كلية العلوم و التقنيات فاس
ⵜⴰⵖⵙⵍⵓⵜ ⵏ ⵜⵎⴰⵙⵙⴰⵏⵜ ⴰ ⵜⵓⵙⵙⵔⵜⵙ
aculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵃⵎⴷ ⴱⵏ ⵄⴱⴷ ⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

DEPARTEMENT DES MATHEMATIQUES

Master Mathématiques et Applications au Calcul Scientifique

(MACS)

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du Diplôme de Master Sciences et Techniques (MST)

Etude et application du réseau ELM (Extreme Learning Machine)

Réalisé par : B0USSAID Essaid **Encadré par :** Pr. OUANAN Mohamed

Soutenu le 22 Juin 2022

Devant le jury composé de :

- Pr. Lahcen Oukhtite FST Fès
- Pr. Hilali Abdlmjid FST Fès
- Pr. OUANAN Mohamed FS Meknès

Année Universitaire 2021 / 2022

FACULTE DES SCIENCES ET TECHNIQUES FES – SAISS

B.P. 2202 – Route d'Imouzzer – FES

Université Sidi Mohamed Ben Abdellah
Faculté des Sciences et Techniques Fès

Département de Mathématiques
Master MACS

MÉMOIRE DE FIN DES ÉTUDES

Thème :

Etude et application du réseau ELM (Extreme Learning Machine)

Réalisé par :
- BOUSSAID *Essaid*

Encadré par :
- *Pr.* OUANAN *Mohammed*

soutenu le 23/06/2022
Devent le jury :
Pr.Lahcen okhtite
Pr. Mohammed Ouanan
Pr. Hilali Abdelmjid

Année Universitaire 2021 - 2022.

Table des matières

Remerciement

Au terme de ce travail, je vais présenter mes remerciements à mon encadrant *Pr. Ouanan Mohmmad* pour ses précieux conseils et ses encouragements au cours de la préparation de ce projet, je lui exprime ma profonde gratitude.

Je remercie également les membres du jury : *Pr. Hilali Abdelmjid*, *Pr. Lhcen Oukhtite* pour leurs conseils avisés et leurs commentaires très constructifs, qui seront sans doute très utile pour mon étude.

Mes remerciements vont aussi à l'ensemble des professeurs du département de Mathématiques de la Faculté des Sciences et Techniques de Fes qui ont assuré avec succès l'encadrement et l'enseignement de la filière Mathématiques Appliquées aux calculs scientifiques.

Mes remerciements vont enfin à maman et à toute personne qui a contribué de près ou de loin à l'élaboration de ce travail.

Acronymes

- IA : intelligence artificielle
- ML :machine learning
- ELM : la machine d'apprentissage extrême
- RBF :fonction de base radiale
- SLFN :réseaux avancés mono-couche
- RNA :réseau de neurone artificiel
- SVM :Séparateurs à Vaste Marge
- RMS :Root mean square(moyenne quadratique)

INTRODUCTION GÉNÉRALE

Le réseau de neurones artificiels inspiré du système nerveux biologique, est un automate caractérisé par un nombre de fonctions mathématiques. Il traite un signal recueilli (signal d'entrée) à travers ses connexions entrantes pour fournir un signal en sortie, calculé par la fonction de transfert du réseau. En général, ils considèrent que chaque neurone (unité de calcul) fournit une information additive aux neurones qui lui sont connectés.

Il existe plusieurs types de réseaux de neurones. En particulier, nous nous sommes intéressés aux réseaux de neurones multicouches de type feed-forward, qui sont les plus utilisés pour la classification de données. Ces réseaux de neurones multicouches fonctionnent comme des approximateurs de fonctions. Les algorithmes d'apprentissage conventionnels, pour la modélisation des réseaux de neurones, doivent toujours ajuster les paramètres du réseau, en particulier ceux de la couche cachée. Ce qui nécessite un temps d'apprentissage relativement long par rapport à la taille et la dimension des données utilisées. En plus, le choix de l'architecture du réseau est d'une grande importance pour la résolution du problème d'approximation.
Contrairement à ces considérations, un nouveau réseau a vu le jour, récemment, qui ne se soucie pas beaucoup de taille du réseau. En plus, d'un point de vue d'approximation de fonctions, les paramètres de la couche cachée sont générés de manière aléatoire et restent constants au lieu d'être ajustés. Seuls les poids de sortie sont calculés par une simple régression linéaire. Ce réseau, connu sous l'appellation réseau de neurones ELM (Extrême Learning Machine), a été proposée par **Guang-Bin Huang** en 2006. Parmi les avantages de ce nouveau réseau nous citons : la vitesse d'apprentissage qui est estimée à 100 fois plus rapide et une bonne capacité de généralisation.

L'objectif de notre travail est d'étudier le réseau ELM, ainsi confirmer ces performances en classification de données. Pour cela, nous avons réparti notre travail comme suit :
Dans le premier chapitre,nous avons commencés avec une introduction sur les réseaux de neurones artificiels,aprés nous avons consacrés à expliquer le réseau RBF qui est la base de notre travail, son architecteure, son principe et son fonctionnement.
Le deuxième chapitre porte sur la description du réseau ELM : architecture, fonctionnement,et

avantages, et nous avons étudiés le cas d'un réseau RBF.

Dans le dernier chapitre, nous avons traités trois catégories des problèmes, le premier est la comparaison des problèmes d'approximation des fonctions artificielles ,et le deuxième consiste à l'analyse comparative avec des problèmes d'approximation des fonctions dans le monde réel ,et le dernier s'interesse à l'analyse comparative avec les applications de classification du monde réel. chaque catégorie se compose de deux exemples ..

Enfin nous terminons notre travail par une conclusion générale.

Chapitre 1

Réseaux des neurones artificiels

1.1 Introduction

Dans ce chapitre, nous allons définir c'est quoi un réseau de neurone artificiel par analogie avec les réseaux de neurones biologiques, nous introduisons, aussi, la modélisation mathématique des réseaux de neurones artificiels. Par la suite, nous verrons, le concept d'apprentissage qui est une faculté très importante pour les réseaux de neurones, d'où leur utilisation dans des domaines très variés.

1.2 historique

De nombreux ouvrages ont permis de documenter l'histoire des recherches en réseaux de neurones. Les recherches menées dans le domaine du connexionnisme ont démarré avec la présentation en 1943 par W. McCulloch et W. Pitts[1] d'un modèle simplifié de neurone biologique communément appelé neurone formel (Artificiels). Ils ont montré théoriquement que les réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes.

En 1958, F. Rosenblatt[1] développe le modèle du Perceptron. Ce dernier possède deux couches de neurones : une couche de perception (sert à recueillir les entrées) et une couche de décision. C'est le premier modèle pour lequel un processus d'apprentissage a pu être défini.

S'inspirant du perceptron, Widrow et Hoff[1], développent, dans la même période, le modèle de l'Adaline (Adaptive Linear Element). Ce dernier sera, par la suite, le modèle de base des réseaux de neurones multicouches.

Les recherches sur les réseaux de neurones ont été pratiquement abandonnées lorsque M. Minsky et S. Papert [1]ont publié leur livre « Perceptrons » en (1969) et démontré les limites théoriques du perceptron, en particulier, l'impossibilité de traiter les problèmes non linéaires par ce modèle.

Une révolution survient alors dans le domaine des réseaux de neurones artificiels, une nouvelle génération de réseaux de neurones capable de traiter avec succès des phénomènes non-linéaires : le perceptron multicouche, il ne possède pas les défauts mis en évidence par Minsky. Proposé pour la première fois par Werbos[1]. Le perceptron Multicouche apparait en 1986 introduit par Rumelhart, et simultanément sous une appellation voisine, chez le Cun(1985). Ces systèmes reposent sur la rétro-propagation du gradient de l'erreur dans des systèmes a plusieurs couches, chacune de types Adaline de Bernard Widrow, proche du perceptron de Rumelhart. L'évolution de la théorie des réseaux de neurones formels est inspirée directement du développement des travaux biologique sur le cerveau humain.

1.3 Le neurone biologique

Le cerveau est l'organe de commande le plus complexe et le plus inconnu de la biologie de l'homme ou de l'animal. Les cellules nerveuses, appelées neurones, sont les éléments de base du système nerveux central.
Dans leur organisation générale et leur système biochimique, ils présentent, cependant, des caractéristiques particulières qui se distinguent par quatre fonctions spécialisées :

- Recevoir des signaux en provenance des neurones voisins ;
- Intégrer ces signaux ;
- Engendrer un influx nerveux ;
- Conduire et transmettre l'influx nerveux à un neurone capable de le recevoir ;

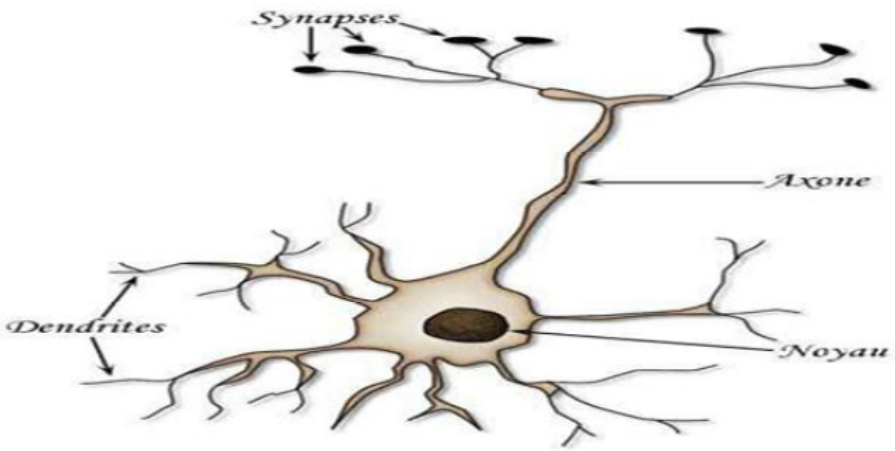


FIGURE 1.1 – neurone biologique..

- **Axone** : l'axone transporte l'influx nerveux vers les synapses.
- **Synapses** : transmettent l'influx nerveux provenant de l'axone vers d'autres cellules à partir neurotransmetteurs.

- **Dendrites** : selon leur longueur et leur perméabilité, affectent la quantité d'influx nerveux qui se rend au noyau.
- **Noyau** : est le centre des réactions électrochimiques, si les stimulations externes sont suffisantes, le noyau provoque l'envoi d'un influx nerveux électrique à travers l'axone.

1.4 Neurone formel (Artificiel)

Inspiré du système biologique, le neurone formel est un automate caractérisé par un petit nombre de fonctions mathématiques. Il traite un signal recueilli (signal d'entrée) à travers ses connexions entrantes pour fournir un signal de sortie calculé par la fonction de transfert. En générale, on considère que chaque neurone fournit une information additive aux unités de calcul (neurones) qui lui sont connectées[1].
La valeur de la somme pondérée est, simplement, la somme des sorties des différents neurones en amont avec lesquels il est connecté, La figure 2 résume la structure détaillée du neurone formel (Artificiel).

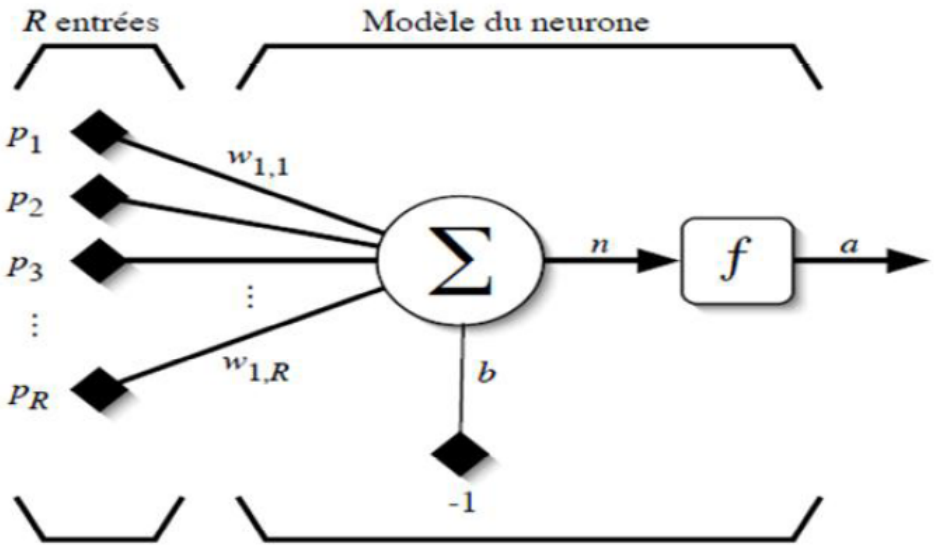


FIGURE 1.2 – Modèle générale du neurone formel.

P se compose de R entrées des neurones correspondent au vecteur de caractéristiques du signal d'entrée.

w : Vecteur des poids du neurone.

b : biais interne.

f : Fonction d'activation.

$a = f(n)$: Sortie du neurone.
avec : $n = \sum_{j=1}^{R} w_{1,j} p_j - b$.

1.5 Fonction d'activation

La fonction d'activation est une fonction mathématique appliquée à un signal en sortie d'un neurone artificiel. Le terme de "fonction d'activation" vient de l'équivalent biologique "potentiel d'activation", seuil de stimulation qui, une fois atteint entraîne une réponse du neurone. La fonction d'activation est souvent une fonction non-linéaire[1].

Les types de fonctions d'activation les plus utilisées sont :

- Fonction seuil :

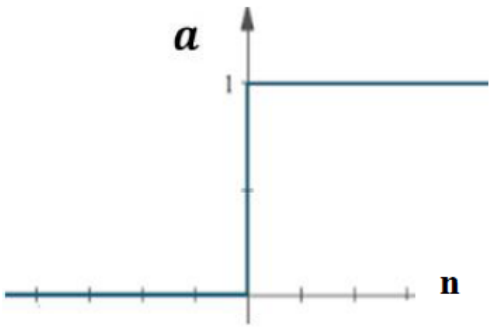


FIGURE 1.3 – Fonction seuil :.

- Fonction linéaire :

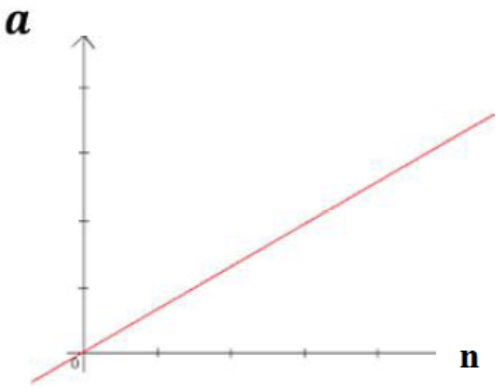


FIGURE 1.4 – Fonction linéaire

.

- Fonction tangente hyperbolique :

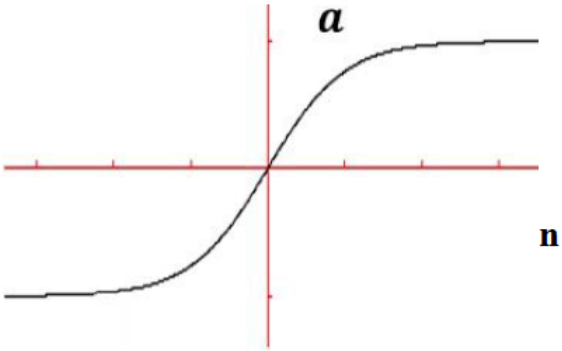


FIGURE 1.5 – Fonction tangente hyperbolique.

- Fonction sigmoïde

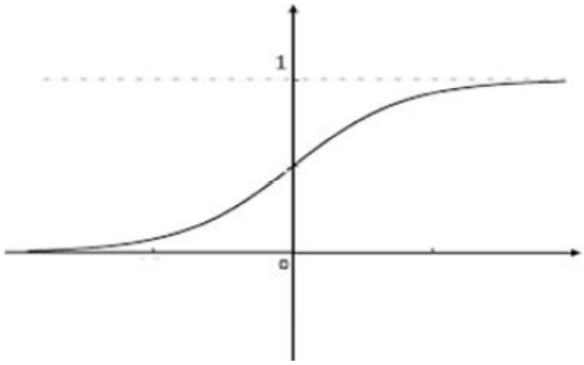


FIGURE 1.6 – Fonction sigmoïde

Les caractéristiques des fonctions d'activation sont les suivantes :
- la monotonie : la fonction d'activation est toujours croissant.
- le seuillage : possède une valeur au-dessous de laquelle sa valeur est négligeable.
- la saturation : elle possède une valeur maximale au-dessus de laquelle sa valeur de réponse est essentiellement fixe, ceci permet d'éviter de propager de grandes valeurs dans le réseau.

1.6 Structure de réseau monocouche et multicouche

1.6.1 Réseau monocouche

La structure d'un réseau monocouche est telle que des neurones organisés en entrée soient entièrement connectés à d'autres neurones organisés en sortie par des poids modifiables[2].

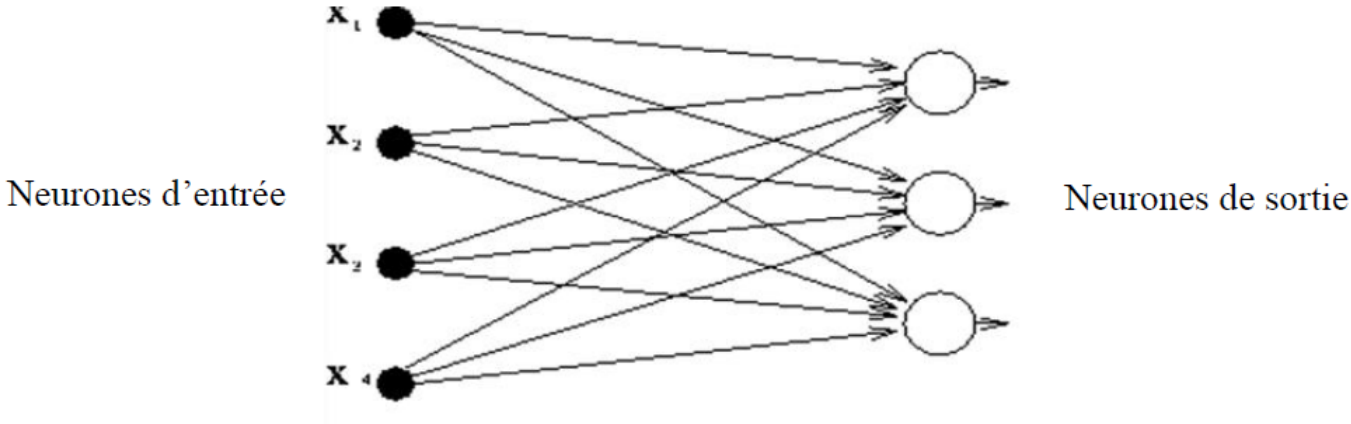


FIGURE 1.7 – Architecture de réseau à une seule couche

1.6.2 Réseau multicouches

Le réseau multicouches MLP (multi Layer Perceptron) est une extension de réseau monocouche, avec une ou plusieurs couches cachées entre l'entrée et la sortie (voir figure 1.8). Les entrées des neurones de la deuxième couche sont, en fait, les sorties des neurones de la première couche. Les neurones de la première couche sont reliés au monde extérieur et reçoivent tous le même vecteur d'entrée. Ils calculent alors leurs sorties qui sont transmises aux neurones de la deuxièmes couche. Les sorties des neurones de la dernière couche forment la sortie du réseau[2].

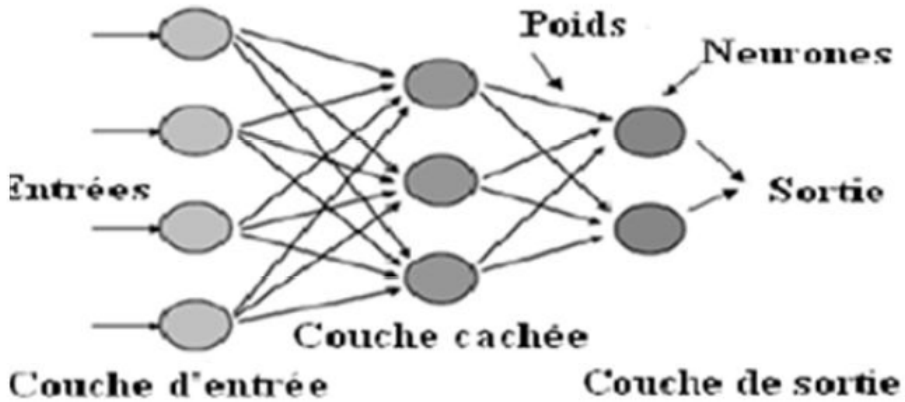


FIGURE 1.8 – architecture de réseaux multicouche

1.7 Types de réseaux de neurones artificiels (RNA)

Un réseau de neurones artificiels (RNA) est un ensemble de neurones formels (d'unités de calcul simples, de noeuds processeur) associés en couches (ou sous-groupes) et fonctionnent en

parallèle.

Dans un réseau, chaque sous-groupe fait un traitement indépendant des autres et transmet le résultat de son analyse au sous-groupe suivant. L'information donnée au réseau va donc se propager couche par couche, de la couche d'entrée à la couche de sortie, en passant soit par aucune, soit par plusieurs couches intermédiaires (dites couches cachées).

On peut distinguer essentiellement deux types de réseaux de neurones artificiels (RNA), les réseaux non boucles (feed-forward) et les réseaux boucles (back-forward) :

1.7.1 Réseaux de neurones non bouclés (statique ou non récurrent)

Un réseau de neurones non bouclé réalise une (ou plusieurs) fonction algébrique de ses entrées par composition des fonctions réalisées par chacun de ses neurones. Dans un tel réseau (voir figure 1.9), le flux d'informations circule des entrées vers les sorties sans retour en arrière (acyclique).

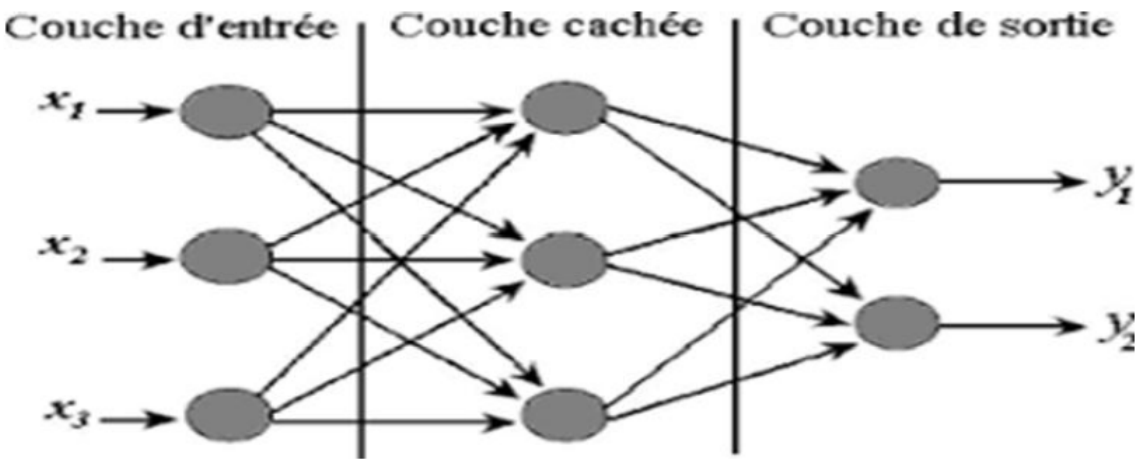


FIGURE 1.9 – Réseau de neurones non bouclé

Tout neurone dont la sortie est une sortie du réseau est appelé « neurone de sortie ».
Les autres, qui effectuent des calculs intermédiaires, sont des « neurones cachés ».

1.7.2 Réseaux de neurones bouclés (dynamique ou récurrents)

L'architecture la plus générale pour un réseau de neurones est le « réseau bouclé », dont le graphe des connexions est cyclique, lorsque on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ (un tel chemin est désigné sous le terme de « cycle »).

La sortie d'un neurone du réseau peut donc être fonction d'elle-même ; cela n'est évidemment concevable que si la notion de temps est explicitement prise en considération[7]

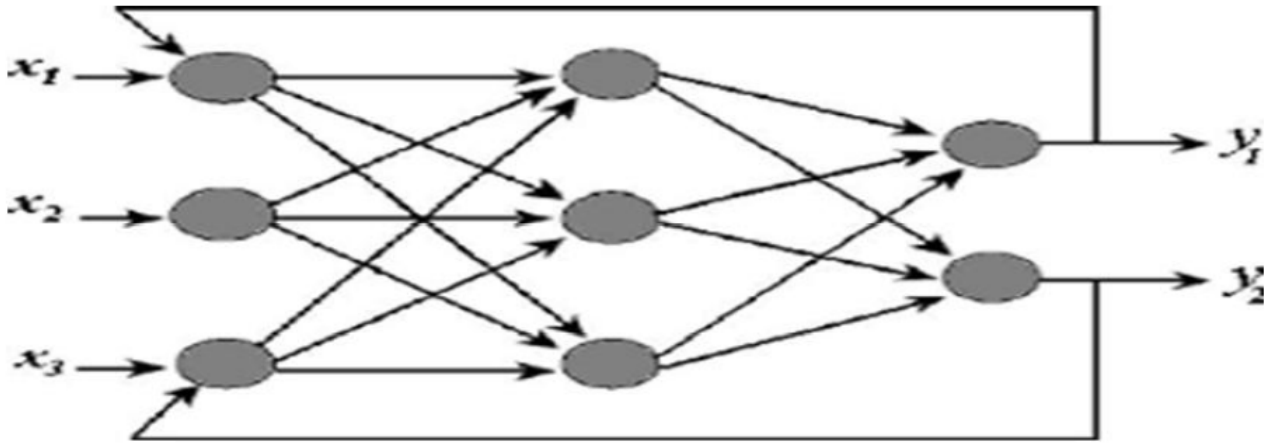


FIGURE 1.10 – Réseau de neurone bouclé

Réseau de neurone bouclé Les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Pour éliminer le problème de la détermination de l'état du réseau par bouclage, on introduit sur chaque connexion « en retour » un retard qui permet de conserver le mode de fonctionnement séquentiel du réseau. Le graphe des connexions de réseaux récurrents est cyclique. Ces réseaux sont décrits par un système d'équations aux différences.

La structure par couche peut être utilisée pour résoudre un certains nombre de problèmes, par exemple l'approximation de fonctions et la classification. Dans ce travail, nous sommes intéressés aux réseaux de neurones perceptron multicouches de type non bouclé. Une fois l'architecture est bien choisie, il reste à former le réseau par apprentissage.

1.8 Apprentissage des réseaux de neurone artificiels

L'apprentissage des réseaux de neurones artificiels est une phase qui permet de déterminer ou de modifier les paramètres du réseau, afin d'adopter un comportement désiré. Cela est réalisé en adaptant, grâce à certaines règles, les vecteurs de poids. Plusieurs algorithmes d'apprentissage ont été développés depuis la première règle d'apprentissage de Hebb en 1949[4] . Selon le degré de supervision, les algorithmes d'apprentissage peuvent être classés en trois catégories : supervisé, non supervisé et par renforcement.

1.8.1 Apprentissage supervisé

Dans ce type d'apprentissage, on cherche à imposer au réseau un fonctionnement donné en forçant à partir des entrées qui lui sont présentées, les sorties du réseau à prendre des valeurs données en modifiant les poids synaptiques. Le réseau se comporte alors comme un filtre dont les paramètres de transfert sont ajustés à partir des couples entrée/sortie présentés. L'adaptation des paramètres de réseau s'effectue à partir d'un algorithme d'optimisation, l'initialisation des poids synaptiques étant le plus souvent aléatoire

1.8.2 Apprentissage non supervisé

Dans ce cas, des exemples au « prototypes » ou « patrons » sont présentés au réseau qu'on laisse s'auto-organiser au moyen de lois locales qui régissent l'évolution des poids synaptiques. Ce mode d'apprentissage est aussi appelé « apprentissage par compétition »[4]. Ce type d'apprentissage est généralement utilisé pour la classification automatique des entrées, où le réseau apprend les caractéristiques des données d'entrée

1.8.3 Apprentissage par renforcement

Le mode par renforcement consiste á apprendre les actions á prendre, á partir d'expériences, de façon á optimser une récompence quantitative au cours de temps.

1.9 Le réseau à fonction de base radiale RBF (radial basis fonction)

Ce réseau est basé sur une architecture qui s'organise en deux couches seulement à savoir une couche cachée et une couche de sortie. La couche cachée donne une transformation non linéaire de l'espace d'entrée, la couche de sortie calcule une combinaison linéaire des sorties de la couche cachée. Chaque neurone élémentaire calcule la distance entre l'entrée et le centre qu'il passe ensuite dans une non linéarité concrétisée par une fonction d'activation généralement gaussienne, la valeur d'entrée la plus proche du centre prend une valeur importante.
Les fonctions de base utilisées ici sont des fonctions Gaussiennes. L'apprentissage le plus utilisé pour les RBF est le mode hybride et les règles sont soit, la règle de correction de l'erreur soit, la règle d'apprentissage par compétition.
Les RBF seront donc employés dans les mêmes types de problèmes que les PMC(le Perceptron Multicouches) à savoir, en classification et en approximation de fonctions,particulièrement.[05]

L'apprentissage du réseau RBF est hybride puisqu'il est non-supervisé pour la couche cachée et supervisé pour la couche de sortie. Le but de l'apprentissage non-supervisé est de trouver

les vecteurs poids des neurones de la couche cachée, qui définissent les centres des régions de sensibilité associées. Les paramètres principaux à régler dans un réseau RBF sont :

- Le nombre de neurones RBF (nombre de neurones dans l'unique couche cachée) : Ce nombre est égal au nombre de vecteurs d'entrée.
- La position des centres des gaussiennes de chacun des neurones.[02]

1.10 Règles d'apprentissage

L'apprentissage se fait selon des règles bien définies, parmi les règles d'apprentissage on peut citer :

1.10.1 Règle de Hebb

La loi d'apprentissage la plus simple est basée sur la règle de Hebb [4] qui suit en fait le comportement du neurone biologique : si deux neurones interconnectés sont simultanément activés, alors le poids de la connexion qui les relie doit être renforcé, d'où l'équation[12].

$$w_{i,j}(t+1) = w_{i,j}(t) + \Delta w_{i,j}(t)$$

Avec :
$w_{i,j}(t)$: Valeur à l'instant t du poids liant le neurone j au neurone i
$\Delta w_{i,j}(t) = x_i * x_j$ (La coactivité est modélisée comme le produit des deux valeurs d'activation).
L'algorithme d'apprentissage modifie de façon itérative (petit à petit) les poids pour adapter la réponse obtenue à la réponse désirée. Il s'agit en fait de modifier les poids lorsqu'il y a erreur seulement comme suit [12] :

1. Initialisation des poids et du seuil S à des valeurs (petites) choisies aléatoire.

2. Présentation d'une entrée $E_l = (e_1, ..., e_l)$ de la base d'apprentissage.

3. Calcul de la sortie obtenue pour cette entrée :

Avec :

$$a = \sum(w_i * e_i) - S$$

(la valeur de seuil est introduite ici dans le calcul de la somme Pondérée)

et x= signe(a)

4. Si la sortie est différente de la sortie désirée d_l pour cet exemple d'entrée E_l alors modification des poids :

$$w_{i,j}(t+1) = w_{i,j}(t) + \mu.x_i.xj$$

avec μ est une constante positive, qui spécifie le pas de modification des poids) :

1.10.2 Règle de rétro propagation

L'algorithme d'apprentissage par rétro-propagation du gradient de l'erreur est un algorithme itératif qui a pour objectif de trouver les poids des connexions minimisant l'erreur quadratique moyenne commise par le réseau sur l'ensemble d'apprentissage [5]. Cette minimisation par une méthode du gradient conduit à l'algorithme d'apprentissage de rétropropagation. Avant d'énoncer l'algorithme de rétro-propagation, nous citons les différentes notations utilisées dans celui-ci, qui sont :

Le problème de l'apprentissage pour les perceptrons multicouches consiste à reconnaitre la contribution de chaque poids sur l'erreur globale du réseau, et l'algorithme de rétro-propagation du gradient permet de faire cela [9] :

1. La propagation de l'information de l'entrée jusqu'à la sortie.
2. Le calcul de l'erreur en sortie.
3. La rétro-propagation de l'erreur de la sortie jusqu'aux entrées.

Nous décrirons dans la suite les différentes équations qui permettent de faire la mise à jour des pondérations de notre réseau.

★ Pour la couche de sortie :

Prenons un neurone j de la couche de sortie, lorsque le $n^{ème}$ exemple est lui présenté, il produit une sortie $y_j(n)$ alors que, c'est la valeur $d_j(n)$ qui est attendue, Notons :

$$e_j(n) = d_j(n) - y_j(n) \tag{1.1}$$

Et de ce fait, l'erreur globale du réseau pour l'exemple n est :

$$\varepsilon(n) = \frac{1}{2}\sum_{j=1}^{m} e_j^2(n) = \frac{1}{2}\|d(n) - y(n)\|_2^2 \tag{1.2}$$

où m est le nombre de neurones dans la couche de sortie. Le but de l'apprentissage est de minimiser l'erreur moyenne correspondante aux N exemples d'apprentissage, c'est à dire minimiser :

$$\varepsilon_{\text{moy}} = \frac{1}{N}\sum_{n=1}^{N} \varepsilon(n) \tag{1.3}$$

On note :

$$v_j(n) = \left(\sum_i \omega_{ij}(n) \times y_i(n)\right) - b_j(n) \tag{1.4}$$

L'entrée totale d'un neurone dont la fonction d'activation est f_j, sa sortie s'écrit alors sous la forme de

$$y_j(n) = f_j\left(v_j(n)\right)$$

Le mécanisme de rétro-propagation est basé sur l'affectation d'une correction $\Delta\omega_{ji}(n)$ et $\Delta b_j(n)$ aux poids et aux biais, on utilisera la règle dite du delta (delta rule), c'est à dire que la modification sera proportionnelle au gradient suivant :

$$\frac{\partial\varepsilon(n)}{\partial\omega_{ji}(n)} = \frac{\partial\varepsilon(n)}{\partial e_j(n)} \times \frac{\partial e_j(n)}{\partial y_j(n)} \times \frac{\partial y_j(n)}{\partial v_j(n)} \times \frac{\partial v_j(n)}{\partial\omega_{ji}(n)}$$

Qui détermine la direction dans laquelle on recherche les valeurs de $\omega_{ij}(n)$, nous appellerons taux d'apprentissage le facteur de proportionnalité et nous le noterons η par la suite.
d'après l'équation 1.2, on peut écrire :

$$\frac{\partial\varepsilon(n)}{\partial e_j(n)} = e_j(n)$$

De plus :

$$\frac{\partial y_j(n)}{\partial v_j(n)} = f_j'\left(v_j(n)\right)$$

Finalement et à partir de 1.4 :

$$\frac{\partial v_j(n)}{\partial\omega_{ji}(n)} = y_i(n)$$

Soit $\delta_j(n)$ le gradient local défini par :

$$\delta_j(n) = -\frac{\partial\varepsilon(n)}{\partial e_j(n)} \times \frac{\partial e_j(n)}{\partial y_j(n)} \times \frac{\partial y_j(n)}{\partial v_j(n)} \tag{1.5}$$

D'après ce qui précède ceci est égal à :

$$\delta_j(n) = e_j(n) \times f_j'\left(v_j(n)\right) \tag{1.6}$$

La correction appliquée au poids $\omega_{ji}(n)$ sera donc la suivante (règle du delta) :

$$\Delta\omega_{ji}(n) = -\eta \times \frac{\partial\varepsilon(n)}{\partial\omega_{ji}(n)} \tag{1.7}$$

Soit encore :

$$\Delta\omega_{ji}(n) = \eta \times \delta_j(n) y_i(n) \tag{1.8}$$

Du même, la correction apportée au biais s'écrira sous la forme :

$$\Delta b_j(n) = \eta \times \frac{\partial\varepsilon(n)}{\partial b_j(n)} \tag{1.9}$$

En suivant un raisonnement analogue à ce qui précède on obtient :

$$\begin{aligned}\frac{\partial\varepsilon(n)}{\partial b_j(n)} &= \frac{\partial\varepsilon(n)}{\partial e_j(n)} \times \frac{\partial e_j(n)}{\partial y_j(n)} \times \frac{\partial y_j(n)}{\partial v_j(n)} \times \frac{\partial v_j(n)}{\partial b_j(n)} \\ &= -\delta_j(n) \times \frac{\partial v_j(n)}{\partial b_j(n)} \\ &= \delta_j(n)\end{aligned}$$

D'où :

$$\Delta b_j(n) = -\eta \cdot \delta_j(n)$$

Ces deux équations de mise à jour ne sont valables que pour la couche de sortie, car on a utilisé l'erreur de sortie $e_j(n) = d_j(n) - y_j(n)$

★ Pour les couches cachées :

En ce qui concerne les autres couches, on ne peut plus utiliser cette formule pour l'erreur, alors nous serons forcement obligés de déterminer les nouvelles équations de mise à jour.

Considérons un neurone j sur la dernière couche cachée, (celle qui se place juste avant celle de sortie.), nous pouvons et en utilisant une formule analogue à 1.6 définir le gradient local par :

$$\begin{aligned}\delta_j(n) &= \frac{\partial \varepsilon(n)}{\partial y_j(n)} \times \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= \frac{\partial \varepsilon(n)}{\partial y_j(n)} \times f_j'\left(v_j(n)\right)\end{aligned}$$

Et d'après 1.2 on peut écrire :

$$\begin{aligned}\frac{\partial \varepsilon(n)}{\partial y_j(n)} &= \sum_k e_k(n) \times \frac{\partial e_k(n)}{\partial y_j(n)} \\ &= \sum_k e_k(n) \times \frac{\partial e_k(n)}{\partial v_k(n)} \times \frac{\partial v_k(n)}{\partial y_j(n)}\end{aligned}$$

Or, le neurone k est sur la couche de sortie, d'où :

$$\begin{aligned}\frac{\partial e_k(n)}{\partial v_k(n)} &= \frac{\partial \left(d_k(n) - y_k(n)\right)}{\partial v_k(n))} \\ &= \frac{\partial \left(d_k(n) - f_k\left(v_k(n)\right)\right)}{\partial v_k(n)} \\ &= f_k'\left(v_k(n)\right)\end{aligned}$$

En outre, et d'après 1.4 :

$$\frac{\partial v_k(n)}{\partial y_j(n)} = \omega_{kj}(n)$$

Donc nous obtenons :

$$\begin{aligned}\frac{\partial \varepsilon(n)}{\partial y_j(n)} &= -\sum_k e_k(n) \times f_k'\left(v_k(n)\right) \times \omega_{kj}(n) \\ &= -\sum_k \delta_k(n) \times \omega_{kj}(n)\end{aligned}$$

Le gradient local pour un neurone de la dernière couche cachée est alors donné par :

$$\delta_j(n) = \left(\sum_k \delta_k(n) \times \omega_{kj}(n)\right) \times f_j'\left(v_j(n)\right)$$

Ceci peut se généraliser et sans aucun souci aux autres couches internes, nous obtenons ainsi les règles de mise à jour des poids et des biais avec la méthode de rétro-propagation du gradient.

Pour cette raison, les poids et les biais doivent être actualisés en utilisant :

$$\omega_{ji} \leftarrow \omega_{ji} + \eta \times \delta_j(n) \times y_i(n)$$
$$b_j \leftarrow b_j - \eta \times \delta_j(n)$$

Avec :

$$\delta_j(n) = e_j(n) \times f'_j(v_j(n))$$

Et :

$$e_j(n) = \begin{cases} d_j(n) - y_j(n) & \text{si } j \text{ est un neurone de sortie.} \\ \sum_k \delta_k(n) \times \omega_{kj}(n) & \text{si } j \text{ est un neurone interne.} \end{cases}$$

Ainsi, l'erreur de sortie se propage vers l'entrée (c'est à dire dans le sens inverse de celui qui est utilisé pour propager un signal à travers le réseau) afin de corriger de couche en couche les valeurs des poids et des biais.

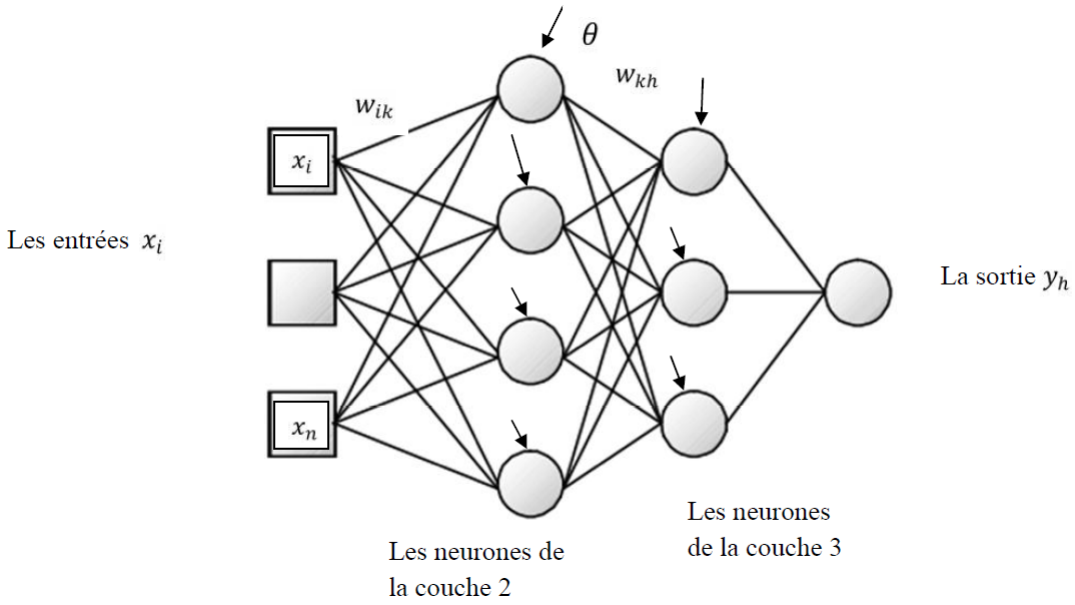


FIGURE 1.11 – Architecture de PMC

1.11 Intérêts et limites des réseaux de neurones

1.11.1 Intérêts des réseaux de neurones

On peut trouver plusieurs intérêts des réseaux de neurones[5] :

- Souplesse : les réseaux de neurones sont capables de traiter une gamme très étendue de problème. Leur résultat peut être une régression, une classification ou encore une analyse de clusters (clustering neuronale).

- Bonne résolution : ils donnent de bons résultats même dans des domaines complexes car ils sont beaucoup plus puissants que les statistiques ou les arbres de décisions.

- Bonne adaptation : une fois que les données sont codées, ils traitent aussi bien des variables continues qu'énumératives.

- Parallélisme massif : les réseaux de neurones sont constitués d'unités de calcul qui peuvent opérer d'une manière parallèle. La plupart des implémentations des réseaux de neurones peuvent être facilement converties d'une version séquentielle à une version parallèle.

- Outils disponibles : il existe de nombreux produits sur le marché intégrant la technique des réseaux de neurones.

1.11.2 Limite des réseaux de neurones

Nous pouvons facilement dégager les limites de l'approche,des réseaux de neurones pour la résolution des problémes [5][8] :

- Codage des entrées : toutes les entrées d'un réseau de neurone doivent se trouver dans un intervalle défini en général entre 0 et 1, ce qui entraine des transformations qui impliquent des traitements supplémentaires, et risque de fausser les résultats.

- Détermination de la taille : afin que l'échantillon fournisse de bons résultats, sa taille doit être calculée en fonction du nombre d'entrée, du nombre des couches et de taux de connexion ce qui entraine une augmentation du nombre d'exemples qui ne sont pas toujours disponibles.

- Performance : le nombre de calcule à effectuer pour définir un réseau optimal peut être très consommateur de puissance, ce qui peut donner de mauvaises performances à cette technique.

- La faiblesse des capacités d'explication des résultats.

- la sensibilité des RNA a certains paramètres, qui fond que le comportement global est susceptible de changer drastiquement à cause d'une perturbation mineure.

1.12 Domaines d'application des réseaux de neurones

On peut trouver de nombreux exemples d'utilisations des réseaux de neurones.

- **Traitement d'image** : reconnaissance de caractères et de signatures, compression d'image,

reconnaissance de forme, cryptage, classification.

- **Traitement de signal :** filtrage, classification, identification de sources, traitement de la parole.

- **Contrôle :** commande de processus, diagnostic de pannes, contrôle qualité, robotique.

- **Optimisation :** planification, allocation de ressources, tournées de véhicules, régulation de trafic, gestion, gestion, finance.

- **Simulation :** simulation boite noir, prévision météorologique, recopie de modèles.

1.13 Conclusion

Les réseaux de neurones formels possèdent une propriété remarquable, qui est l'apprentissage, qui est à l'origine de leurs intérêts dans des domaines et des applications très divers.

Un réseau de neurones se distingue, en général, par le type de neurone formel qu'il utilise, la règle d'apprentissage qui le décrit et l'architecture définissant les interconnexions entre les neurones.

Pour un problème posé, le choix du type de réseau de neurones, a utilisé, est basé sur les points essentiels suivants : le nombre de couches, le nombre de neurones par couches, la fonction d'activation, le type d'apprentissage, le choix et la préparation de données utilisées pour l'apprentissage.

Chapitre 2

La machine d'apprentissage extrême :Le cas d'un réseau RBF

2.1 Introduction

Les réseaux neuronaux (NN) et les machines vectorielles de soutien (SVM) jouent un rôle clé dans l'apprentissage automatique et l'analyse de données. Cependant, il est connu qu'il existe des problèmes difficiles avec eux tels que l'intervention humaine intensive, la vitesse d'apprentissage lente, la faible évolutivité de l'apprentissage. Cette présentation lance une nouvelle technique d'apprentissage appelée Extreme Learning Machine (ELM). ELM apprend non seulement des dizaines de milliers de fois plus vite que NN et SVM, mais fournit également une implémentation unifiée pour les applications de régression, binaires et multi-classes. ELM est efficace pour les séries temporelles, les applications séquentielles en ligne et incrémentales. ELM est efficace pour les grands ensembles de données.

Dans ce chapitre, nous allons vous présenter ce réseau (ELM) qui a été proposée par **Guang-Bin Huang** en 2006

2.2 Calcul l'inverse généralisé Moore-Penrose

En mathématiques, et en particulier l'algèbre linéaire. Pseudo inverse d'une matrice est une généralisation de la matrice inverse, le type de matrice pseudo inverse le plus largement connu est le pseudo inverse de Moore-Penrose, qui a été décrit indépendamment par **EH Moore** en 1920[6], **Arne Bjerhammar** en 1951[5] et **Roger Penrose** en 1955[6]. Le terme inverse généralisé est parfois utilisé comme synonyme de pseudo inverse.

Notation et Définition 2.1 *: Une matrice G d'ordre $n * m$ est l'inverse généralisé de la matrice de Moore-Penrose A d'ordre $m * n$ si :*
$AGA = A,\ GAG = G,\ (AG)^T = AG,\ (GA)^T = GA$
pour les raisons de commodité l'inverse généralisé de la matrice A de Moore-Penrose sera dénoté par $A^\dagger$.

Définition 2.2 *$x_0 \in \mathbb{R}^n$ est considéré comme une solution minimale des moindres carrés d'un système linéaire général $Ax = y$ avec $y \in \mathbb{R}^n$ si :*

$\|x_0\| \leqslant \|x\|,\ \forall x \in \{x : \|Ax - y\| \leqslant \|Az - y\|,\ \forall z \in \mathbb{R}^n\}$ *avec* $\|.\|$ *est la norme euclidienne de l'espace.*

Dans autre manière , x_0 *est la solution minimale des moindres carrés d'un système linéaire général* $Ax = y$ *si elle a la plus petite norme parmi toutes les solutions des moindres carrés.*

Lemme 2.3 $x_0 \in \mathbb{R}^n$ *est une solution minimale des moindres carrés d'un système linéaire général* $Ax = y$ *avec* $y \in \mathbb{R}^n$ *si et seulement si* x_0 *est une solution de l'équation normale suivante :*

$$A^T A x_0 = A^T y$$

.

Preuve 2.4 *:* $\Rightarrow$*) Supposons que* x_0 *est une solution minimale des moindres carrés d'un système linéaire* $Ax = y$ *alors :*

$$Ax_0 = P_W y$$

avec : $W = \{x^T Ax, x \in \mathbb{R}^n, x^T x = 1\}$ *et* P_W *est la projection orthogonale sur* W *Donc :*

$$y - P_W y \in W^{\perp}$$

Ainsi :

$$< a, y - P_W y >= 0 \quad \forall a \in W$$

En particulier :

$$< Ae_i, y - P_W y >= 0$$

avec e_i *la* $i^{\text{ème}}$ *colonne de* A. $(Ae_i)^T(y - P_W y) = 0 \quad \forall i \in 1, 2, ..., n$
alors :

$$A^T(y - P_W y) = 0$$

$$(A)^T y = (A)^T P_W y$$

et d'aprés l'hypothése on a :

$$(A)^T y = (A)^T A x_0$$

Donc x_0 *est une solution de l'équation normale.*
$\Leftarrow$*) suppposons que* x *est une solution de l'équation normale ,alors*

$$A^T A x = A^T y$$

donc :

$$A^T(Ax - y) = 0$$

d'ou :

$$Ax - y \in W^{\perp}$$

Or y a une seule décomposition donc : $y = w + v$ avec : $w \in W$ et $v \in W^{\perp}$ Donc $Ax - y = v$ alors $y = Ax_0 - v = Ax_0 - (Ax - y) = P_W y - Ax + y$ En fin :

$$Ax = P_W y$$

Donc x est une solution minimale des moindres carrés.

Lemme 2.5 *Pour toute matrice $A \in \mathbb{R}^{m*n}$ alors $AA^{\dagger}$ est appelée la matrice de la projection orthogonale.*

Preuve 2.6 *: Soit $A \in \mathbb{R}^{m*n}$,alors $A^{\dagger}$ reste une inverse généralisée de la matrice A,donc $AA^{\dagger}$ est une matrice de la projection ,de plus $A^{\dagger}$ est une pseudo-inverse de A,alors $AA^{\dagger}$est une matrice symétrique.En fin $AA^{\dagger}$ est une matrice de la projection orthogonale.*

Corollaire 2.7 *: Soit $A \in \mathbb{R}^{m*n}$ la matrice de l'inverse généralisée de Moore-Penrose,alors Les conditions de la matrice de l'inverse généralisée de Moore-Penrose sont équivalentes aux conditions suivantes :*

$$AG = P_A \, et \, GA = P_G$$

avec P_A et P_G sont les projections orthogonaux de A et G.

Preuve 2.8 *: il suffit de remarquer que $P_A^2 = P_A$ et $P_A^T = P_A$ et d'une maniére analogue pour la matrice G.*

Théorème 2.9 *soit une matrice G telle que Gy est une solution minimale standard des moindres carrés d'un système linéaire $Ax = y$.Alors ,il faut et il suffit que $G = A^{\dagger}$ est une matrice de l'inverse généralisée de Moore-Penrose de la matrice A.*

Preuve 2.10 *: $\Rightarrow$) Supposons que $z = Gy$ est une solution minimale des moindres carrés ,donc :*

$$z = A^{\dagger} y$$

alors :

$$Az = AA^{\dagger} y$$

donc :

$$AGy = P_A y$$

en fin :

$$AG = P_A$$

d'autre part

$$z = A^{\dagger} y$$

donc :

$$Gy = A^{\dagger} Ay$$

alors :

$$GAx = P_{A^\dagger}x$$

En fin :

$$GA = P_{A^\dagger}$$

alors d'aprés le corollaire précedente :A est la matrice de l'inverse généralisée de Moore-Penrose de la matrice A.

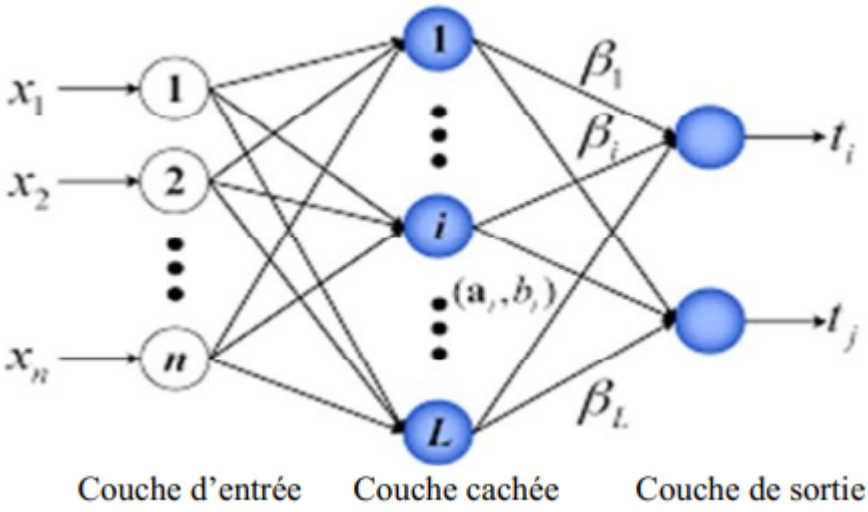
$\Leftarrow$) *Supposons que* $A^\dagger = G$ *,d'aprés le premier lemme de ce chapitre il suffit de montrer que z est une solution de l'équation :*$A^TAx = A^Ty$

$$A^TAz = A^TAGy = A^TAGAx = A^TAx = A^Ty$$

car :$G = A^\dagger$ *donc z vérifie l'équation donc z est une solution minimale des moindres carrés.*

2.3 Architecture du réseau ELM

Le réseau ELM est un réseau de type multicouche feed-forward avec une seule couche cachée comme représenté sur la figure suivante :



La couche d'entrée permet d'introduire les données à l'intérieur du réseau, le nombre de neurones dans cette couche, dépend de la dimension du vecteur de caractéristiques qui décrit les données d'entrée en question. Concernant le nombre de neurones de la couche de sortie, dans le cadre de classification, est égal au nombre de classes. Reste maintenant le nombre de neurones de la couche cachée, nous étudierons, dans les parties suivantes, son influence sur le taux de la classification.

2.4 Fonctionnement du réseau ELM pour SLFN

Pour N échantillons (x_i, t_i) *, où* $x_i = [x_{i1}, x_{i2}, ...x_{in}]^T \in \mathbb{R}^n$ *et le vecteur de caractéristiques de l'iéme échantillons de dimension n et* $x_i = [x_{i1}, x_{i2}, ...x_{in}]^T \in \mathbb{R}^n$ *, est le vecteur de valeurs*

d'apprentissage de l'iéme échantillons ou m classes. (x_i, t_i) *et la fonction d'activation* $g(x)$ *sont mathématiquement des modèles comme :* $\sum_{i=1}^{M} \beta_i * g(w_i * x_j + b_i) = \Theta_j$ *pour tout* $j = 1...N$
L'équation ci-dessus (1) peut être écrite de manière suivante : $H\beta = T$ *tel que :*

$$H(w_1, w_2...w_M, b_1, b_2...b_M x_1, x_2...x_N) = \begin{bmatrix} g(w_1.x_1 + b_1) & \dots & g(w_M.x_1 + b_M) \\ \vdots & \dots & \vdots \\ g(w_1.x_N + b_1) & \dots & g(w_M.x_N + b_M) \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \cdots \\ \beta_M^T \end{bmatrix} \qquad et \qquad T = \begin{bmatrix} t_1^T \\ \cdots \\ t_M^T \end{bmatrix}$$

Comme indiqué par Huang et al [5],H s'appelle la matrice de sortie de couche cachée du réseau neuronal. la $i^{éme}$ *colonne de* H *est la* $i^{éme}$ *neurone de la couche cachée avec le respect des entrées* $x_1...x_N$.

2.4.1 Algorithme ELM pour SLFN

Considérons le jeu d'apprentissage $\lambda = \{(x_i, t_i) \quad avec \quad (x_i, t_i) \in \mathbb{R}^n \times \mathbb{R}^m \quad pour\ tout\ i = 1...N\}$. *et* $g(x)$ *la fonction d'activation et* M *le nombre de neurone de la couche cachée.Alors algorithme peut être résumé en ces étapes suivantes :*

Etape1 *:affectez de manière aléatoire les poids d'entrée* W_i *et le bais* $b_i \quad i = 1...M$

Etape2 *: calculer la matrice de sortie de la couche cachée* H.

Etape3 *:calculer le poids de sortie* β*,tel que* $\beta = H^{\dagger} * T$.

Contrairement à la compréhension la plus commune, tous les paramètres des SLFN doivent être ajustés, les poids d'entrée w_i *et le biais* b_i *ne sont, en fait, pas nécessairement réglés et la matrice de sortie de couche cachée* H *peut effectivement rester inchangée, une fois que les valeurs aléatoires ont été attribuées à ces paramètres au début de l'apprentissage.*

Cependant, dans la plupart des cas, le nombre de nœuds cachés est beaucoup moins élevé que le nombre d'échantillons de formation, H *est une matrice non carrée et elle ne peut d'être inverser, un tel système* $H\beta = T$ *n'admet pas de solution. La seule solution acceptable à ce problème est de chercher la solution de la norme minimale des moindres carrés qui est donnée par la relation suivante :*

$$\hat{\beta} = \min_{\beta} \|H\beta - T\| \tag{2.1}$$

La solution de la norme minimale des moindres carrés est l'unique solution parmi toutes celles qui réalisent le minimum$\|H\beta - T\|$. *Cette solution de la norme minimale est calculée en utilisant la relation suivante :*

$$\hat{\beta} = H^{\dagger} * T \tag{2.2}$$

Avec $H^{\dagger}$ *est l'inverse généralisé de Moore-Penrose de la matrice* H

2.5 La Machine d'apprentissage extrême à RBF

Dans cette section ,nous montrons que l' ELM précédemment proposé pour SLFN peut être étendue de façon linéaire aux réseaux RBF.

2.5.1 Problème d'approximation de RBF

La sortie d'un réseau RBF avec des M noyaux pour un vecteur d'entrée $x \in \mathbb{R}^n$ est donnée par : $f_M(x) = \sum_{i=1}^{M} \beta_i * \phi_i(x)$ *avec :* $\beta_i = [\beta_{i1}, \beta_{i2}, ...\beta_{in}]^T \in \mathbb{R}^n$ *est le vecteur de poids reliant le* $i^{éme}$ *noyau et les neurones de la sortie et* $\phi_i(x)$ *et la sortie du* $i^{éme}$ *noyau qui est habituellement gaussien :*

$$\phi_i(x) = \phi(\mu_i, \sigma_i, x) = exp(\frac{\| x - \mu_i\|^2}{\sigma_i}) \tag{2.3}$$

$\mu_i = [\mu_{i1}, \mu_{i2}, ...\mu_{in}]^T \in \mathbb{R}^n$ *est le* $i^{éme}$ *centre du noyau et* σ_i *sa largeur d'impact.*
Pour N *exemples distincts arbitraires* (x_i, t_i),*avec* $x_i = [x_{i1}, x_{i2}, ...x_{in}]^T \in \mathbb{R}^n$ *et* $t_i = [t_{i1}, t_{i2}, ...t_{in}]^T \in \mathbb{R}^n$ *. RBF avec* M *noyaux peut être modélisé mathématiquement comme suit :*
$\sum_{i=1}^{M} \beta_i * \phi_i(x_j) = \Theta_j \qquad j = 1...N$
Comme dans le cas de SLFN ,les RBFs standards avec M *noyaux peuvent se rapprocher de ces* N *exemples avec zero erreur cela signifie que*

$$\sum_{j=1}^{M} \|\Theta_j - t_j\| = 0 \tag{2.4}$$

C'est à dire qu'il existent $\beta_i, \mu_i, et \sigma_i$ *tels que :*

$$\sum_{i=1}^{M} \beta_i * exp(\frac{\| x_j - \mu_i\|^2}{\sigma_i}) = t_j \quad pour \quad j = 1...N \tag{2.5}$$

Les N *équations peuvent s'écrire sous forme matricielle suivante :*
$H\beta = T$
avec :

$$H(\mu_1, \mu_2, ..\mu_M, \sigma_1, ..., \sigma_M, x_1, ..., x_N) = \begin{bmatrix} \phi(\mu_1, \sigma_1, x_1) & \dots & \phi(\mu_M, \sigma_M, x_1) \\ \vdots & \dots & \vdots \\ \phi(\mu_1, \sigma_1, x_N) & \dots & \phi(\mu_M, \sigma_M, x_N) \end{bmatrix} \tag{2.6}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \cdots \\ \beta_M^T \end{bmatrix} \qquad et \qquad T = \begin{bmatrix} t_1^T \\ \cdots \\ t_M^T \end{bmatrix} \tag{2.7}$$

Comme dans le cas de SLFNs ;H *est appelée la matrice sortie de la couche cachée du réseau RBF .*
La $i^{éme}$ *colonne de* H *est la sortie de le* $i^{éme}$ *noyau par rapport aux entrées* $x_1...x_N$.

2.5.2 Solution minimale des moindres carrés de RBF

Dans la plupart des cas, le nombre de noyaux M est bien inférieur au nombre des exemple d'apprentissage distincts N, donc H est une matrice non carrée et μ_i, σ_i, β_i ne peuvent pas exister pour tout $i = 1...M$ tel que $H\beta = T$.Ainsi,on doit chercher les $\hat{\mu}_i, \hat{\sigma}_i, \hat{\beta}_i$ spécifiques tel que :

$$\|H(\hat{\mu}_1, \hat{\mu}_2...\hat{\mu}_M, \hat{\sigma}_1, \hat{\sigma}_2...\hat{\sigma}_M)\hat{\beta} - T\| = \min_{\mu_i,\sigma_i,\beta} \|H(\mu_1, \mu_2...\mu_M, \sigma_1, \sigma_2...\sigma_M)\beta - T\| \tag{2.8}$$

Ce qui équivalent de minimiser la fonction cout :

$$E = \sum_{j=1}^{N}(\sum_{i=1}^{M} \beta_i * g(\mu_i, \sigma_i, x_j) - t_j)^2$$

Traditionnellement, lorsque H est inconnu en fonction du gradient les algorithmes d'apprentissage sont généralement utilisés pour chercher le minimum de $\|H\beta - T\|$.
Dans la procédure de la minimisation en utilisant des algorithmes basés sur gradient,vecteur W,qui est l'ensemble des centres des noyaux μ_i et l'impact largeur σ_i et les poids de sortieβ_i
Ainsi l'ajustement se fait comme suit :

$$W_k = W_{k-1} - \eta * \frac{\partial E(W)}{\partial W} \tag{2.9}$$

2.5.3 Algorithme d'ELM pour RBF

Soit l'ensemble d'apprentissage $\lambda = \{(x_i, t_i) \quad avec \quad (x_i, t_i) \in \mathbb{R}^n * R^m \quad pour\ tout \quad i = 1...N\}$.et M le nombre des noyaux alors :
***Etape 1** :Attribuer arbitrairement les μ_i et σ_i $\quad i = 1...M$*
***Etape 2** :calculer la matrice de sortie H de la couche cachée*
***Etape 3** :calculer le poids de sortie β, $\quad \beta = H^{\dagger} * T$*

2.6 conclusion :

Après la description et l'étude du réseau ELM proposé par Huang, nous pouvons conclure, que tous les paramètres de la couche cachée (ai et bi) de l'ELM pourraient être générés de manière aléatoire et restent constants, selon une distribution de probabilité continue donnée, sans aucune connaissance préalable (même avant que les données ne soient présentées et que l'apprentissage ELM commence). Donc ces paramètres sont indépendants des données de l'apprentissage, ce qui rend le réseau ELM un approximateurs universel. Vu que seul les poids de sortie sont calculés par une simple régression linéaire, implique que ce réseau a un temps de réponse beaucoup plus court. Dans le chapitre suivant, nous allons évaluer le réseau ELM sur un ensemble de bases de données

Chapitre 3

Performance et l'évaluation :

3.1 Introduction

Dans ce chapitre nous avons basé sur la mise en place d'un réseau RBF pour trois catégories des problèmes, le premier est la comparaison des problèmes d'approximation des fonctions artificielles ,et le deuxième consiste à l'analyse comparative avec des problèmes d'approximation des fonctions dans le monde réel ,et le dernier s'interesse à l'analyse comparative avec les applications de classification du monde réel. chaque catégorie se compose de deux exemples .

3.2 Comparaison des problèmes d'approximation des fonctions artificielles

3.2.1 Approximation de la fonction Friendmann

3.2.1.1 Introduction

Dans cet exemple, ELM-RBF et SVM sont utilisés pour approcher trois fonctions populaires de Friedmann :

$$Friendmann1: \quad Y(x) = 10sin(\pi \times x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$$

$$Friendmann2: \quad Y(x) = (x_1^2 + (x_2 x_3 - (1/(x_2 x_4))^2)^{1/2})$$

$$Friendmann3: \quad Y(x) = arctan(\frac{x_2 x_3 - 1/(x_2 x_4)}{x_1})$$

Pour la premiére fonction $0 \leqslant x_i \leqslant 1$*,mais pour les deux dernières on a :*

$$0 \leqslant x_1 \leqslant 100$$

$$40\pi \leqslant x_2 \leqslant 560\pi$$

$$0 \leqslant x_3 \leqslant 1$$

$$0 \leqslant x_4 \leqslant 11$$

3.2.1.2 Simulation

Dans cette application on a choisi 50 essais chacun avec des données d'entraînement et de test générées aléatoirement ont été réalisés pour tous les algorithmes.Pour chaque essai, 1000 données d'entraînement et 1000 données de test sont générées aléatoirement pour les trois fonctions de Friendman, respectivement.Afin de trouver les paramètres appropriés pour SVM, 225 combinaisons de paramètres de coût C et de paramètres du noyau γ et 10 répétitions pour chaque combinaison ont été essayées, qui ont passé plus de 7 heures. D'après les simulations, les performances de la SVM peuvent être sensibles au paramètre de coût C et au paramètre noyau γ Par exemple, pour Friedman 2, si nous définissons $C = 2$ et γ comme valeur par défaut, la prédiction RMSE de SVM pourrait être $261,3830$*, ce qui est similaire aux résultats obtenus par Drucker, et. al[14] mais beaucoup plus grand que* $2,7769$ *obtenu dans nos simulations puisque nous essayons de trouver la meilleure performance de SVM et de la comparer avec la ELM-RBF proposée ,les tableaux suivants indiqués les résultats obtenus pour les trois fonctions de friedmann :*

Friedmann 1 :

Algorithme	*temps d'entrainement*	*la moyenne de RMS*	*Ecart type de RMS*	*N.de noyaux*
ELM-RBF	*0,0112*	*0,5812*	*0,0091*	*10*
SVR$(2^{-1}, 2^{-7})$	*0,6352*	*0,5847*	*0,0073*	*901,2*

Friedmann 2 :

Algorithme	*temps d'entrainement*	*la moyenne de RMS*	*Ecart type de RMS*	*N.de noyaux*
ELM-RBF	*1,0121*	*2,705*	*0,1663*	*160*
SVR$(2^{12}, 2^{-2})$	*776,64*	*2,7769*	*0,3342*	*885,5*

Friedmann 3 :

Algorithme	*temps d'entrainement*	*la moyenne de RMS*	*Ecart type de RMS*	*N.de noyaux*
ELM-RBF	*0,0962*	*0,1084*	*0,0108*	*50*
SVR$(2^{10}, 2^{-6})$	*11,8560*	*0,1091*	*0,0071*	*178,2*

3.2.1.3 Conclusion

ELM-RBF peut obtenir des résultats aussi bons que SVM pour ces trois fonctions Friedman avec beaucoup moins de noyaux, mais apprend jusqu'à des centaines de fois plus vite que SVM.

3.2.2 Approximation de la fonction "Sin(C)" avec bruit

3.2.2.1 Introduction

Dans cet exemple, ELM-RBF et SVM sont utilisés pour rapprocher la fonction 'Sin(C)', un choix populaire pour illustrer Support Vector Machine for Regression dans la littérature :

$$y(x) = \begin{cases} sin(x)/x, & x \neq 0 \\ 1, & x = 0 \end{cases}$$

Un ensemble d'entraînement (x_i, y_i) *et un ensemble de test* (x_i, y_i) *avec* 5000 *données respectivement sont créés où les* $(x_i$ *sont uniformément répartis aléatoirement sur l'intervalle* $[-10, 10]$*. Afin de rendre le problème de régression 'réel', le bruit uniformément réparti dans*

$[-2, 2]$ *a été ajouté à tous les échantillons de formation tout en testant les données restent sans bruit.*

3.2.2.2 Simulation

50 essais ont été réalisés pour tous les algorithmes ELM et SVM et les résultats moyens sont présentés dans le tableau suivant. Les données d'entrainement et de test sont générées au hasard pour chaque essai de simulation. Afin d'obtenir la meilleure performance de généralisation de SVM , plus de 6 *jours ont été consacrés à la recherche d'une combinaison appropriée :des paramètres de coût* C *et des paramètres du noyau* γ*. Par exemple, pour la combinaison de paramètres* $(C = 2^8, \gamma = 2^2)$*, le temps d'apprentissage est de* $1278,2$ *secondes et la précision des tests est de* $0,0072$*, le temps d'entrainement est beaucoup plus grand que celui indiqué dans le tableau suivant. Le temps d'entrainement passé pour la SVM avec la combinaison de paramètres* $(C = 2^{12}, \gamma = 2)$ *est de 15168,3 secondes,donc il est très grand,comme indiqué dans le tableau suivant :*

Algorithme	*temps d'entrainement*	*la moyenne de RMS*	*Ecart type de RMS*	*N.de noyaux*
ELM-RBF	*1,8652*	*0,00075*	*0,0011*	*100*
SVR$(2^3, 2^2)$	*68,0459*	*0,0067*	*0,0012*	*2502,0*

3.2.2.3 Conclusion

ELM-RBF peut obtenir des résultats aussi bons que SVM avec beaucoup moins de noyaux, mais apprend jusqu'à 40 fois plus vite que SVM.

3.3 Analyse comparative avec des problèmes d'approximation des fonctions dans le monde réel

3.3.1 prévision de logement en Californie

3.3.1.1 Introduction

California Housing est un ensemble de données obtenu à partir du dépôt StatLib. Il y a 20640 *observations pour prédire le prix des maisons en Californie. Les données sur les variables ont été recueillies en utilisant tous les groupes de blocs en Californie à partir du Recensement de* 1990*. Dans cette application, un groupe de blocs comprend en moyenne* 1425,5 *personnes vivant dans une zone géographiquement compacte. Naturellement, la zone géographique incluse varie inversement avec la densité de population. Les distances entre les centroïdes de chaque groupe de blocs ont été calculées en latitude et en longitude. Tous les groupes de blocs qui ont déclaré zéro entrée pour les variables indépendantes et dépendantes ont été exclus. Les données définitives contenaient* 20640 *observations sur* 9 *variables, soit* 8 *entrées continues (revenu médian, âge médian du logement, nombre total de chambres, nombre total de chambres, population, ménages, latitude et longitude) et une sortie continue (valeur médiane du logement).*

3.3.1.2 Simulation

Dans nos simulations, 50 *essais chacun avec des données d'entrainement et de test générées aléatoirement,les essais ont été effectués pour les algorithmes ELM-RBF et SVM, et* 8000 *données d'entrainement et* 12640 *données de test générées aléatoirement à partir de la base de données de California Housing pour chaque essai de simulation. La valeur de sortie est normalisée en* $[0, 1]$*. Afin d'obtenir les meilleures performances de généralisation de SVM comme indiqué dans le tableau suivant, plus de* 11 *jours ont été consacrés à la recherche d'une combinaison appropriée des paramètres de coût* C *et des paramètres du noyau* γ*. Par exemple, pour la combinaison des paramètres* $(C = 2^{12}, \gamma = 2^4)$*, le temps d'apprentissage est supérieur à* 14000 *secondes et la précision des tests est de* $0,2079$*, le temps d'entraînement est* 250 *fois le temps d'entraînement passé pour la SVM avec la combinaison de paramètres* $(C = 2^2, \gamma = 2^1)$ *comme indiqué dans le tableau suivant :*

Algorithme	*temps d'entrainement*	*la moyenne de RMS*	*Ecart type de RMS*	*N.de noyaux*
ELM-RBF	*71282*	*0,1265*	*0,0043*	*100*
SVR$(2^2, 2)$	*566582*	*0,1181*	*0,0011*	*2193,2*

3.3.1.3 Conclusion

ELM-RBF peut obtenir des performances de généralisation proches de la SVM avec beaucoup moins de noyaux et une vitesse d'apprentissage plus rapide.

3.3.2 Prédiction de l'âge de l'ormeau

3.3.2.1 Introduction

Le problème de l'ormeau compte 4177 *cas de prédiction de l'âge de l'ormeau à partir de mesures physiques. Chaque observation se compose de* 1 *entier et* 7 *attributs d'entrée continue et* 1 *entier de sortie. Il y a* 3 *valeurs pour le premier attribut entier : Homme, Femme et Nourrisson. Pour ce problème,*3000 *données d'entrainement et* 1177 *données de test sont générées aléatoirement à partir de la base de données Abalone pour chaque essai de simulation, comme on le fait habituellement dans la littérature. La valeur de sortie est normalisée en* $[0, 1]$*.*

3.3.2.2 Simulation

Afin d'obtenir les meilleures performances de généralisation de SVM comme indiqué dans le Tableau suivant, $1,5$ *jour a été consacré à la recherche d'une combinaison appropriée des paramètres de coûts* C *et des paramètres du noyau* γ*. Par exemple, pour la combinaison de paramètres* $(C = 2^{12}, \gamma = 2^1)$*, le temps d'apprentissage est supérieur à* 7000 *secondes et la précision des tests est de* $0,1178$*, le temps d'entraînement est* 160 *fois le temps d'entraînement consacré à la SVM avec la combinaison de paramètres* $(C = 2^{10}, \gamma = 2^{-6})$ *comme indiqué dans le tableau suivant.Nous essayons de trouver la meilleure performance de prédiction de SVMs par*

l'essai approfondi de la combinaison déférente de paramètres (C, γ),le tableau suivant indiqué les résultats trouvés :

Algorithme	*temps d'entrainement*	*la moyenne de RMS*	*Ecart type de RMS*	*N.de noyaux*
ELM-RBF	*0,0325*	*0,0779*	*0,0022*	*15*
SVR($2^{10}, 2^{-6}$)	*44,0474*	*0,0785*	*0,0023*	*457,83*

3.3.2.3 Conclusion

ELM-RBF peut obtenir des performances de généralisation aussi bonnes que SVM avec beaucoup moins de noyaux et une vitesse d'apprentissage beaucoup plus rapide.

3.4 Analyse comparative avec les applications de classification du monde réel

3.4.1 Application de diagnostic médical :diabète

3.4.1.1 Introduction

Le diabète est une maladie chronique qui apparaît lorsque le pancréas ne produit pas suffisamment d'insuline ou que l'organisme n'utilise pas correctement l'insuline qu'il produit. L'insuline est une hormone qui régule la concentration de sucre dans le sang. L'hyperglycémie, ou concentration sanguine élevée de sucre, est un effet fréquent du diabète non contrôlé qui conduit avec le temps à des atteintes graves de nombreux systèmes organiques et plus particulièrement des nerfs et des vaisseaux sanguins.

En 2014,la pourcentage de 8,5 de la population adulte (18 ans et plus) était diabétique. En 2015, le diabète a été la cause directe de 1,6 million de décès et en 2012 l'hyperglycémie avait causé 2,2 millions de décès supplémentaires.

La base de donnée pour notre application est "Le Pima Indian Diabetes", originaire de l'Institut national du diabète et des maladies digestives et rénales, contient des informations sur 768 femmes d'une population près de Phoenix, Arizona, USA. Les résultats étaient le diabète, 258 étaient positifs et 500 étaient négatifs. Par conséquent, il y a une variable objective (dépendante) et les 8 attributs : grossesses, OGTT (test de tolérance au glucose oral), tension artérielle, épaisseur de la peau, insuline, IMC (indice de masse corporelle), âge, fonction du diabète généalogique.

3.4.1.2 Simulation

Pour ce problème, 0,75 et 0,25 des échantillons sont choisis respectivement au hasard pour l'entrainement et le test à chaque essai. Les résultats moyens de plus de 50 *essais sont présentés dans le tableau suivant.Dans nos simulations, le SVM peut atteindre le taux d'essai* $0,777$ *avec* $294,07$ *vecteurs de soutien en moyenne, ce qui est mieux que le taux d'essai* $0,765$ *obtenu par Ratsch et al[11]. Pour cette application, le taux de test du réseau RBF obtenu par Wilson et*

Martinez[12] est de $0,763$*, tandis que l'algorithme d'apprentissage ELM-RBF obtient un résultat similaire. Contrairement aux autres applications, pour cette petite application de jeu de données, il ne faut que* 360 *secondes pour trouver la combinaison de paramètres appropriée de SVM pour chaque essai de combinaison de paramètres. Étant donné que le temps d'entrainement est très court, afin d'obtenir la meilleure combinaison de paramètres pour la SVM pour différents ensembles de données d'entrainement et de test,* 50 *répétitions peuvent être faites et ont été faites pour chaque combinaison de paramètres et la combinaison de paramètres est choisie ce qui produit la meilleure performance de généralisation en moyenne ,le tableau suivant présente les résultats obtenus :*

Algorithme	*temps d'entrainement(s)*	*le taux de test*	*Ecart type de test*	*N.de noyaux*
ELM-RBF	*0,0408*	*0,7648*	*0,281*	*30*
SVR$(2^{11}, 2^{-7})$	*0,9436*	*0,7770*	*0,294*	*294,07*

3.4.1.3 Conclusion

ELM-RBF peut obtenir des performances de généralisation proches de la SVM avec beaucoup moins de noyaux et une vitesse d'apprentissage plus rapide.

3.4.2 Image satellite Landsat :SatImage

3.4.2.1 Introduction

Les performances de l'ELM-RBF ont également été testées sur des applications multi-classes comme Landsat Satellite Image (SatImage) de la collection Statlog[13]. Landsat Satellite Image (SatImage) est une application de 7 classes qui possède 36 attributs d'entrée. À chaque essai de simulation de Satlmage, 4435 ensembles de données d'entrainement et 2000 ensembles de données de test sont générés aléatoirement à partir de leur base de données globale.

3.4.2.2 Simulation

Les résultats moyens de plus de 50 essais sont présentés dans le tableau suivant :

Algorithme	*temps d'entrainement(s)*	*le taux de test*	*Ecart type de test*	*N.de noyaux*
ELM-RBF	*8,1766*	*0,8801*	*0,0048*	*200*
SVR$(2^4, 2^0)$	*13,6979*	*0,9183*	*0,0048*	*1603,7*

3.4.2.3 Conclusion

ELM peut encore atteindre un bon rendement de généralisation, mais légèrement inférieur et proche des SVM.

3.5 Conclusion

Ce travail a étendu la machine d'apprentissage extrême (ELM) des réseaux neuronaux à une seule couche couche cachée (SLFN) à la fonction de base radiale (RBF). La principale ca-

ractéristique de la ELM proposée pour les réseaux RBF est qu'ELM assigne arbitrairement les noyaux au lieu de les accorder. Par rapport au SVM populaire, l'ELM proposé peut être utilisé facilement et l'ELM peut compléter la phase d'apprentissage à une vitesse très rapide et fournir un réseau plus compact. Comme l'ont montré quelques simulations sur des problèmes de référence réels et artificiels, ELM proposé pour les réseaux de RBF peut atteindre des performances de généralisation aussi bonnes que la SVM pour la régression et une performance de généralisation bonne mais légèrement inférieure à la SVM pour certains problèmes de classification. Il vaut la peine de poursuivre l'enquête systématique sur l'arbitraire des noyaux de RBF.

Bibliographie

[1] Marc Parizeau « Réseaux de Neurones. GIF- 21140 et GIF-64326 » Université LAVAL, 2006.

[2] Pierre Bome, Mohamed Benrejeb et Joseph Haggége « Les Réseaux de Neurone » TECHNIP, 2007.

[3] Léon PERSONNAZ et Isabelle RIVALS « Réseaux de neurones formels pour la modélisation la commande et la classification », CNRS Paris, 2003

[4] Dreyfus G., Martinez J.M., Samuelides M., Gordon M.B., Badran F., Thiria S. et Hérault L., « Réseaux de neurones Méthodologie et applications », EYROLLES, 2004.

[5] l'article de base extrême Learning machine théorie and application par « Guang-Bin Huang, Qin-yu zhu et Kheong siew » en 2006.

[6] Hervé Abdi et Dominique Valentin « Mathématique pour les sciences cognitives avec des applications aux réseaux de neurones au traitement de signal à l'imagerie cérébrale et à la statistique » Presses Université de Grenoble, 2006.

[7] Mathhew F.Dixon Igor Halperin Paul Bilokon, Machine Learning in finance From theory to practice..

[8] Mehryar Mohri,Afshin Rostamiszadeh,and Ameet Talwalkar, Foundations of Machine learning *second edition.*

[9] Shai Shalev-Shwartz and Shai Ben-David , Understand Machine learning from theory to algorithms.

[10] Claude TOUZET « Les Réseaux de Neurone Artificiel Introduction au connexionnisme » HAL, 1992.

[11] G. Ratsch, T. Onoda, and K. R. Miiller, "An iJnprovement of AdaBoost to avoid overfitting," in Proceedings of the 5th International Conference on Neural Information Processing (ICONIP' 1 998) , 1998.

[12] D. R. Wilson and T. R. Martinez, "Heterogeneous radial basis funct ion networks," in Proceedings of the International Conference on Neural Networks (ICNN 96), pp. 1263-1267, June 1996.

[13] C. Blake and C. Merz, "UCI repository of machine , lenrning databases," in http ://www.ics.ucLedu/ :"'mlearn/MLRepository.html, Department of Information and Computer Sciences, University of California, Irvine, USA, 1998.

[14] H. Drucker, C. J. B urges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in Neural Information Processing Systems 9 (M. Mozer, J. Jordan, and T. Petscbe, eds.), (MIT Press), pp. 1 55- 161 , 1997.