette d'entrée a_k et le y_k la sortie de $k^{éme}$ neurone de sortie peut être écrite comme suit :

$$a_k = w_{0k} + \sum_{j=1}^{M} w_{jk} z_j \tag{3.53}$$

$$y_k = g\left(a_k\right) \tag{3.54}$$

Là où M et c sont le nombre de neurones cachés et de neurones de sortie respectivement, w_{ok} est le poids biais du neurone de sortie k , w_{jk} le poids de raccordement entre le neurone caché j et le $k^{éme}$ neurone de sortie, et le $g\left(a_k\right)$ est la fonction d'activation du $k^{éme}$ neurone de sortie, qui transforme a_k vers sa sortie finale[22].

Remarque 23 *Il existe un problème médicale de discriminer les nodules pulmonaires solitaires (le malin et benin). Dans la résolution traditionnelle, il est utilisé une base statistique bayésienne. Pour la résolution de ce problème par le PMC,on extrait le tableau des données d'après le service de radiologie de CHU Constantine (présenté en annexe C), mais l'exécution de ce problème par le logiciel MATLAB version 7 n'existe pas. Dans la dernière version (v10), il considère ce problème comme un problème de reconnaissance de forme.*

L'analyse discriminante est le nom donné à la classification, dans le cadre supervisé. Le mot supervisé désigne que l'appartenance aux classes est disponible pour faire une correction d'erreur entre les sorties de réseau et les

valeurs cibles. Cette erreur sera explorée de manière assez détaillée prochainement.

Chapitre 4

La fonction d'erreur et ses dérivées

Jusqu'ici, nous avons vu que l'utilisation de perceptron ou en terme plus générale les réseaux avec une seule couche, sont beaucoup limités avec les données non linéairement séparables.

Nous mettons en lumière les perceptrons multicouches PMC, qui fournissent un cadre pratique pour la prédiction (régression et analyse discriminante) multivariable et non linéaire,

Dans ce chapitre, nous commençons d'abord par discuter les propriétés importantes d'une fonction d'erreur. Afin de minimiser cette erreur, nous discutons la technique de rétropropagation qui permet de calculer les dérivés de fonction d'erreur quadratique.

4.1 Définition de la fonction d' erreur

Pour des problèmes associatifs de prédiction, il est commode de décomposer la densité de probabilité conjointe $f(x,t)$ en produit de la densité conditionnelle de la variable, conditionnée sur l'ensemble d'entrées , et la densité sans condition des données, de sorte que

$$f(x,t) = f(t/x)f(x) \tag{4.1}$$

Où $f(t/x)$ dénote la densité de probabilité de t étant donnée que x prend une telle valeur définie, alors que $f(x)$ représente la densité sans condition de x et elle est donnée par :

$$f(x) = \int f(x,t) \ \ dt \tag{4.2}$$

Nous pouvons motiver la fonction d'erreur par le principe de maximum de vraisemblance. Pour un ensemble d'apprentissage $\{(x^n, t^n)\ \ n = 1, ..., N\}$ la vraisemblance peut être écrite comme suit :

$$L = \prod_n f(x^n, t^n) \tag{4.3}$$

$$= \prod_n f(t^n/x^n) f(x) \tag{4.4}$$

Où nous avons supposé que chaque point d'ensemble d'apprentissage (x^n, t^n) est dessiné indépendamment de la même distribution , et par conséquent nous pouvons multiplier les densités (les probabilités). Au lieu de maximiser la vraisemblance , il est généralement plus commode de réduire au minimum le logarithme négatif de la vraisemblance. . Nous réduisons au minimum E, où il est donné par

$$E = -\ln L = -\sum_n \ln f(t^n/x^n) - \sum_n f(x^n) \tag{4.5}$$

Où E s'appelle une fonction d'erreur

La deuxième terme dans (4.5) ne dépend pas des paramètres de réseau et représente ainsi une constante additive qui peut être déduite de la fonction d'erreur. Nous avons donc

$$E = -\sum_n \ln f(t^n/x^n) \tag{4.6}$$

Pour des problèmes d'interpolation , les variables cibles t sont des quantités continues, tandis que pour des problèmes de discrimination, les variables cibles t sont des quantités discrètes.

Notons que la fonction d'erreur prend la forme d'une somme sur toutes les fonctions d'erreur pour chaque point (x^n, t^n) séparément (indépendance des distribution de $(t^n, x^n)\ \ n = 1, ..., N$)

4.2 La fonction d'erreur de somme quadratique

Nous supposons que les distributions de c variables cibles t_k où $k = 1, ..., c$ sont indépendantes, de sorte que nous pouvons écrire :

$$f(t/x) = \prod_{k=1}^{c} f(t_k/x) \tag{4.7}$$

Nous supposons que la variable cible t_k est donnée par certaines fonctions déterministes de x avec le bruit gaussien supplémentaire ε, de sorte que :

$$t_k = h_k(x) + \varepsilon_k \tag{4.8}$$

Nous supposons maintenant que l'erreur ε_k a une distribution normale avec zéro comme moyenne et un écart type σ qui ne dépend pas de x ou de k. Alors, la distribution de ε_k est donné par :

$$f(\varepsilon_k) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left(\frac{-\varepsilon_k^2}{2\sigma^2}\right) \tag{4.9}$$

Nous cherchons maintenant à modeler $h_k(x)$ par un réseau de neurone avec des sorties $y_k(x, w)$ où w est l'ensemble de paramètres de poids régissant la courbe de réseau de neurone.

En employant (4.8) et (4.9), nous voyons que la distribution de probabilité des variables cibles est donnée par :

$$f(t_k/x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[-\frac{\{y_k(x, w) - t_k\}^2}{2\sigma^2}\right] \tag{4.10}$$

Où nous avons remplacé la fonction inconnue $h_k(x)$ par notre modèle $y_k(x, w)$

En même temps nous, substituons (4.7) dans (4.6) pour écrire :

$$\begin{aligned} E &= -\sum_n \ln\left(\prod_{k=1}^{c} f(t_k^n, x^n)\right) \\ &= -\sum_{n=0}^{N}\sum_{k=1}^{C} \ln f(t_k^n, x^n) \end{aligned} \tag{4.11}$$

D'après (4.10) on a :

$$\begin{aligned} \ln f(t_k^n, x^n) &= \ln\left(\frac{1}{(2\pi\sigma^2)^{1/2}}\right) - \frac{\{y_k(x^n, w) - t_k^n\}^2}{2\sigma^2} \\ &= -\frac{1}{2}\ln\ 2\pi\sigma^2 - \frac{\{y_k(x^n, w) - t_k^n\}^2}{2\sigma^2} \end{aligned} \tag{4.12}$$

Nous substituons (4.12) dans (4.11) pour obtenir la forme suivante de E

$$E = \frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{k=0}^{C}\{y_k\,(x^n, w) - t_k^n\}^2\, NC\ln\sigma + \frac{NC}{2}\ln(2\pi) \tag{4.13}$$

Dans le but de la minimisation d'erreur , les deuxièmes et les troisièmes termes du côté droite de (4.12) sont indépendants des poids w et alors, ils peuvent être omis. De même, le facteur global de$\frac{1}{\sigma^2}$ dans le premier terme peut être également omis. Alors, nous obtenons finalement l' expression utilisable pour la fonction de somme quadratique comme suit :

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}\left\{y_k\left(x^n, w\right) - t_k^2\right\}^2 \tag{4.14}$$

$$= \frac{1}{2}\sum_{n}\left\|y\left(x^n, w\right) - t^n\right\|^2 \tag{4.15}$$

Remarque 24 *Noter qu'il est parfois utile pour une execution plus commode des réseaux d'employer une fonction d'erreur différente de l' expression 4.15 qui est la racine de la moyenne quadratique (RMS) de la forme*

$$E^{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} y\left(x^n, w^*\right) - t^n} \tag{4.16}$$

Où w^* dénote le vecteur de poids du réseau traité , et les sommes sont maintenant sur N points d'ensemble de test.

4.3 Interprétation des sorties du réseau

Nous considérons le cas ou la taille N d'ensemble d'apprentissage tend vers l'infini. Dans cette limite nous pouvons remplacer la somme finie sur tous les points d'entrées dans l'erreur de somme quadratique par une intégrale de la forme

$$E = \lim_{n\longrightarrow+\infty}\frac{1}{2N}\sum_{n=1}^{N}\sum_{K}\left\{y_k\left(x^n, w\right) - t_k^n\right\}^2 \tag{4.17}$$

$$= \frac{1}{2}\sum_{k}\iint\left\{y_k\left(x, w\right) - t_k^n\right\}^2 f\left(t_k, x\right) dt_k dx$$

Où nous avons présenté un facteur supplémentaire de $\frac{1}{N}$ dans la définition de la fonction d'erreur de somme quadratique afin de faire un processus significatif de limite.

Nous utilisons l'expression (4.1) dans (4.17) pour donner

$$E = \frac{1}{2}\sum_k \iint \left(y_k\left(x,w\right) - t_k\right)^2 f(t_k/x) f(x) dt_k dx \tag{4.18}$$

Ensuite, nous définissons les moyennes conditionnelles suivantes des données cibles (moment conditionnel d'ordre un et deux)

$$\begin{aligned} \langle t_k/x\rangle &= \int t_k f(t_k/x) dt_k \\ \left\langle t_k^2/x\right\rangle &= \int t_k^2 f(t_k/x) dt_k \end{aligned} \tag{4.19}$$

Nous écrivons maintenant le terme entre parenthèse dans (4.17) sous la forme

$$\begin{aligned} \left\{y_k - t_k\right\}^2 &= \left\{y_k - \langle t_k/x\rangle + \langle t_k/x\rangle - t_k\right\}^2 \\ &= \left\{y_k - \langle t_k/x\rangle\right\}^2 + 2\left\{y_k - \langle t_k/x\rangle\right\}\left\{\langle t_k/x\rangle - t_k\right\} + \left\{\langle t_k/x\rangle - t_k\right\}^2 \end{aligned} \tag{4.20}$$

Après nous substituons(4.20) dans (4.18)

$$\begin{aligned} E &= \frac{1}{2}\sum_k \int \left\{y_k - \langle t_k/x\rangle\right\}^2 f\left(x\right) dx \\ &\quad + \frac{1}{2}\sum_k \int \left\langle \left\{\langle t_k/x\rangle - t_k^2\right\}/x\right\rangle f(x) dx \\ &= \frac{1}{2}\sum_k \int \left\{y_k - \langle t_k/x\rangle\right\}^2 f\left(x\right) dx + \frac{1}{2}\sum_k \int \left\{\left\langle t_k^2/x\right\rangle - \langle t_k/x\rangle^2\right\} f(x) dx \end{aligned} \tag{4.21}$$

Notons que le deuxième terme dans (4.21) est indépendant de la sortie de réseau $y_k\left(x,w\right)$ et par conséquent est indépendant des poids de réseau w. Dans le but de l'ajustement de poids de réseau par minimisation d'erreur , ce terme peut être négligé. Puisque la fonction à intégrer dans le premier terme dans (4.21) est non négative , le minimum absolu de la fonction erreur se produit quand ce premier terme disparaît , ce qui correspond au résultat suivant pour les sorties de réseau

$$y_k\left(x,w^*\right) = \langle t_k/x\rangle \tag{4.22}$$

Où w^* est le vecteur de poids qui minimise la fonction d'erreur

L'équation (4.22) est un résultat principal qui indique que la sortie de réseau est donnée par la moyenne conditionnelle des données de cible t_k, en d'autres termes, par la régression du t_k conditionné sur x.

4.3.1 Les conditions de ce résultat

Il y a deux conditions principales pour obtenir ce résultat qui sont :

•L' ensemble de données doit être suffisamment grand pour qu'il approche un ensemble de données infini .

•La fonction de sortie de réseau y_k doit être suffisamment générale, cela exige un choix des paramètres qui fait le premier terme de (4.21) suffisamment petit. Cette deuxième condition implique que le nombre de poids adaptatifs (ou de manière équivalente le nombre des unités cachées) doit être suffisamment grand

Il est important que les deux conditions (le grande taille de l'ensemble d'apprentissage et le grande nombre des poids) doivent être réalisées d' une manière couplée pour réaliser le résultat désiré. L'optimisation des paramètres de réseau est exécutée ainsi quand on a trouvé le minimum approprié de la fonction de coût (ces techniques seront discutées dans le chapitre 05)

Dans ce travail nous utilisons les perceptrons avec deux couches (perceptron multicouche) qui fournissent un cadre pratique pour la modélisation de la fonction multivariable non linéaire par la moyenne conditionnelle.

Pour mettre le perceptron multicouche en phase d'utilisation,nous commençons d'abord par un point important qui est l'espace de poids adaptatifs de réseau.

4.4 La symétrie d'espace de poids

Nous considerons un réseau de deux couches ayant M unités cachées avec une fonction d'activation tanh et une connexion complète dans les deux couches. Si w_{ji} le poids qui est entre l'unité i et l'unité caché j. où : $j = 1\ldots, M$ et $i = 1\ldots, n$ et w_{j0} est un vecteur de biais et w_{kj} est le poids entre l'unité caché j et l'unité de sortie k. Où $j = 1\ldots, M$ et $k = 1\ldots, c$, et w_{k0}est le deuxieme vecteur de biais .

Premièrement, si nous changeons le signe de tous les poids w_{ji} où j est fixé et $i = 1\ldots, n$ alors pour un vecteur d'entrées $(x_0, x_1, \ldots, x_n)$, le signe de l'activation de l'unité j sera inversée. Comme la fonction tanh est une fonction impaire, ceci peut être compensé par le changement du signe de tous les poids w_{kj} pour cette unité j et $k = 1\ldots, c$, alors la sortie qui présente la relation entre les variables d'entrées et les variables de sorties par le réseau n'est pas changée. Nous obtenons alors deux vecteurs différents de poids qui provoquent la même sortie, il y a ainsi un ensemble des vecteurs de poids équivalents de $2M$, où M est le nombre d'unités cachées.

Deuxièmement, imaginez que nous échangeons les valeurs de tous les poids

w_{ij} et w_{jk} où j est fixé (pour unité cachée fixée) et nous échangeons les valeurs de poids (et biais) liés à une unité cachée différente de j, Encore une fois, la sortie de réseau n'est pas changée. Alors pour M unités cachées, un vecteur de poids aura $M!$ vecteurs de poids équivalents.

D'après le premier et le deuxième raisonnement pour un réseau de M unités cachées, il y a $M!2M$ facteurs de symétrie dans l'espace de poids.

Remarque 25 *L'existence de ces symétries n'est pas une propriété de la fonction "tanh" seulement mais s'applique pour plusieurs fonctions d'activation différentes.*

Remarque 26 *Dans plusieurs cas, ces symétries, dans l'espace de poids ont peu de conséquences pratiques.*

4.5 Rétropropagation d'erreur

Jusqu'ici dans cette partie, nous avons mis l'accent sur les possibilités de représentation des réseaux multicouches. Nous considérerons après comment un tel réseau peut apprendre une fonction appropriée d'ensemble d'entrées. Comme dans le paragraphe précédent l'apprentissage sera basée sur la définition d'une fonction appropriée qui est alors réduite au minimum par rapport aux poids et aux biais dans le réseau.

Si nous considérons un réseau avec des fonctions d'activation différentiables alors les activations des unités de sorties deviennent des fonctions différentiables par rapport aux variables d'entrées et aux poids (et aux biais).

Si nous définissons une fonction d'erreur telle que la somme quadratique d'erreur présentée dans le paragraphe *(4.2)* qui est une fonction différentiable par rapport aux sorties de réseau $(y_k(x))$, alors cette fonction d'erreur elle même est différentiable par rapport aux poids (et les biais). Nous pouvons donc évaluer les dérivés de fonction d'erreur par rapport aux poids, et ces dérivés peuvent alors être employés pour trouver les valeurs de poids qui réduisent la fonction d'erreur au minimum.

Pour la minimisation d'une fonction d'erreur et les ajustements des poids de réseau, nous pouvons distinguer deux étapes différentes. La première est l'algorithme de Retropropagation et la deuxième est le calcul d'ajustement de poids.

4.5.1 Définition de la rétropropagation d'erreur

Définition 27 *La retropropagation d'erreur est une technique pour évaluer les dérivés de la fonction d'erreur, cela correspond à une propagation des*

erreurs vers l'arrière par le réseau et fournit un information efficace pour évaluer les dérivés de la fonction d'erreur par rapport aux poids de réseau.

Remarque 28 *Le mot retropropagation dans la littérature du calcul neuronal signifie une variété de différentes choses, par exemple,*

$\triangleright$*L'architecture de perceptron multicouche s'appelle parfois un réseau à Retropropagation*

$\triangleright$ *Employer pour décrire la formation d'un perceptron multicouche employant la déscente de gradient et une fonction d'erreur de somme quadratique.*

Remarque 29 *En général, cette technique comprend des procédés itératifs.*

4.5.2 La procédure de rétropropagation

- Supposons une architecture de réseau pondéré ayant une seule couche cachée des unités avec une fonction d'activation sigmoïde (logistique), et considérons une fonction d'erreur de somme quadratique.
- Dans chaque unité de la couche cachée, il y a une transformation de somme pondérée d'entrées de la forme :

$$z_j = g(a_j) \tag{4.23}$$

Où

$$a_j = \sum_i w_{ij} x_i$$

Où $x_i, i = 1 \ldots, N$ l'entrée qui envoie une connexion à l'unité j des réseaux ; g une fonction (logistique). Et w_{ji} est le poids associé à ces connexions.

- Nous avons considérés une fonction d'erreur E qui est différentiable par rapport à $y_k$et par rapport aux poids de réseau (et biais). Alors il est différentiable par rapport à y_k et par rapport aux poids. Nous pouvons écrire E comme somme sur tous les n points d'ensemble d'entrée. Alors

$$E(w) = \sum_n E^n(y_k(w.x^n).t^n) \qquad\qquad k = 1..., c \tag{4.24}$$

Et dans les unités de sortie.

$$y_k(x) = g(a_k)$$

Où

$$a_k = \sum_j w_{kj} z_j \tag{4.25}$$

Considérons maintenant l'évaluation de dérivé de E^n par rapport àw_{ji} :

$$\frac{\partial E^n}{\partial w_{ji}} = \frac{\partial E^n}{\partial a_j} . \frac{\partial a_j}{\partial w_{ji}} \tag{4.26}$$

Nous notons δ_j :la dérivé de E^n par rapport à a_j
On écrit :

$$\frac{\partial E^n}{\partial a_j} = \delta_j \tag{4.27}$$

- δ_j sont souvent mentionnés comme des erreurs.
- Alors pour les unités de sortie nous notons δ_k la dérivé de E^npar rapport à a_k

$$\frac{\partial E^n}{\partial a_k} \equiv \delta_k \tag{4.28}$$

Pour évaluer δ_j pour les unités cachées j ,utilisons la règle de chaînes pour les dérivés partielles, ou la somme est sur toutes les unités k auxquelles l'unité j envoie des connexions.
D'après (4.27) nous pouvons écrire :

$$\delta_j = \sum_k \frac{\partial E^n}{\partial a_k} . \frac{\partial a_k}{\partial a_j} \tag{4.29}$$

D'après (4.23) et (4.25) nous avons la formule suivante :

$$a_k = \sum_k w_{kj} g(a_j)$$

Alors,

$$\frac{\partial a_k}{\partial a_j} = \quad w_{kj} g'(a_j) \tag{4.30}$$

Si nous substituons (4.30) et (4.28) dans (4.29) nous obtenons :

$$\delta_j = g'(a_j) \sum_k w_{kj} . \delta_k \tag{4.31}$$

Ce qui nous indique que la valeur δ_j pour une unité cachée particulière j peut être obtenue pour propager les δ_k vers l'arrière à partir des unités plus haut vers des unités plus bas dans le réseau. Comme c'est illustré dans la figure (4.1)

.

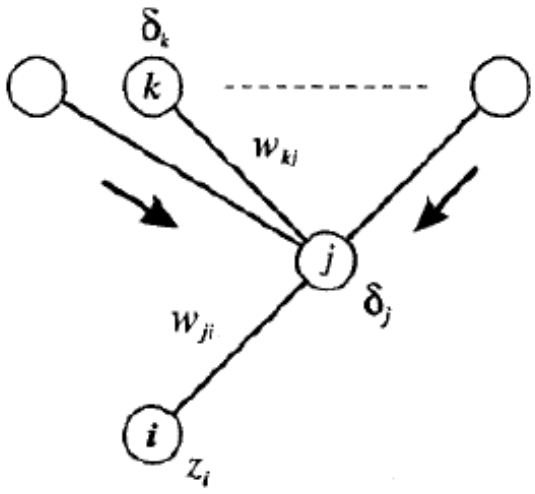

FIG. 4.1 – La procédure de rétropropagation δ_k à partir de ces unités k auxquelles l'unité j envoie des connexions.

D'après (4.28) :

$$\delta_k \equiv \frac{\partial E^n}{\partial a_k} = \frac{\partial y_k}{\partial a_k} . \frac{\partial E^n}{\partial y_k} \tag{4.32}$$

et nous avons d'après (4.25) :

$$y_k = g(a_k)$$

alors :

$$\frac{\partial y_k}{\partial a_k} = g'(a_k) \tag{4.33}$$

Nous substituons (4.33) dans (4.32), nous obtenons la formule suivante :

$$\delta_k = g'(a_k) \frac{\partial E^n}{\partial y_k} \tag{4.34}$$

Remarque 30 *Dans cette démonstration nous supposons que* $g \equiv \breve{g}$ *pour simplifier les calculs, et nous pouvons généraliser le résultat.*

Nous pouvons récapituler la procédure de rétropropagation en quatre étapes :

1)Mettre à jour le vecteur d'entrée x^n dans le réseau et propager en avant et trouver les activations de toutes les unités cachées et les unités de sorties.

2)Evaluer le δ_k , $k = 1 \ldots, c$, en utilisant (4.34) .

3)En utilisant (4.31) pour calculer $\delta_j, j = 1 \ldots, M$.

4)Enfin, nous avons les dérivés de E^n par rapport $w_{kj}, j = 1 \ldots, M$ et $k = 1 \ldots, c$ par l'utilisation de (4.26).

Si nous répétons ce précédé pour chaque x^n où $n = 1 \ldots, N$ et d'après $\left[\frac{\partial E^n}{\partial w_{ji}} = \sum_n \frac{\partial E^n}{\partial w_{ji}}\right]$ nous avons le résultat.

La dérivation du procédé de retropropagation a tenu compte de la forme générale de la fonction d'erreur et des fonctions d'activation. Afin d'illustrer l'application de cette technique, nous prenons des exemples simples particuliers.

4.5.3 L'application de rétropropagation

L'application de retropropagation à un réseau avec une seule couche

Dans ce cas, et afin d'illustrer l'application de cette technique, nous considérons la forme de la descente de gradient donnée par (3.2) et nous utilisons la fonction d'activation linéaire (un réseau linéaire) donnée par.

$$a_k(x) = \sum_{j=1}^{N} w_{kj} x_j \tag{4.35}$$

Alors les dérivés de la fonction d'erreur quadratique sont données par

$$\frac{\partial E^n}{\partial w_{kj}} = \{y_k(x^n) - t_k^n\} x_j^n = \delta_k^n x_j^n \tag{4.36}$$

Où

$$\delta_k^n = \{y_k(x^n) - t_k^n\} \tag{4.37}$$

En substituant (4.36) dans (4.35) nous obtenons l'équation suivante ;

$$w_{kj}^{(\tau+1)} = w^{(\tau)} - \eta \delta_k^n x_j^n \tag{4.38}$$

Et nous pouvons conclure que le chargement de poids est donné par :

$$\triangle w_{kj} = -\eta \delta_k^n x_j^n. \tag{4.39}$$

L'application de retropropagation à un réseau PMC

Nous fixons l'architecture de réseau à deux couches de poids adaptatif. Les fonctions d'activation des unités de la couche cachée sont des fonctions logistiques et les fonctions d'activation des unités de sorties de réseau sont linéaires. Nous utilisons la fonction d'erreur de somme quadratique, qui satisfait la propriété suivante $E = \sum_n E^n$ pour un vecteur d'entrée x^n

Où E^n donnée par

$$E^n = \frac{1}{2}\sum_{k=1}^{c}(y_k - t_k)^2 \tag{4.40}$$

Où y_k représente la sortie de l'unité de sortie k, t_k est la valeur cible correspondante à l'unité k.

En appliquant la première étape dans la procédure de retropropagation, alors il résulte .

$$a_j = \sum_i w_{ij}x_i$$

$$a_k = \sum w_{kj}z_j$$

$$z_j = g(a_j)$$

Où

$$g(x) = \frac{1}{1+\exp(-x)} \tag{4.41}$$

D'après la deuxième

$$\delta_k = y_k - t_k$$

Et on a

$$\delta_j = g'(a_j)\ \sum_{k=1}^{n} w_{kj}.\delta_k$$

où

$$g'(a) = g(a)(1 - g(a))$$

et

$$g(a_j) = z_j$$

Alors,

$$\delta_j = z_j(1 - z_j) \sum_{k=1}^{n} w_{kj}.\delta_k \tag{4.42}$$

Les dérivés par rapport aux poids de la première couche sont indiquées par :

$$\frac{\partial E^n}{\partial w_{ji}} = \delta_j x_i \tag{4.43}$$

Et les dérivés par rapport aux poids de la deuxième couche sont indiquées par :

$$\frac{\partial E^n}{\partial w_{kj}} = \delta_k z_j \tag{4.44}$$

La matrice de Jacobi

Ici, nous considérons l'évaluation de la matrice de **JACOBI**, dont ses éléments sont donnés par les dérivés des sorties de réseau par rapport aux entrées, telle que chaque dérivé est évaluée avec toutes les autres entrées en les supposant fixées.

$$J_{kj} = \frac{\partial y_k}{\partial x_i} \tag{4.45}$$

. L'utilisation dans la théorie :

* La matrice *JACOBIENNE* est parfois employée pour décrire les dérivés de la fonction d'erreur par rapport aux poids de réseau de retropropagation.

* Elle fournit une mesure locale de la sensibilité des sorties aux changements de chacune de variables d'entrées.

* Elle est utilisée dans plusieurs contextes dans l'application des réseaux de Neurones.

* Elle peut être évaluée en utilisant un procédé de retropropagation, ce qui est très semblable à ce qui a été est décrit plus haut pour évaluer les dérivés d'une fonction d'erreur par rapport aux poids (biais). Nous commençons par l'élément J_{ki} sous la forme :

$$J_{ki} = \frac{\partial y_k}{\partial x_i} = \sum_j \frac{\partial y_k}{\partial a_j} . \frac{\partial a_j}{\partial x_i} = \sum_j w_{ji} . \frac{\partial y_k}{\partial a_j} \tag{4.46}$$

Car $g_j = \sum w_{ij} . x_i$ où l'unité d'entrée i envoie une connexion à toutes les unités j de couche cachée.

Nous notons maintenant une formule récursive de retropropagation pour déterminer $\frac{\partial y_k}{\partial a_j}$.

$$\frac{\partial y_k}{\partial a_j} = \sum_k \frac{\partial y_k}{\partial a_k} . \frac{\partial a_k}{\partial a_j} = g'(a_j) \sum_k w_{kj} . \frac{\partial y_k}{\partial a_k} \tag{4.47}$$

puisque

$$a_k = \sum_k w_{kj} g(a_j)$$

et nous avons $g_k = g(a_k)$ alors,

$$\frac{\partial y_k}{\partial a_k} = g'(a_k) \tag{4.48}$$

Nous appliquons maintenant la procédure de rétropropagation comme suit :

1/On applique le vecteur d'entrée correspondant au point d'espace d'entrée à ce que la matrice de *JACOBI* doit être trouvée, et on applique une propagation vers l'avant de manière habituelle afin d'obtenir les activations de toutes les couche cachées et des neurones de sorties dans le réseau.

2/ On calcule les dérivés de sortie de réseau par apport a_k d'après (4.46)

3/ On substitue le résultat de (4.48) dans (4.47) pour trouver les dérivés dey_k par apport a_j

4/On utilise le résultat de (4.47) pour obtenir J_{kj} avec l'expression (4.46).

La matrice de HESSIEN

Nous avons montré comment la technique de retropropagation peut être employée pour obtenir les premiers dérivés d'une fonction d'erreur par rapport aux poids dans le réseau.

Cette technique peut être employée pour évaluer les deuxièmes dérivés de la fonction d'erreur donnée par :

$$\frac{\partial^2 E}{\partial w_{ji}\, \partial w_{\varrho k}} \tag{4.49}$$

Ces dérivés forment les éléments de la matrice de *HESSIEN*, qui joue un rôle important dans le calcul neuronal. Dans ce travail, nous utilisons cette matrice pour évaluer plusieurs algorithmes d'optimisation utilisée pour les réseaux de neurones. Pour une application importante de la matrice de HESSIEN, on peut être évaluer les premières derivées de la fonction d'erreur par un calcul exact, par utilisation d'une prolongation de la technique de rétropropagation.

Comme dans ce qui précède , nous prenons un modéle à la fois. Considérons l'expression générale pour la dérivé de la fonction d'erreur par rapport à $w_{\varrho k}$

D'après (4.44)

$$\frac{\partial E^n}{\partial w_{\varrho k}} = \delta_\varrho z_k \tag{4.50}$$

Différenciement à ceci par rapport à un autre poids w_{ij},nous obtenons :

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{\varrho k}} = \frac{\partial a_j}{\partial w_{ji}} . \frac{\partial}{\partial a_j} \left(\frac{\partial E^n}{\partial w_{\varrho k}}\right) = z_j \frac{\partial}{\partial a_j}\left(\frac{\partial E^n}{\partial w_{\varrho k}}\right) \tag{4.51}$$

D'après(4.50), nous pouvons écrire :

$$\begin{aligned} \frac{\partial}{\partial a_j}\left(\frac{\partial E^n}{\partial w_{\varrho k}}\right) &= \frac{\partial}{\partial a_j}(\delta_\varrho z_k) \\ &= \frac{\partial z_k}{\partial a_j}.\delta_\varrho + \frac{\partial \delta_\varrho}{\partial a_j}.z_k \end{aligned} \tag{4.52}$$

Et nous avons :

$$z_k = g(a_k)$$

alors,

$$\begin{aligned} \frac{\partial z_k}{\partial a_j} &= \frac{\partial g(a_k)}{\partial a_k}. \frac{\partial a_k}{\partial a_j} \\ &= g'(a_k). \frac{\partial a_k}{\partial a_j} \end{aligned} \tag{4.53}$$

Nous pouvons écrire (4.52) sous la forme :

$$\frac{\partial}{\partial a_j}(\frac{\partial E^n}{\partial w_{\varrho k}}) = \delta_\varrho \; g'(a_k). \frac{\partial a_k}{\partial a_j} + z_k \; \frac{\partial \delta_\varrho}{\partial a_j} \tag{4.54}$$

Nous substituons (4.54) dans l'expression (4.51), nous obtenons :

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{\varrho k}} = z_j \delta_\varrho g'(a_k) \frac{\partial \; a_k}{\partial a_j} + z_j z_k \frac{\partial \delta_\varrho}{\partial a_j} \tag{4.55}$$

Nous pouvons considérer la notation suivante :

$$h_{kj} \equiv \frac{\partial \; a_k}{\partial a_j} \tag{4.56}$$

Et

$$b_{\varrho j} \equiv \frac{\partial \delta_\varrho}{\partial a_j} \tag{4.57}$$

Les quantités h_{kj} peuvent être évaluées par la propagation vers l'avant ainsi : En utilisant la règle de chaine pour les dérivés partielless, nous aurons

$$h_{kj} = \sum_r \frac{\partial \; a_k}{\partial a_r}. \frac{\partial \; a_r}{\partial a_j} \tag{4.58}$$

Où la somme ici, est sur toutes les unités r qui envoient les connexions à l'unité k

D'après (4.56), nous avons

$$\frac{\partial \; a_r}{\partial a_j} \equiv h_{rj} \tag{4.59}$$

Et nous avons

$$a_k = \sum_r w_{kr} g'(a_r) \tag{4.60}$$

Alors :

$$\frac{\partial\ a_k}{\partial a_r} = w_{kr} g'(a_r) \tag{4.61}$$

Nous substituons (4.61) et (4.59) , nous pouvons écrire (4.58) sous la forme :

$$h_{kj} = \sum_r g'(a_r) w_{kr} h_{rj} \tag{4.62}$$

Remarque 31 *Les conditions initiales pour évaluer* $\{h_{kj}\}$ *suivant l'expresion(4.56) peuvent s'énoncer comme suit pour chaque unité* j *dans le réseau.*

$$\left\{ \begin{array}{llll} Si & k = j & alors & h_{kj} = 1 \\ Si & k \neq j & alors & h_{kj} = 0 \end{array} \right.$$

(car il n'y a aucune connexion directe entre$j$ *et* k *)*

Alors les éléments restants de h_{kj} peuvent être trouvés par l'équation (4.62), nous pouvons évaluer $\{h_{kj}\}$ pour la rétropropagation comme suit :

D'aprés (4.62), nous avons vu que :

$$\delta_\varrho = g'(a_\varrho) \sum_s w_{s\varrho} \delta_s \tag{4.63}$$

En substituant (4.63) dans la définition $b_{\varrho j}$ dans (4.57), nous obtenons :

$$\begin{aligned} b_{\varrho j} &= \frac{\partial}{\partial a_j} \left\{ g'(a_\varrho) \sum_s w_{s\varrho} \delta_s \right\} \qquad (4.64) \\ &= \frac{\partial}{\partial a_j} g'(a_\varrho) . \sum_s w_{s\varrho} \delta_s + g'(a_\varrho) . \frac{\partial}{\partial a_j} \left\{ \sum_s w_{s\varrho} \delta_s \right\} \\ &= \frac{\partial g'(a_\varrho)}{\partial a_\varrho} . \frac{\partial a_\varrho}{\partial a_j} \left(\sum_s w_{s\varrho} \delta_s \right) + g'(a_\varrho) \sum_s w_{s\varrho} \frac{\partial \delta_s}{\partial a_j} \\ b_{\varrho j} &= g^{"}(a_\varrho) h_{ij} \sum_s w_{s\varrho} \delta_s + g'(a_\varrho) \sum_s w_{s\varrho} b_{sj} \end{aligned}$$

Ou nous faisons la somme sur toutes les unités s auxquelles l'unité ϱ envoie des connexions.

4.5.4 L'efficacité de la rétropropagation

Un des aspects les plus importants de cette technique est son efficacité informatique. Pour cela nous supposons que w indique tout le nombre de poids et biais dans le réseau ; pour un w suffisamment grand :

1/ l'évaluation simple de la fonction d'erreur (pour les x^n) exigerait $O(w)$ opérations.

2/ Si le réseau est à connexion complète, le nombre de poids est typiquement beaucoup plus grand que le nombre d'unités. Alors, il y a un effort informatique pour calculer z_j et y_k.

3/Le calcul de chaque z_j exige une multiplication et une addition alors il y a un coût informatique global qui est $O(w)$.

4/Pour w (poids totale dans le réseau), il y a w dérivés à évaluer.

5/ Pour une telle expression de la fonction d'erreur et les formes explicites des dérivés, nous l' évaluons par propagation vers l'avant pour chaque élément de w (pour chaque poids ou biais) exigeant $O(w)$opérations. Alors évaluer toutes les dérivés, demande $O(w^2)$ opérations.

Par comparaisons, si nous utilisons la retropropagation ;

▶La phase de propagation avant demande $O(w)$opérations.

▶L'évaluation des dérivés par la technique de retropropagation a ramené la complexité informatique de $O(w^2)$ à $O(w)$ pour chaque vecteur d'entrée. Alors pour cette raison, l'utilisation de retropropagation dans un apprentissage dans les réseaux multicouches consiste a consommer beaucoup de temps, ainsi on a une efficacité cruciale.

La technique de retropropagation peut être appliqué à beaucoup d'autres genres de réseau et pas seulement le perceptron multicouche. Malgré l'efficacité et l'application large de cette technique, elle possède des limitations spécialement au niveau du plan pratique.

Afin de faire une exécution pratique de résolution du problème prédictif multidimensionnel complexe, nous allons détaillé des algorithmes d'optimisation de poids adaptatifs dans ce qui suit.

Chapitre 5

Les algorithmes d'optimisation des poids adaptatifs

Le problème de l'étude dans les réseaux de neurone a été formulé en termes de minimisation de la fonction d'erreur E . Cette erreur est une fonction des paramètres adaptatifs (poids et biais) dans le réseau , que nous pouvons regrouper dans un seul vecteur avec les composants soit $w_1, w_2, ..., w_W$ .Jusqu' ici nous avons montré que pour un perceptron multicouche , les dérivés d'une fonction d'erreur par rapport aux paramètres (poids et biais) de réseau peuvent être obtenues de manière informatique efficace en utilisant la technique de retropropagation .

Dans ce chapitre, nous passerons en revue plusieurs algorithmes, les plus importants en pratique. Le plus simple de ces derniers est la déscente de gradient qui a été décrite brièvement dans le chapitre précédent.

Il est impossible de recommander un algorithme universel simple d'optimisation. Au lieu de cela, nous accentuons les avantages et les limitations relatives aux différents algorithmes .

5.1 La surface d'erreur

Il est utile d'avoir une image géométrique simple de processus de minimisation d'erreur qui peut être obtenue en regardant $E(w)$ comme surface d'erreur dans l' espace de poids , comme c'est représenté dans la figure(5.1)

Pour des réseaux ayant deux couches de poids adaptatifs , la fonction d'erreur sera typiquement une fonction non linéaire de poids , il peut existe aux moins un minimum qui satisfait :

$$\nabla E = 0 \tag{5.1}$$

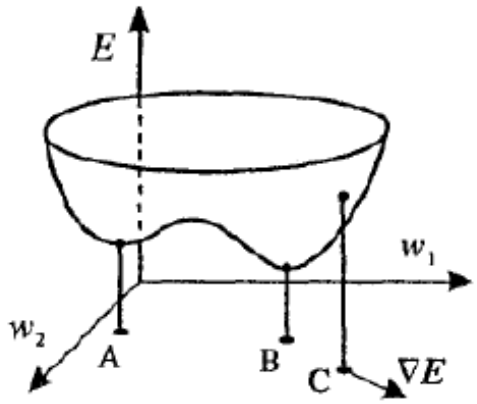

FIG. 5.1 – À un point quelconque C, le gradient local de la surface d'erreur est donné par le vecteur ∇E [6].

Où ∇E dénote le gradient de E dans l'espace de poids

Nous discutons l'utilisation de cette notation :

A cause de la non linéarité de la fonction d'erreur, il est impossible de trouver en générale la solution de forme fermée pour le minimum . Pour cela nous considérons des algorithmes qui sont impliqués dans la recherche de l'espace de poids se composant d'une succession d'étape de la forme

$$w^{(\tau+1)} = w^{(\tau)} + \triangle\, w^{(\tau)} \tag{5.2}$$

Où τ dénote l'étape d'itération.

Les différents algorithmes impliquent différents choix de $\triangle\, w^{(\tau)}$ de vecteur de poids et les choix des poids initiaux pour l'algorithme détermine vers quel minimum il convergera .

Remarque 32 *Il peut y avoir plusieurs points stationnaire ,ces points satisfont la condition (5.1),par exemple des maximum globalux , et minimum locaux (voir figure 5.02)*

Remarque 33 *Nous avons vu que dans un réseau de deux couches de poids adaptatif avec M unités dans la couche cachée , il y a un facteur de symétrie de $M!2^M$,alors n'importe quel minimum local ou global sera répété un grand nombre de fois dans tout l'espace de poids .*

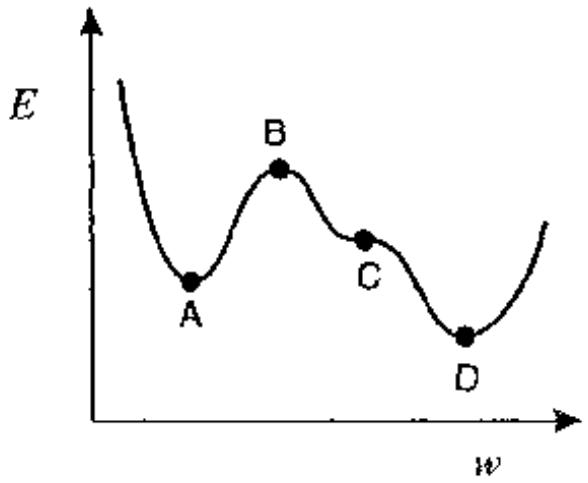

FIG. 5.2 – ∇E dénote le gradient de E dans l'espace de poids w [6]

5.2 L'approximation quadratique locale

Pour des techniques divers, le problème d'optimisation peut être obtenu en considérant une équation d'approximation quadratique locale de la fonction d'erreur.Nous considérons l'expression de TAYLOR pour $E(w)$ autour d'un certain point dans l'espace de poids.

$$E(w) = E(\hat{w}) + (w - \hat{w})b + \frac{1}{2}(w - \hat{w})H(w - \hat{w}) \tag{5.3}$$

Où b est défini pour être le gradient de E évalué à $\hat{w}$:

$$b = \nabla E \mid_{\hat{w}} \tag{5.4}$$

Et la matrice de HESSIEN H est définie par :

$$(H)_{ij} = \frac{\partial^2 E}{\partial w_i . \partial w_j} \mid_{\hat{w}} \tag{5.5}$$

D'après (5.3) l'approximation locale correspondante pour le gradient est donnée par :

$$\nabla E = \nabla E(\hat{w}) + (w - \hat{w})H$$

$$\nabla E = b + H(w - \hat{w}) \tag{5.6}$$

Ces expressions forment une base pour une grande partie de la discussion sur les algorithmes d'optimisation.

Si nous supposons que$\hat{w} = w^*$ où w^* est le point qui minimise $E(w)$. Alors, $\nabla E \mid_{w^*} = 0$ et (5.3) devient :

$$E(w) = E(w^*) + \frac{1}{2}(w - w^*)H(w - w^*) \tag{5.7}$$

Où H est évaluée au point w^*

Afin d'interpréter ceci géométriquement, nous considérons l'équation de la valeur propre pour la matrice de HESSIEN suivante :

$$H\mu_i = \lambda_i \mu_i \tag{5.8}$$

Où le vecteur μ_i forme un ensemble orthogonale complet, Alors :

$$\mu_i^T \mu_j = \delta_{ij} \tag{5.9}$$

Où δ_{ij} est le symbole de *KRONEKER* $\delta_{ij} = \begin{cases} 1 & si \;\; j = i \\ 0 & si \;\; j \neq i \end{cases}$

Alors nous pouvons écrire $(w - w^*)$ comme combinaison linéaire des vecteurs propres sous la forme :

$$w - w^* = \sum_i \alpha_i \mu_i \tag{5.10}$$

En substituant (5.10) dans (5.7), nous obtenons :

$$\begin{aligned} E(w) &= E(w^*) + \frac{1}{2}(\alpha\mu)^T H \alpha\mu \\ &= E(w^*) + \frac{1}{2}\sum_i \alpha_i^2 \mu_i^T H \mu_i \end{aligned} \tag{5.11}$$

En utilisant (5.8) dans (5.11) nous obtenons :

$$E(w) = E(w^*) + \frac{1}{2}\sum_i \alpha_i^2 \lambda_i \mu_i^T \mu_i \tag{5.12}$$

D'après (5.9), l'expression (5.12) devient :

$$E(w) = E(w^*) + \frac{1}{2}\sum_i \lambda_i \alpha_i^2 \tag{5.13}$$

5.2.1 L'interprétation géométrique

L'expression (5.10) peut être considérée comme une transformation du système de coordonnés dans laquelle l'origine estet au point w^* , et les axes sont les vecteurs propres (alors la matrice est orthogonale dont les colonnes sont les μ_i)

La convergence au point stationnaire w^* sera minimum si tou les λ_i sont positifs,où λ_i est la valeur propre de H, en effet :

Nous avons ,dans l'espace unidimensionnel, un point stationnaire qui sera un minimum si :

$$\frac{\partial^2 E}{\partial w} \mid_{w^*} > 0 \tag{5.14}$$

Et le résultat correspondant dans l'espace de d-dimension est que la matrice H (de HESSIEN) évaluée à w^* devrait être définie positive.

Ensuite,pour que les vecteurs propres $\{\mu_i\}$ forment un ensemble complet, soit un vecteur arbitraire V, nous pouvons écrire V sous la forme :

$$V = \sum_i \beta_i \mu \tag{5.15}$$

D'après (5.8) et (5.9), nous avons alors :

$$VHV = \sum_i \beta_i^2 \lambda_i \tag{5.16}$$

Alors, H sera définie positive si $\lambda_i > 0, \forall i$.

Les contours de E sont des ellipses centrées à l'origine, dont les axes sont les valeurs propres et dont les longueurs sont l'inverse de la racine des valeurs propres, comme c'est indiqué dans le schéma (5.3).

5.3 Descente du gradient

Un des algorithmes de traitement de réseau les plus simples est la descente de gradient[16]. Nous commençons avec une conjecture initiale pour le vecteur de poids (qui est choisi au hasard) et noté par w_0.Nous mettons à

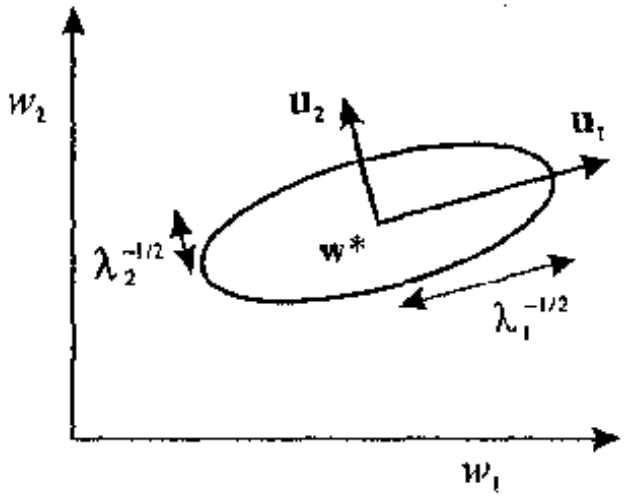

FIG. 5.3 – Les longueurs des axes sont l'inverse proportionnelles aux racines carrées des vecteurs propres correspondants λ_i [6].

jour alors itérativement le vecteur de poids tel que, à l'étape τ nous déplaçons une distance courte dans la direction du plus grand taux de diminution de l'erreur c'est-à -dire. dans la direction du gradient négatif évaluée à $w^{(\tau)}$:

$$\nabla w^{(\tau)} = -\eta \nabla E \mid_{w^{(\tau)}} \tag{5.17}$$

Où η s'appelle le taux d'apprentissage. On note que le gradient est évalué à chaque étape τ.Onl peut évaluer le gradient de la fonction d'erreur juste en un point x^n dans l'ensemble de données de N points , et les poids sont mis à jour en utilisant :

$$\Delta w^{(\tau)} = -\eta \Delta E^n \mid_{w^{(\tau)}} \tag{5.18}$$

Pour la mise à jour séquentielle de cette expression, nous pourrons espérer une réduction régulière par l'erreur puisque pour η suffisamment petit, la direction moyenne de mouvement dans l'espace de poids devrait approcher le négatif de gradient local. Si le paramètre de taux d'étude est telle qu'il diminue à chaque étape de l'algorithme selon les conditions de théorème (LUO 1991), celle ci peut être satisfait en choisissant :

$$x^{(\tau)} \alpha \frac{1}{\tau} \tag{5.19}$$

L'analogie devient précise, et nous sommes assurés de la convergence.

Ce choix entraine que la convergence est très lente. Alors nous supposons que η a une valeur fixée.

D'après (5.10)

$$\nabla E = H(w - w^*) \tag{5.20}$$

En utilisant (5.12) l'équation (5.20) devient :

$$\nabla E = \sum_i \alpha_i \mu_i H \tag{5.21}$$

Et avec l'utilisation (5.11)

$$\nabla E = \sum_i \alpha_i \lambda_i \mu_i \tag{5.22}$$

Nous avons l'expression (5.13)

$$\begin{aligned} \Delta w &= (w_1 - w^*) - (w_2 - w^*) \\ &= \sum_i \alpha_{i1} \mu_i - \sum_i \alpha_{i2} \mu_i \\ &= \sum_i \Delta\alpha_i \mu_i \end{aligned} \tag{5.23}$$

D'après (5.22) et (5.23), nous pouvons écrire (5.18) sous la forme :

$$\sum_i \Delta\alpha_i \mu_i = -\eta \sum_i \Delta\alpha_i \lambda_i \mu_i$$

Alors :

$$\Delta\alpha_i = -\eta \alpha_i \mu_i$$

Donc :

$$\alpha^{nouveau} = (1 - \eta\lambda_i)\alpha_i^{\in ncien} \tag{5.24}$$

En employant (5.13),nous avons,

$$w - w^* = \sum_i \alpha_i \mu_i$$

Nous multiplions (5.13) parμ_i^T, nous obtenons :

$$\alpha_i = \mu_i^T(w - w^*) \tag{5.25}$$

Alors α_i peut etre interprété comme distance minimum le long de la direction μ_i. D'après (5.24), nous voyons que ces distances sont évaluées indépendamment tel que, à chaque étape, la distance le long de la direction de μ_i est multiplié par un facteur $(1 - \eta\lambda_i)$.

Après un nombre d'étape total T, nous avons :

$$\alpha_i^{(T)} = (1 - \eta\lambda_i)^T \alpha_i^{(0)} \tag{5.26}$$

*Si $|1 - \eta\lambda_i| < 1 et T \to +\infty$. alors $\alpha_i = 0$ et d'après (5.25) $w = w^*$ (le vecteur de poids atteint le minimum de l'erreur)

*Si nous rendons η plus grand nous pouvons rendre le facteur $(1 - \eta\lambda_i)$ plus petit et par conséquent la vitesse de convergence sera amélioré.

*Si $|1 - \eta\lambda_i| > 1$, $\alpha_i$divergera ceci limite la valeur de η à $\eta < 2/\lambda$ max où λ max est la plus grandes des valeurs propres et $\eta > \lambda$ min la plus petite valeur propre.

Si η est placée à sa plus grande valeur autorisée, alors la convergence le long de direction correspondant à la plus petite valeur propre sera régie par $\left(1 - \frac{2\lambda_{\min}}{\lambda_{\max}}\right)$

5.3.1 Les avantages de cet algorithme

Il y a deux avantages principaux de cet algorithme :

∘Le premier c'est qu'il est classé parmi les méthodes les plus importantes pour l'approche séquentielle.

∘∘Le deuxième avantage c'est que cet algorithme a employé l'information de gradient.

5.3.2 Les inconvénients (limitation d'algorithme)

1)Dans la pratique, une valeur utilisé de η constante mene, généralement, à améliorer les résultats quoique la garantie de la convergence est perdue.

2)Il y a une difficulté grave avec le fixage de η car

·Si η est trop grand, l'algorithme peut entrainer à une augmentation de E et probablement aux oscillations divergentes ayant pour résultat une panne complète dans l'algorithme.

·Si η est choisi très petit, la recherche peut etre extrémement lente (le temps de calcul est très long)

3)La courbe de E change de manière significative avec la direction. Dans la plupart des points sur la surface d'erreur, le gradient local ne se dirige pas vers le minimum. Alors, la descente de gradient prend beaucoup de petites mesures pour atteindre le minimum, et c'est donc clairement un procédé totalement inefficace. Comme c'est présenté dans le schéma *(5.04)* pour un espace de poids bidimensionnel.

4)Si le rapport $\lambda_{min}/\lambda_{max}$ est très petit, alors le progrès vers le minimum sera extrêmement lent. Comme c'est présenté dans le schéma *(5.04)*

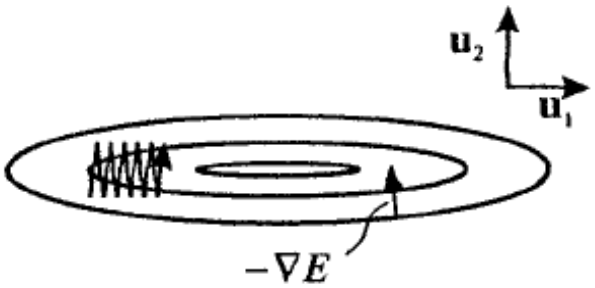

FIG. 5.4 – Les étapes successives de la descente de gradient peuvent être converge vers le minimum très lentement [6]

5)Nous pourrions trouver le minimum employé exactement comme évaluation de la fonction d'erreur $w(w+3)/2$ étapes **.** La descente de gradient est un algorithme inefficace, car le nombre des évaluations de fonction d'erreur peuvent facilement être beaucoup plus grandes que ce nombre.

Il y a eu plusieurs tentatives ces dernières années pour améliorer l'exécution de descente de gradient pour la formation de réseau de neurones,en faisant les diverses modifications suivantes :

5.4 L'algorithme de gradient conjugué

5.4.1 Ligne de recherche

Les algorithmes qui sont décrits dans ce chapitre impliquent une suite d'étapes dans l'espace de poids, il est commode de considérer chacune de ces étapes dans deux partis, premièrement nous devons décider la direction dans

laquelle se déplacer, et en second lieu, nous devons décider à quelle distance est déplacée cette direction. Par exemple, pour la descente de gradient simple, la direction de chaque étape est donnée par le gradient négatif locale $-\nabla E$ de fonction d'erreur, et la distance de déplacement à l'étape est déterminée par un paramètre de taux d'apprentissage arbitraire η .

*Plus généralement nous pouvons considérer une certaine direction de recherche dans l'espace de poids, et puis nous trouvons le minimum de fonction d'erreur le long de cette direction. Ce procédé est référé en tant qu'une ligne de recherche. Elle est la base de plusieurs algorithmes qui sont considérablement plus puissants que la descente du gradient.

*Nous supposons qu'à l'étape τ, la minimisation le long d'une direction de recherche particulière noté d^{τ}dans la prochaine valeur dans l'espace de poids est donnée par :

$$w^{(\tau+1)} = w^{(\tau)} + \lambda^{(\tau)} d^{(\tau)} \tag{5.27}$$

Où $\lambda^{(\tau)}$ est choisi pour réduire au minimum.

$$E(w) = E(w^{(\tau)} + \lambda d^{(\tau)}) \tag{5.28}$$

Si nous choisissons la direction de recherche, l'expression (5.26) nous donnera une procédure automatique pour placer la longueur d'étape.

5.4.2 La procédure d'algorithme de gradient conjugué

*Pour appliquer la ligne de recherche du problème de minimisation de la fonction d'erreur, nous devons choisir une direction approprié de recherche à chaque étape de l'algorithme.

*Nous notons que, au minimum de la ligne de recherche dans (5.26) nous avons :

$$\frac{\partial}{\partial \lambda} E(w^{(\tau)} + \lambda d^{(\tau)}) = 0 \tag{5.29}$$

D'après (5.26) alors :

$$\begin{aligned} d^2 \quad \frac{\partial}{\partial \lambda} E(w^{(\tau)} + \lambda d^{(\tau)}) &= 0 \\ w^{(\tau)} + \lambda d^{(\tau)} &= w^{(\tau+1)} \\ d^{\tau} \quad \frac{\partial}{\partial \lambda} E(w^{(\tau+1)}) &= 0 \\ g^{(\tau+1)T} d^{(\tau)} &= 0 \end{aligned} \tag{5.30}$$

Où $g \equiv \nabla E$, alors le gradient au nouveau minimum est orthogonal à la direction de recherche précédente.

l'equation (5.28) est équivalente à E au point $w^{(\tau+1)}$, nous avons :

$$g(w^{(\tau+1)})^T \; d^{(\tau)} = 0 \tag{5.31}$$

*Nous choisissons maintenant la prochaine direction de recherche $d^{(\tau+1)}$
D'après (5.27),

$$g(w^{(\tau+1)})^T \; d^{(\tau+1)} = 0 \tag{5.32}$$

Alors

$$\begin{aligned} g(w^{(\tau+1)} + \lambda d^{(\tau+1)})^T \; d^{(\tau+1)} &= 0 \\ g(w^{(\tau+1)} + \lambda d^{(\tau+1)})^T \; d^{(\tau)} &= 0 \\ g(w^{(\tau+1)})^T \; d^{(\tau)} + \lambda g(d^{(\tau+1)})^T d^{(\tau)} &= 0 \end{aligned} \tag{5.33}$$

Nous avons :

$$d^{(\tau+1)} H d^{(\tau)} = 0 \tag{5.34}$$

Où H est la matrice de *HESSIEN* évalué au point $w^{(\tau+1)}$

Si la surface d'erreur est quadratique, la matrice de *HESSIEN* contient des termes constants et évalués dans l'expression de (5.28) et ou le terme puissance multiplicatif de λ a disparu. Nous recherchons les directions conjugués qui satisfont (5.34) .

En effet, nous verrons qu'il est possible de construire une suite de direction de recherche $d^{(\tau)}$ successives tel que chaque direction est conjuguée à toutes les directions précédentes ; c'est l'algorithme d'optimisation de gradient conjugué.

Fonction d'erreur quadratique

Afin de présenter l'algorithme de gradient conjugué, nous considérons le cas d'une fonction d'erreur quadratique de la forme :

$$E(w) = E_0 + b^T w + \frac{1}{2} w^T H w \tag{5.35}$$

Dans ce cas les paramètres b et H sont constants.

Et nous assumons que H est défini. Le gradient local de ceci est donné par :

$$g(w) = b + Hw \tag{5.36}$$

Et la fonction d'erreur (5.35) est réduite au minimum au point w^* et en utilisant (5.36) pour donner :

$$b + Hw^* = 0 \tag{5.37}$$

Supposons que nous pouvons trouver un ensemble de w vecteurs (w est la dimensionnalité d'espace de poids) qui sont mutuellement conjugués par rapport H, alors

$$d_j^T H d_i = 0 \qquad \text{pour } j \neq i \tag{5.38}$$

Alors, nous montrons facilement que ces vecteurs seront linéairement indépendant si H est défini positif. De tels vecteurs forment une base complete mais non orthogonale dans l'espace de poids. Alors nous pouvons écrire la différence entre w_1 et w^* (ou w_1 sont les points de départ) comme combinaison linéaire des vecteurs de direction conjugués [16] sous la forme :

$$w^* - w_i = \sum_{i=1}^{w} \alpha_i d_i \tag{5.39}$$

Noter que, si nous posons :

$$w_j = w_1 + \sum_{i=1}^{j-1} \alpha_i d_j \tag{5.40}$$

Alors (5.39) peut être écrit comme équation itérative sous la forme :

$$w_{j+1} = w_j + \alpha_j d_j \tag{5.41}$$

Afin de trouver des expressions pour les α, nous multiplions (5.39) par $d_j^T H$ pour donner :

$$d_j^T H(w^* - w_1) = \sum_{i=1}^{w} \alpha_i d_j^T H d_i$$

$$d_j^T H w^* - \quad d_j^T H w_1 = \sum_{i=1}^{w} \alpha_i d_j H d_i \tag{5.42}$$

D'après (5.37) $Hw^* = -b$, en substituant ce résultat dans (5.42), nous obtenons :

$$-d_j^T (b + H w_1) = \sum_{i=1}^{w} \alpha_i d_j^T H d_i \tag{5.43}$$

Et nous utilisons l'expression (3.36) pour obtenir :

$$-d_j^T (b + H w_1) = \alpha_j d_j^T H d_j$$

Alors

$$\alpha_j = -\frac{d_j^T (b + H w_1)}{d_j^T H d_j} \tag{5.44}$$

Sans cette propriété, (5.44) qui représente un ensemble d'équations couplées pour α_i

nous pouvons écrire (5.41) sous une forme plus commode(5.38) comme suit. :

$$d_j^T H w_j = d_j^T H w_1 \tag{5.45}$$

Où nous avons encore employé l'état de conjugué (5.36). Ceci permet au numérateur du côté droit de (5.41) de s'écrire sous la forme :

$$d_j^T (b + H w_1) = d_j^T (b + H w_j) = d_j^T g_j \tag{5.46}$$

Où $g_j = g(w_j)$ et nous utilisons (5.34). Alors, α_j, peut être écrit dans la forme

$$\alpha_j = -\frac{d_j^T g_j}{d_j^T H d_j} \tag{5.47}$$

Nous donnons maintenant un argument inductif simple pour montrer cela, si les poids sont employés incrémentés (5.39) avec le $\alpha_i$donné par (5.44)

alors le vecteur de gradient de g_j à l'étape j sera orthogonal à toutes les directions conjuguées précédentes, et alors nous serons arrivés au minimum de la forme quadratique .Voir le schéma (5.5)

Pour dériver la propriété d'orthogonalité, nous notons (5.34) de celui

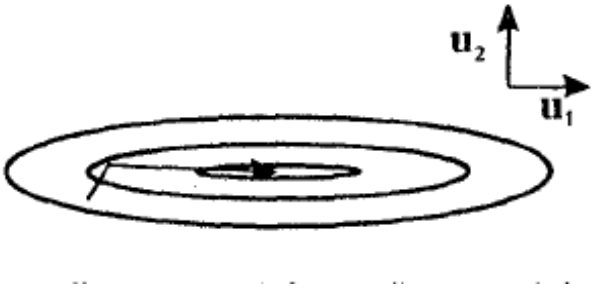

FIG. 5.5 – L'application de l'algorithme de gradient conjugué à la minimisation d'une fonction erreur quadratique bidimensionnelle [6]

$$g_{j+1} - g_j = H\,(w_{j+1} - w_j) = \alpha_j H d_j \tag{5.48}$$

Où nous avons employé (5.39).

Nous prenons maintenant le produit scalaire de cette équation avec (5.48), et employons la définition de α_j donnée par (5.44), pour donner

$$d_j^T g_{j+1} = 0 \tag{5.49}$$

De même, de (5.45), nous avons :

$$d_k^T(g_{j+1} - g_j) = \alpha_j d_k^T H d_j = 0 \quad \text{pour tout } k \prec j \leq W \tag{5.50}$$

En appliquant la technique de l'induction (5.46) et de (5.47), nous obtenons le résultat suivant

$$d_k^T g_j = 0 \qquad \text{pour tout } k \prec j \leq W \tag{5.51}$$

Le prochain problème est comment construire un ensemble avec des directions mutuellement conjuguées. Ceci peut être réalisé en choisissant la première direction pour être le gradient négatif $d_1 = -g_1$, et puis le choix de chaque direction successive pour être une combinaison linéaire du gradient courant et de la direction précédente de recherche

$$d_{j+1} = -g_{j+1} + \beta_j d_j \tag{5.52}$$

Les coefficients β_j peuvent être trouvés en imposant l'état de conjugué ce qui donne

$$\beta_j = \frac{g_{J+1}^T H d_j}{d_j^T H d_j} \tag{5.53}$$

D'après (5.49) le d_k est donné par une combinaison linéaire de touts les vecteurs de gradient précédents

$$d_k = -g_k + \sum_{l=1}^{k-1} \gamma_l g_l \tag{5.54}$$

Nous employons (5.48), nous avons alors

$$g_k^T g_j = \sum_{l=1}^{k-1} \gamma_l g_l^T g_j \text{ pour tout } k \prec j \leq W \tag{5.55}$$

Puisque la direction initiale de recherche est $d_1 = -g_1$,nous pouvons employer (5.49)pour montrer que $g_1^T g_j = 0$, de sorte que le gradient à l'étape j soit orthogonal au gradient initial.

Si nous appliquons l'induction de (5.55), nous constaterons que le gradient courant est orthogonal à touts les gradients précédents

$$g_k^T g_j = 0 \text{ pour tout } \quad k \prec j \leq W \tag{5.56}$$

Nous avons maintenant développé un algorithme pour trouver le minimum d'une fonction erreur quadratique général dans w étapes au maximum.

La procédure de l'algorithme de gradient conjugué

Nous avons maintenant développé un algorithme pour trouver le minimum d'une fonction d'erreur quadratique générale, puisque l'évaluation de H est couteuse au point de vue calcul informatique. Pour les réseaux non linéaire, nous voudrions éviter d'employer la matrice de *HESSIEN.*

En effet, il s'avère que les coefficients α_j et β_j peuvent être trouvés sans connaissance explicitée de H.

*Considérons d'abord le coefficient β_j :

D'après (5.53), nous avons :

$$\frac{g_{j+1} - g_j}{\alpha_j} = H d_j \tag{5.57}$$

Nous substituons (5.51) dans (5.57) nous obtenons :

$$\beta_j = \frac{g_{j+1}^T (g_{j+1} - g_j)}{d_j^T (g_{j+1} - g_j)} \tag{5.58}$$

Ce qui est connu comme expression de *HESTENES-SLIEFET.*

Nous multiplions (5.52) par g_{j+1} nous obtenons :

$$d_j^T g_{j+1} = -g_{j+1}^T g_{j+1} + \beta_j d_j^T g_{j+1} \tag{5.59}$$

D'après (5.50), nous avons que $d_j^T g_{j+1} = 0$. Alors, nous utilisons (5.59) pour écrire :

$$d_j^T g_j = -g_j^T g_j \tag{5.60}$$

Alors, (5.58) est écrite dans la forme de *POLAK-RIBIERE*

$$\beta_j = \frac{g_{j+1}^T (g_{j+1} - g_j)}{g_j^T g_j} \tag{5.61}$$

Nous avons d'après (5.56) (la propriété d'orthogonalité des gradients) que $g_{j+1}^T g_j = 0$

Alors, simplifier (5.61) donne le résultat de la forme de *FELTCHU-REEVES* :

$$\beta_j = \frac{g_{j+1}^T g_{j+1}}{g_j^T g_j} \tag{5.62}$$

Nous notons que ces trois expressions pour β_j sont équivalentes[17]. Ils expriment tous que la fonction d'erreur est exactement quadratique.

La forme de *POLAK-RIBIERE* s'avère généralement donner des résultats légèrement meilleurs que les autres expressions.(petite valeur de β_j pour que les vecteurs successifs de gradient soient très semblables). Nous souhaitons également éviter l'utilisation de la matrice de *HESSIEN* pour évaluer α_j. Pour voir ceci, considérer une erreur quadratique donnée par (5.47) comme fonction du paramètre α donné par :

$$\alpha_j = \frac{d_j^T g_j}{d_j^T H d_j} \tag{5.63}$$

Nous voyons que le résultat dans (5.47) est équivalent à celui trouvé dans (5.63). Alors, nous pouvons remplacer l'évaluation explicite de α_j par une procédure numérique impliquant une ligne (direction) de recherche de minimisation d_j.

Nous récapitulons maintenant les étapes principales de l'algorithme :

1.Choisir un vecteur de poids initial w_1.

2.Evaluer le vecteur de gradientg_1, et prendre la direction de recherche initiale $d_1 = -g_1$

3.A l'étape j, réduire au minimum $E(w_j + \alpha dj)$ par rapport à α pour donner $w_{j+1} = w_j + \alpha_{min} d_j$

4.Tester pour voir si le critère d'arrêt est satisfait.

5.Evaluer le nouveau vecteur de gradient g_{j+1}.

6.Evaluer la direction de recherche nouvelle d_{j+1},nous employons (5.52) dans laquelle β_j est donné par (5.62) ,(5.61) ou la formule (5.58).

7.Prendre $j = j + 1$ et aller à 3.

Pour une fonction d'erreur non quadratique générale, l'erreur dans le voisinage d'un point donné est approximativement quadratique, et alors nous pouvons espérer de l'application du procédé ci-dessus une convergence efficace avec un minimum de l'erreur.

5.4.3 Les avantages de l'algorithme de gradient conjugué

1)Dans la pratique, l'utilisation des techniques de la ligne de recherche pour la minimisation assure que l'erreur ne peut pas augmenter à aucune étape et de tels algorithmes s'avèrent généralement avoir la bonne exécution dans de vraies applications.

2)Comme nous avons vu, l'algorithme de gradient conjugué fournit une technique de minimisation qui exige seulement l'évaluation de la fonction d'erreur et de son gradient.

3)Pour une fonction d'erreur quadratique, la technique de minimisation est garantie pour trouver le meilleur dans la plupart des étapes w.

4)L'algorithme de gradient conjugué présente clairement une amélioration significative à l'approche simple de descente de gradient (ne prend pas plusieurs mesure pour atteindre le minimum) schémas (5.4) et (5.5).

5)Dans l'algorithme de gradient conjugué et pour une fonction d'erreur non linéaire générale, La matrice locale de *HESSIEN* n'a pas besoin d'être définie positive.

6)L'utilisation d'une ligne de recherche permet la taille d'étape α dans l'algorithme choisie sans évaluer la matrice de *HESSIEN*.

5.4.4 Les inconvénients de l'algorithme

Dans la pratique, la fonction d'erreur ne sera pas quadratique. Alors, les différentes expressions pour β_j donnent différents résultats.

Dans la pratique, la fonction d'erreur peut être loin d'une équation quadratique, l'algorithme donc doit généralement être connu pour beaucoup d'itérations jusqu'à ce qu'un un critère d'arrêt soit atteint(une erreur suffisamment petite).

L'utilisation de la ligne de recherche présente quelques problèmes :

-Pour chaque ligne de recherche, la minimisation implique plusieurs évaluations de la fonction d'erreur, dont chacune est chère(au plan informatique).

-L'exécution globale de l'algorithme peut être sensible à la valeur de paramètre (ci-dessus), alors précisément, la ligne de recherche peut représenter beaucoup de gaspillage de calculs.

5.5 L'algorithme de Newton

Nous nous tournons maintenant vers une classe d'algorithmes qui font une utilisation explicite de la matrice de *HESSIEN.*

En utilisant l'approximation quadratique locale, nous pouvons obtenir une expression directe pour la direction du minimum de la fonction d'erreur.

D'après (**??**), le gradient avec un poids quelconque w est donné par :

$$g = \nabla E = H(w - w^*) \tag{5.64}$$

Où w^* est le minimum de la fonction d'erreur.

Alors d'après (5.64) nous obtenons,

$$\begin{aligned} H^{-1}g &= w - w^* \\ w^* &= w - H^{-1}g \end{aligned} \tag{5.65}$$

Le vecteur $-H^{-1}g$ s'appelle la direction de *Newton* ou étape de *Newton.*

La direction de *Newton* pour une surface d'erreur quadratique est évaluée à tout point w directement au minimum de la fonction erreur.

Puisque l'approximation quadratique employé pour obtenir (5.65) (non exacte, alors elle est nécessaire pour s'appliquer (5.65) itérativement, avec la matrice de HESSIEN à chaque nouveau point de recherche)

5.5.1 Les inconvénients de cette méthode

1). L'évaluation exacte de la matrice de *HESSIEN* pour les réseaux non linéaire exige $O(Nw^2)$ étapes, où N est le nombre de donnée et w le nombre de poids dans le réseau.

2). La matrice de *HESSIEN* doit être inversée, ce qui exige $O(w^3)$étapes et alors une plus grande demande informatique.

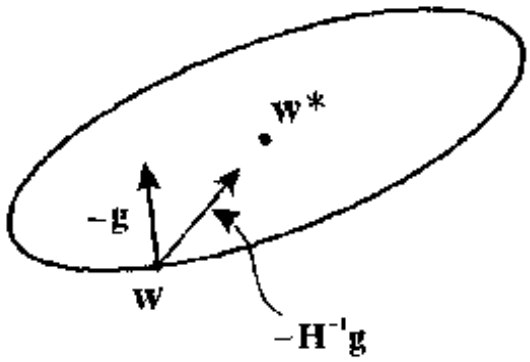

FIG. 5.6 – Le minimum de la fonction erreur, tandis que la direction de newton $-H^{-1}g\,(w)$

3). La taille d'étape prévue par $d = -H^{-1}g$ peut être suffisamment grande et etre en dehors de la gamme de validité de l'approximation quadratique. Dans ce cas, l'algorithme peut devenir instable.

Nous faisons diverses (différentes) modifications à la règle de *NEWTON* pour être transformé en méthode d'optimisation pratique.

5.6 Les algorithmes quasi-Newton

Il y a des approches alternatives (de l'application directe de la méthode *NEWTON)* connues sous le nom de l'algorithme *quasi-Newton.* ces méthodes sont basées sur (5.65) , mais au lieu de calculer la matrice de *HESSIEN* directement, ils évaluent son inverse. Alors, l'algorithme de *quasi-Newton* accumule une approximation de l'inverse de la matrice H ,il implique de produire une suite des matrices $G^{(\tau)}$ qui représente des approximations de plus en plus précises de l'inverse de *HESSIEN* H^{-1}. De la formule de Newton (5.65), nous voyons que les vecteurs de poids aux étapes τ et $\tau+1$ sont liés aux gradients correspondants par :

$$w^{(\tau+1)} - w^{(\tau)} = H^{-1}(g^{(\tau+1)} - g^{(\tau)}) \tag{5.66}$$

L'approximation de $G^{(\tau)} = H^{-1}$est construite afin de satisfaire cette condition également. La formule la plus utilisée de mise à jour est la procédure de *Baoyden - Fletchen - Goldfard - Seramro (BFGS)* donnée par :

$$G^{(\tau+1)} = G^{(\tau)} + \frac{PP^T}{P^TV} - \frac{(G^{(\tau)}V)V^TG^{(\tau)}}{V^TG^{(\tau)}V} + (V^TG^TV)\mu\mu^T \tag{5.67}$$

Où nous avons défini les vecteurs suivants :

$$P = w^{(\tau+1)} - w^{(\tau)} \tag{5.68}$$

$$V = g^{(\tau+1)} - g^{(\tau)} \tag{5.69}$$

$$\mu = \frac{P}{P^T V} - \frac{G^{(\tau)} V}{P^T G^{(\tau)} V} \tag{5.70}$$

Il est facile de vérifier par la substitution directe que (5.67)satisfait la condition de quasi-Newton donnée en (5.66)

Donc, nous pouvons faire une discussion comme suit

■L'initialisation du procédé employant la matrice d'identité correspond à la première étape dans la direction du gradient négatif .

■A chaque étape de l'algorithme , la direction $-Gg$ est garantie pour être une direction de descente , la matrice G etant définie positive .

■La solution est d'employer l'algorithme de ligne de recherche, comme utilisé avec des gradients conjugués, pour trouver le minimum de la fonction d'erreur le long de la direction de recherche. Alors, le vecteur de poids varie comme dans ce qui suit :

$$w^{(\tau+1)} = w^{(\tau)} + \alpha^{(\tau)} G^{(\tau)} g^{(\tau)} \tag{5.71}$$

où $\alpha^{(\tau)}$ est trouvé (5.71) par la ligne de minimisation.

Nous avons vu que les deux méthodes quasi-Newton et le gradient conjugué évitent l'utilisation de la matrice de Hessien, mieux , la méthode de quasi-Newton abandonne une autre étape qui est l'approximation de l'inverse de la matrice de Hessien. En plus la méthode de quasi-Newton atteint une convergence plus rapide avec la méthode de gradient conjugué. Mais le résultat net est que la complexité informatique de la méthode de quasi-Newton est $O(w^2)$, par contre la complexité informatique de la méthode de gradient conjugué est $O(w)$. La méthode de gradient conjugué est préférable à la méthode de quasi-Newton en terme d'informatique . À cause de ce dernier point," l'utilisation de quasi-Newton est restreint, dans la pratique, au réseau de neurone de petite taille"[14].

Conclusion Générale

Notre étude nous a essentiellement permis de constater les horizons immenses qui s'ouvrent à la statistique par la voie des réseaux de neurones.

D'une part, l'approche nouvelle des anciennes méthodes, développées dans le cadre traditionnel. D'autre part, l'apparition de nouvelles techniques permettant une extension des techniques connues afin de répondre à des problèmes reconnus difficiles.

Tout découle d'un même concept, et les diverses méthodes relevant de la statistique traditionnelle découlent par la simple action sur les trois paramètres de base :

- Le processus d'apprentissage (mode, règle et algorithme d'apprentissage)
- La nature des neurones constituant le réseau (représentée par la fonction d'activation)
- L'architecture du réseau.

Ne serait-ce que sous le mode d'apprentissage supervisé, auquel nous sommes limités dans cette étude, et avec l'architecture la plus triviale (qui est faite d'un seul neurone), nous sommes arrivés à voir comment se reproduisent un certain nombre de techniques classiques.

Ceci est obtenu uniquement par le fait d'utiliser une fonction d'activation ou une autre.

- Fonction seuil : avec un réseau constitué d'un seul neurone et qui est muni de la fonction d'activation la plus basique, la fonction seuil, nous pouvons reproduire la discrimination linéaire simple.

L'espace des observations est séparé en deux sous-espaces par une droite discriminante qui est l'équation du neurone.

Si nous devons diviser l'espace en plusieurs régions, il suffit de mettre en parallèle des neurones identiques en nombre égal au nombre de frontières que nous voulons obtenir.

- Fonction linéaire : avec un seul neurone muni d'une fonction linéaire, nous pouvons reproduire la régression linéaire simple.

La régression linéaire multiple est obtenue en mettant simplement en parallèle autant de neurones que de variables à expliquer.

- Fonction non linéaire : avec un seul neurone muni d'une fonction non linéaire, nous pouvons reproduire la régression logistique.

Nous pouvons tracer une ligne de frontière discriminante dans l'espace des données.

Ainsi, avec l'architecture la plus élémentaire, et sous le seul mode d'apprentissage supervisé, nous obtenons déjà plusieurs techniques.

L'augmentation du nombre de neurones et du nombre de couches permet

une meilleure flexibilité. Elle permet, comme c'est le cas des PCM, d'ajuster toute sorte de fonctions, quelle que soit sa complexité. Les problèmes de discrimination non linéaire trouvent aussi leurs solutions par le moyen de ce type de réseau.

Annexe A

n°de malade	chol/100g/l	TA M	groupe
1	102	100	0
2	112	105	0
3	113	92	0
4	117	106	0
5	123	94	0
6	125	95	0
7	131	93	0
8	146	105	0
9	149	103	0
10	151	107	0
11	151	115	0
12	152	95	0
13	155	102	0
14	157	118	0
15	157	116	0
16	159	113	0
17	162	98	0
18	167	104	0
19	169	109	0
20	169	90	0
21	173	100	0
22	175	110	0
23	179	119	0
24	182	112	0
25	186	106	0
26	186	109	1
27	189	105	0
28	192	108	0
29	194	110	0
30	200	110	1
31	205	119	0
32	208	120	0
33	208	116	0
34	217	110	0
35	218	161	0
36	225	143	0
37	227	116	0
38	227	138	1
39	229	110	0
40	235	124	1
41	238	119	1
42	240	149	1
43	242	131	1
44	249	105	0
45	252	125	1
46	255	134	0
47	257	154	1
48	265	141	1
49	269	129	1
50	273	150	1
51	290	142	1
52	295	132	1
53	298	148	1
54	301	160	1
55	305	138	1
56	307	150	1
57	352	142	1
58	361	154	1
59	401	162	1

Annexe B

ind1	gly j/100gr/l	gl PP /100gr/l	groupe
1	72	106	0
2	80	112	0
3	82	132	0
4	83	126	0
5	86	115	0
6	90	130	0
7	90	135	0
8	92	137	0
9	93	129	0
10	94	129	0
11	94	124	0
12	94	134	0
13	95	131	0
14	96	137	0
15	96	121	0
16	97	118	0
17	98	121	0
18	98	117	0
19	98	122	0
20	99	121	0
21	99	120	0
22	100	132	0
23	100	119	0
24	100	127	0
25	101	118	0
26	102	121	0
27	105	135	0
28	107	130	0
29	108	126	0
30	108	123	0
31	109	164	1
32	111	173	1
33	112	156	1
34	113	145	1
35	116	174	1
36	118	153	1
37	118	161	1
38	120	181	1
39	122	154	1
40	122	301	1
41	124	198	1
42	128	275	1
43	129	166	1
44	132	186	1
45	134	215	1
46	136	205	1
47	137	290	1
48	150	286	1
49	154	196	1
50	156	180	1
51	172	285	1
52	175	301	1
53	178	254	1
54	100	312	1
55	195	285	1
56	195	302	1
57	200	321	1
58	205	315	1
59	205	328	1

Annexe C

n° de mal	taille(mm)	v de densité(UH)	groupe
1	3	5	1
2	3	7	1
3	4.5	4	1
4	5	10	1
5	5	3	1
6	6	11	1
7	6	9	1
8	6.5	4	1
9	6.5	10	1
10	7.8	6	1
11	8	8	1
12	8.2	10	1
13	8.5	5	1
14	8.8	12	1
15	8.8	11	1
16	9	3	1
17	9.3	7	1
18	9.3	2	1
19	9.6	14	1
20	10	18	1
21	10.2	26	1
22	10.4	42	1
23	10.7	25	2
24	10.8	19	2
25	11	6	2
26	11.3	20	2
27	11.5	3	2
28	11.7	22	2
29	11.9	11	2
30	13	31	2
31	13.2	17	2
32	14	9	2
33	14.5	12	2
34	14.8	23	2
35	14.8	30	2
36	15.2	14	2
37	15.5	13	2
38	16.5	9	2
39	16.9	19	2
40	17.2	35	2
41	18.3	46	2
42	22.2	29	2
43	22.6	17	3
44	22.8	26	3
45	23	36	3
46	24.5	49	3
47	25	60	3
48	25.4	61	3
49	25.8	72	3
50	25.9	49	3
51	26.5	53	3
52	26.5	61	3
53	27	49	3
54	28.7	32	3
55	29	37	3
56	30	69	3
57	32	52	3
58	35	67	3
59	37	72	3

Bibliographie

[1] Anderson, M. J. and P. Legendre. (1999). An empiriques comparison of permutation methods for tests of partial regression coefficients in a linear model. Journal of Statistical Computation and Simulation. PP 271-303.

[2] Aima. (2005). Neurones Artificiels : modèle et réseau. C. Pellegrini.

[3] Alani, T. (2008). Réseaux de neurones formels. Département Informatique A2SI –ESIEE-Paris. PP 05-28

[4] Bishop, C. M. (1991). A fast procedure for retraining the multilayer perceptron. International Journal of Neural System. 2(3). PP 299-236.

[5] Bishop, C. M. (1993). Curvature – driven smoothing : a learning algorithm for feedforword networks. IEEE Transactions on Neural Networks. PP 882-884.

[6] Bishop, M. (1995). Neural Networks for Pattern Recognition. Oxford University Press. PP253-290

[7] Bouveyron, Chales. (2006). Modélisation et classification des donnés de grande dimension : application à l'analyse d'images. Université Joseph Fourier –Grenoble1. PP 24-28.

[8] Bouzy, Bruno. (18 October 2005). Réseau de neurones.

[9] Cannu, Stéphane. (19 Novembre 2001). Théorie Bayesienne de la décision.

[10] Château, Thieray. Reconnaissance des formes : décision Bayesienne. 1. LASMEA, UMR 6602 CNRS.

[11] Denis, Dollfus. (24 novembre 1997). Reconnaissance de Formes Naturelles par des Reseaux de Neurones Artificiels : Application au Nannoplanction Calcaire. Aix-Marseille III. PP

[12] Forsythre, D et J. Ponce. (2003). Computer Vision : A Modern Approach. Prentice-Hall. PP

[13] Gosselin, Bernard. (1996). Application de Reseaux de Neurones Artificiels a la Reconnaissance Automatique de Caracteres Manuscrits. Faculté Polytechnique de Mons.

[14] Haykin, Simon. (1999). A Neural Networks : A Comprehensive Foundation. 2eme Ed. Prentice- Hall,Inc : USA. PP

[15] Hristev, A.M. (1998). Artificial Neural Networks. The GNU Public License, ver 2. PP129-137

[16] Jedrzejewski, Franck. (2005). Introduction aux méthodes numériques. 2eme Ed. Spring-Verlag : France, Paris. PP 119-121.

[17] Kroise, Ben et Patrick Van Der Smaget. (1996). An Introduction to Neural Networks. 8eme Ed. The University of Amsterdam. PP 14-44.

[18] Liming, Chen et Emmanuel Dellandéra. Reconnaissance de forme : théorie de la décision Boyésiénne. Ecole centrale de lyon.

[19] Marine, Campedel. (avril 2005). Classification supervisée. TELECOM PARIS Ecole Nationale Supérieure des Télécommunications. PP

[20] McCulloch, W. et W. Pitts. (1988). A logical calculus of the ideasimmanent in nervous activity. Bulletin of Math, Biophysics. Vol 5. PP115-133.

[21] Parizeau Marc. (Automne 2004). Réseaux de Neurones. University Laval. PP 27-51

[22] Rakotomalala, Rick. Réseaux de neurons artificiel : perceptron simple et multicouche. Application du reseaux de neurone à l'apprentissage supervisé. Laboratoire Eric.

[23] Samarasinghe, Sandhya. (2007). Neural Networks for applied Sciences and Enginering From Fundamentals to Complex Patten Recognition. Taylor and Francis Group, LLC. PP

[24] Samprit, Chatterjee et Bert Bamprice. (1991). Regression Analysis by Example. 2eme Ed. John Wiley and sons, Inc. PP 193-197.

[25] Tomassone, R et al. (1992). La Régression nouveaux regards sur une ancienne méthode statistique. 2eme Ed. MASSON. PP 107-131.

[26] Touzet, Claude. Les réseaux de Neurones Artificiels : Introduction au Conexionisme Neurosysteme. Parc Scientifique Georges Besse, 30000 Nime. PP22-26

Résumé

Le but de l’étude permis de constater les horizons immenses qui s'ouvrent à la statistique par la voie des réseaux de neurones.

D'une part, l'approche nouvelle des anciennes méthodes, développées dans le cadre traditionnel. Sous le mode d'apprentissage supervisé, nous sommes arrivés à voir comment se reproduisent un certain nombre de techniques classiques. Spécifiquement nous avons été montre qu’un perceptron simple et un classificateur linéaire, sont équivalentes á l’analyse discriminante linéaire dans la statistique.

D'autre part, l'apparition de nouvelles techniques permettant une extension des techniques connues afin de répondre à des problèmes reconnus difficiles. Par exemple le cas des PCM, qui traitent des problèmes non linéaires quelle que soit son complexité. Les problèmes de discrimination non linéaire trouvent aussi leurs solutions par le moyen de ce type de réseau.

Summary

The purpose of the study allowed noting the immense horizons which open with the statistics by the way networks of neurons.

On the one hand, new approach of the old methods, developed within the traditional framework. Under the mode of supervised training, we managed to see how reproduce a certain number of traditional techniques. Specifically we were shows that a simple perceptron and a linear classifier, are equivalent á the linear discriminating analysis in the statistics.

In addition, appearance of new techniques allowing an extension of the known techniques in order to answer problems found difficult. For example the case of the MLP which deal with nonlinear problems whatever its complexity. The problems of nonlinear discrimination find also their solutions by the means of this type of network.

الملخـــــــص

الغرض من هذه الدراسة فتح آفاق واسعة للإحصاء من خلال الشبكات العصبية الصناعية؛ إذ تعتبر هذه الأخيرة من جهة نهجا جديدا للأساليب القديمة حيث و تحت إشراف وضع التعلم، وصلنا إلى معرفة كيفية تطبيق عدد من التقنيات التقليدية على وجه التحديد أظهرنا أن البرسبترون البسيط والمصنف الخطي يعادلان التحليل التميزي الخطي في مجال الإحصاء من جهة أخرى، أوجدت تقنيات جديدة لتمديد التقنيات المعروفة لمعالجة المشاكل الصعبة، على سبيل المثال حالة البرسبترون متعدد الطبقات الذي تعامل مع المشاكل غير الخطية بغض النظر عن تعقيدها. فقضايا التمييز غير الخطي هي أيضا لها حلولا استعمال هذا النوع من الشبكات.