كلية العلوم و التقنيات فاس
aculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
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)

Approche bayésienne et apprentissage des RNA basée sur la simulation des variables aléatoires : Application à la reconnaissance des visages

Réalisé par: BAMOU Salah eddine

Encadré par: Pr. ETTAOUIL Mohamed
Pr. JOUDAR Nour-eddine

Soutenu le 15 Juillet 2021

Devant le jury composé de:

- Pr. ETTAOUIL Mohamed — FST Fès
- Pr. MOUHCINE Zakariyae — FST Fès
- Pr. EL KHAOULANI EL IDRISSI Rachid — FST Fès
- Pr. JOUDAR Nour-eddine — ENSET Rabat

Année Universitaire 2020 / 2021

FACULTE DES SCIENCES ET TECHNIQUES FES – SAISS

B.P. 2202 – Route d'Imouzzer – FES

DÉDICACE

Avec les sentiments d'amour et de gratitude les plus sincères
Je dédie ce modeste ce travail.

Á

Mes chers parents, en témoignage de ma fidèle affection pour l'amour que vous m'avez toujours réservé et de ma reconnaissance pour vos sacrifices et vos encouragements. Que dieu vous bénisse et vous procure une longue vie pleine de joie ;

Á

Mes chers frères et mes chères sœurs que dieu les protège ;

Á

Tous mes amis chacun par son nom ;

Á

Tous ceux et celles qui me sont chers.

REMERCIEMENT

Je tiens à témoigner ma reconnaissance à Dieu tout puissant, de m'avoir donné le courage et la force de mener à terme ce mémoire. Qui m'a ouvert les portes du savoir.

Je tiens à remercier chaleureusement mes encadrants Monsieur **Mohamed ETTAOUIL**, et Monsieur **Nour-eddine JOUDAR**, pour le temps qu'ils ont consacré pour m'avoir apporté les outils méthodologiques indispensables à la conduite de ce mémoire et qui m'ont offert la possibilité de réaliser ce travail dans de bonnes conditions, et aussi pour leurs bénéfiques conseils et explications, j'ai eu l'honneur de travailler sous leur direction.

Sans oublier à remercier mes camarades de stage BOUFSSASSE Ali, ELANSARI Taoufik, EL-AOUNI Abderazzak et EL MACHOUTI Sana qui m'ont rendu ce travail assez spécial.

Je remercie tous les enseignants qui ont assuré ma formation, et qui sont présentés aujourd'hui comme membre de jury afin d'évaluer mon travail, Monsieur **EL KHAOULANI EL IDRISSI Rachid**, et **Monsieur MOUHCINE Zakariyae**.

Enfin, je voudrais remercier ma famille et en particulier ma mère et mon père qui m'ont donné l'éducation sans laquelle je n'en serais pas là aujourd'hui.

ABRÉVIATIONS

RNA : Réseaux de neurones artificiels.

V.A : Variable aléatoire.

PMC : Perceptron Multicouches.

MCMC : Markov chain Monte Carlo

i.i.d : indépendantes identiquement distribuées.

FDP : Fonction Densité de Probabilité.

GLCM : Grey Level Co-occurrence Matrix.

ORL : Olivetti Research Laboratory.

TABLE DES MATIÈRES

Dédicace **1**

Remerciement **2**

Abréviations **3**

Table des matières **6**

Liste des algorithmes **7**

Table des figures **8**

Liste des figures **8**

Liste des tableaux **9**

Liste des tableaux **9**

Introduction générale **11**

1 Introduction aux réseaux de neurones artificiels **12**

1.1 Introduction 12

1.2 Modèles d'un neurone 12

1.2.1 Neurones biologiques 13

1.2.2 Neurones Formels 14

1.3 Structure des réseaux de neurones 16

1.3.1 Réseaux de neurones feedforward ou non bouclés 16

1.3.2 Réseaux de neurones récurrents 16

1.4 Propriété d'approximation universelle 16

1.4.1 Propriété d'approximation universelle 17

1.4.2 Propriété de parcimonie 17

1.5 Apprentissage des réseaux de neurones 18

1.5.1 Apprentissage supervisé 18
1.5.2 Apprentissage non supervisé 18
1.6 Modélisation d'un neurone artificiel 18
1.7 Modèles des réseaux de neurones 20
1.7.1 Le modèle de Perceptron 20
1.7.2 Le modèle de Perceptron Multicouches 21
1.7.3 Le modèle de Adaline 21
1.8 L'apprentissage du Perceptron Multicouches : L'algorithme de Rétro-propagation du gradient 22
1.9 Problème de sur-ajustement 26
1.9.1 L'idée centrale du réseau de neurones 26
1.9.2 Méthodes pour limiter le sur-ajustement 27
1.10 Conclusion 28

2 Méthodes MCMC pour la simulation des variables aléatoires 29
2.1 Introduction 29
2.2 Vision statistique et Bayésienne de l'apprentissage 29
2.2.1 Apprentissage statistique 29
2.2.2 Statistique Bayésienne 32
2.3 Méthodes classiques d'approximation 35
2.3.1 Intégration numérique 35
2.3.2 Méthodes de Monte Carlo 35
2.4 Méthode de Monte Carlo par chaînes de Markov 36
2.4.1 Introduction aux chaînes de Markov 37
2.4.2 Principe des méthodes MCMC 40
2.5 Conclusion 43

3 Intégration des méthodes MCMC dans l'apprentissage des RNAs 44
3.1 Introduction 44
3.2 Introduction à l'approche bayésienne 44
3.2.1 Principe de l'approche bayésienne 44
3.2.2 Avantages de l'approche bayésienne 45
3.2.3 Inconvénients de l'approche bayésienne pour les RNA 46
3.3 Procédure d'évidence 46
3.3.1 Distribution a priori des poids 46
3.3.2 Fonction de vraisemblance 47
3.3.3 Distribution a posteriori des paramètres 47
3.3.4 Approximation gaussienne de la distribution a posteriori 48
3.3.5 Structure d'évidence 49
3.3.6 Sélection du modèle bayésien 51
3.3.7 Procédure d'évidence pour un problème de classification 52
3.4 Intégration des méthodes MCMC dans l'apprentissage des RNA 53
3.5 Conclusion 55

4 Application à la reconnaissance des visages **57**
4.1 Introduction 57
4.2 La biométrie 57
4.2.1 Biométrie du visage 58
4.2.2 Principales difficultés de la reconnaissance de visage 60
4.2.3 Applications des systèmes biométriques 62
4.2.4 L'extraction des paramètres 63
4.3 Traitement d'image et Description des visages 63
4.3.1 Traitements des images 63
4.3.2 Description des visages par Matrice de co-occurrence 64
4.4 Généralité sur les bases de données 67
4.4.1 Base de données ORL 67
4.4.2 Base des données Yale 67
4.5 Apprentissage par les réseaux de neurones bayésiens 67
4.5.1 Codage 68
4.5.2 Structure du réseau neuronal 69
4.5.3 Résultats numériques 69
4.6 Conclusion 70

Conclusion générale **71**

Bibliographie **72**

LIST OF ALGORITHMS

1 Algorithme de rétropropagation du gradient. 26
2 Algorithme de Metropolis-Hastings 41
3 Échantillonneur de Gibbs 42
4 Evidence framework 51

TABLE DES FIGURES

1.1 Structure des neurones biologiques contenus dans le cerveau humain. 13
1.2 Illustration de l'accroissement d'intérêt pour les réseaux de neurones : évolution du nombre de participants au congrès Neuro-Nîmes. 13
1.3 Example d'un neurone biologique. 14
1.4 Mise en correspondance neurone biologique / neurone artificiel. 15
1.5 Modèle d'un neurone artificiel. 15
1.6 Réseau de neurones feedforward à deux couches 16
1.7 Exemple de réseau récurrent simple 17
1.8 Synoptique de l'apprentissage supervisé et non supervisé d'un réseau de neurones. 19
1.9 les fonctions d'activations. 20
1.10 Schéma d'un réseau de neurones non bouclé (Perceptron milticouches). 21
1.11 La Séparation Linéaire Entre La Classe A et B. 22

3.1 L'architecture générale de la procédure d'évidence. 52

4.1 Les principales caractéristiques biométriques pouvant servir comme moyen d'identification : a) forme de l'oreille, b) visage 2D, c) visage 3D, d) visage infrarouge, e) iris, f) rétine, g) empreinte de la main, h) thermogramme de la main, i) forme de la main, j) empreinte digitale, k) voix, l) signature et m) réseau veineux de la main. 58
4.2 Système de reconnaissance de visage. 59
4.3 Exemples de variation de poses (Images recueillies à partir d'Internet). 61
4.4 Exemples de variation d'expressions. 61
4.5 Exemple de variation d'éclairage. 62
4.6 Exemples d'occlusion du visage (Image recueillie à partir d'Internet). 62
4.7 Orientations principales considérées pour le calcul des GLCMs. 64
4.8 Extrait de la base ORL. Pour chacune des 40 personnes enregistrées, on dispose de 10 vues avec des changements de pose, d'expression et d'éclairage. 67
4.9 Quelques exemples extraits de la base de test Yale. 68
4.10 Les étapes et la procédure avant apprentissage. 68
4.11 Structure du réseau neuronal 69

LISTE DES TABLEAUX

1.1 Analogie entre le neurone biologique et le neurone formel. 19

2.1 Nombres aléatoires utilisés pour simuler la distribution bivariée avec un échantillonneur de Gibbs. 43

4.1 Comparaison entre deux algorithmes par leur taux de reconnaissance sous l'influence du pourcentage des images. 70

INTRODUCTION GÉNÉRALE

La reconnaissance de visage est un domaine de la vision par ordinateur consistant à reconnaitre automatiquement une personne à partir d'une image de son visage. C'est un domaine de recherche riche en publications en raison de ses nombreuses applications. Parmi celles-ci les systèmes de contrôle d'accès, de surveillance, d'interaction homme machine dans des applications multimédia de divertissement ou à finalités éducatives, de vérification de carte de crédit et bien d'autres.

Parmi les nombreux modèles proposés pour résoudre le problème de la reconnaissance des visages, on a les réseaux neuronaux, qui occupent dernièrement une place notable dans des problèmes difficiles de classification et de reconnaissance des formes que l'on rencontre précisément en reconnaissance de des visages.

Les méthodes bayésiennes ont été appliquées ces dernières années aux réseaux de neurones par différents auteurs, notamment dans les travaux de MacKay 1992, Neal 1994, Buntine et Weigened 1991, Bishop 1995, et plus récemment Vehtari 1999 et Freitas 2000.
Weigened (1991), Mackay (1992) et Neal (1994) ont montré que les méthodes bayésiennes pour l'apprentissage des réseaux de neurones peuvent apporter plusieurs avantages, car il n'est pas nécessaire de limiter la taille du réseau pour éviter le sur-ajustement, et que le nombre de neurones cachés peut tendre vers l'infini, le seul facteur qui doit limiter la taille du réseau est la capacité des ordinateurs utilisés et le temps disponible pour effectuer les calculs nécessaires, mais comme les paramètres utilisés sont issus d'une distribution de probabilité, il est nécessaire pour connaître un paramètre de calculer des intégrales faisant intervenir les distributions des autres paramètres.

Il est, en général, impossible de calculer ces intégrales analytiquement, et plusieurs approches ont été proposées pour effectuer ces calculs. Mais soit ces méthodes sont très lourdes à implémenter, soit elles reposent sur des approximations qui peuvent fausser les résultats.
Dans ses travaux, Neal 1994 utilise les méthodes Monte Carlo couplées à des modèles de Markov cachés pour calculer les différentes intégrales intervenant dans les différentes étapes. Les calculs sont très lourds à mettre en place et nécessitent beaucoup de temps de calcul.

MacKay (1992) a proposé des approximations reposant sur des hypothèses gaussiennes des probabilités a posteriori. Grâce à ces hypothèses, les calculs d'intégrales se trouvent simplifiés et

peuvent être effectués plus ou moins simplement. Ces approximations sont parfois discutables, surtout pour les problèmes de classification. Néanmoins, grâce à ces approximations, les calculs sont simplifiés de sorte que l'approche bayésienne devienne utilisable en pratique.

Ce mémoire est constitué de quatre chapitres :

Le premier chapitre englobe, dans une première partie, une introduction aux réseaux de neurones artificiels, en définissant les différentes structures fondamentales, les différents types d'apprentissage, ainsi que les propriétés fondamentales des réseaux de neurones. Dans une seconde partie, nous allons voir le modèle le plus célèbre des réseaux de neurones, nommé Perceptron Multicouches. Nous allons définir sa topologie, et réaliser son apprentissage par la méthode de rétropropagation du gradient. Enfin, dans une troisième partie, nous allons introduire le problème de sur-ajustement ainsi que quelques méthodes pour le limiter.

Le deuxième chapitre est consacré à une introduction générale à l'apprentissage statistique, en formulant le problème d'apprentissage statistique, définissant les deux grands types de ce dernier. Après nous allons présentons les différentes techniques pour la simulation des nombres aléatoires distribués suivant une loi donnée, ces nombres sont la base de toute approximation de Monte Carlo. Par suite nous allons décrire le principe général des méthodes de Monte Carlo comme des méthodes numériques pour approximer l'espérance d'une variable aléatoire. A la fin de ce chapitre nous allons présenter un autre type de méthode de simulation des variables aléatoires, c'est exactement les méthodes de Monte Carlo par chaînes de Markov (MCMC). Et on va présenter aussi deux algorithmes de base pour méthodes MCMC, elles s'agit de l'algorithme de Metropolis-Hastings et l'algorithme de Gibbs.

Le troisième chapitre est une combinaison entre les deux premiers chapitres, dans lequel nous allons intégrer les méthodes bayésiennes dans l'apprentissage des réseaux de neurones. Après avoir présenté des généralités sur l'approche bayésienne des réseaux de neurones, à savoir : le principe, les avantages et les inconvénients, nous allons détailler l'approche bayésienne des réseaux de neurones la plus connue à la littérature nommée procédure d'évidence.

L'application des réseaux de neurones à la reconnaissance des visages fait l'objet du quatrième chapitre. Dans une première partie, nous allons présenter la définition de la biométrie et la biométrie du visage comme étant une technique biométrique, ainsi quelques application et quelques principales difficultés de cette technique. Dans une seconde partie, nous allons présenter quelque méthodes de description de visage et d'extraction des caractéristiques de texture, à savoir les Matrices de co-occurrence. Enfin, nous allons adapter les réseaux de neurones à l'apprentissage et la reconnaissance des visages.

CHAPITRE 1

INTRODUCTION AUX RÉSEAUX DE NEURONES ARTIFICIELS

1.1 Introduction

Un réseau de neurones est un modèle de calcul dont la conception est très schématiquement inspirée du fonctionnement de vrais neurones biologiques. Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type statistique.

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 propriété d'approximation universelle. Par la suite, nous verrons la modélisation mathématique des réseaux de neurones artificiels et les différents modèles de ces derniers, en particulier, le modèle de Perceptron Multicouches et son apprentissage par l'algorithme de rétro-propagation du gradient. En fin nous allons définir le problème de sur-ajustement et quelques méthodes pour le limiter.

1.2 Modèles d'un neurone

L'origine des réseaux de neurones vient de l'essai de modélisation mathématique du cerveau humain, les premiers travaux datent de 1943 et qui sont l'oeuvre de W.M. Culloch et W.Pitts. Ils supposent que l'impulsion nerveuse est le résultat d'un calcul simple effectué par chaque neurone et que la pensée est née grâce à l'effet collectif d'un réseau de neurones interconnectés (figure 1.1).

L'utilisation des réseaux de neurones artificiels (RNA) s'est développée dans de nombreuses disciplines (sciences économiques, écologie et environnement, biologie et médecine...). Ils sont notamment appliqués pour résoudre des problèmes de classification, de prédiction, d'optimisation, de reconnaissance des formes et de mémoire associative... Dans le cadre du traitement des données, les RNA constituent une méthode d'approximation de systèmes complexes, particulièrement utile lorsque ces systèmes sont difficiles à modéliser à l'aide des méthodes statistiques classiques.

Figure 1.1 – Structure des neurones biologiques contenus dans le cerveau humain.

En France, elle est à l'image du congrès Neuro-Nîmes qui a pour thème les réseaux neuromimétiques et leurs applications. Créé en 1988, le chiffre de ses participants croit chaque année et reflète bien l'intérêt que le monde scientifique et industriel [1] (figure 1.2).'

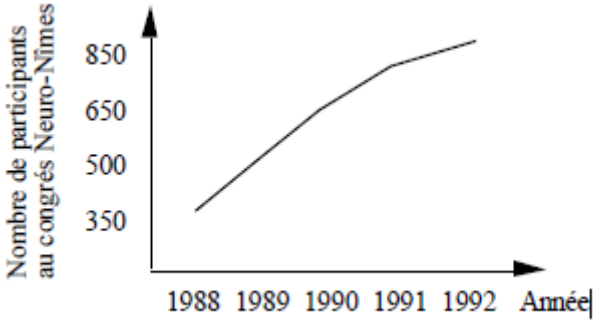


Figure 1.2 – Illustration de l'accroissement d'intérêt pour les réseaux de neurones : évolution du nombre de participants au congrès Neuro-Nîmes.

1.2.1 Neurones biologiques

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le cerveau humain se compose d'environ 100 milliards de neurones, avec 1000 à 10000 synapses (connexions) par neurone. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone.

L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire entre l'axone du neurone afférent et les dendrites (on dit une dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse (figure 1.3).

Le neurone biologique en quelques mots :

★ **Corps cellulaire** :

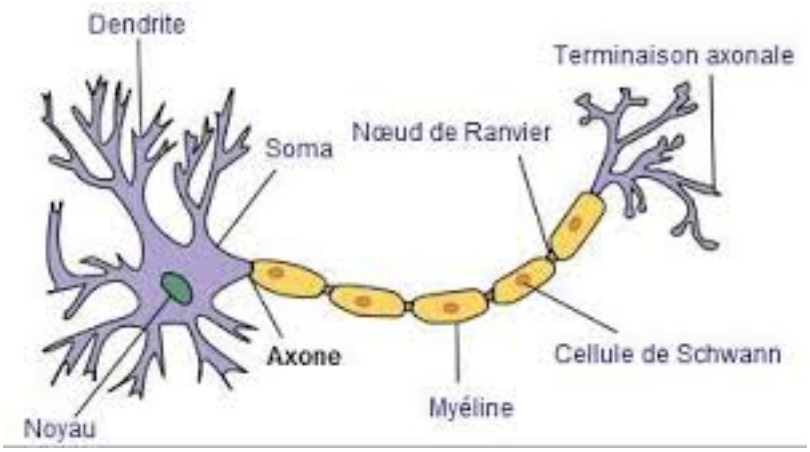


Figure 1.3 – Example d'un neurone biologique.

- ▶ Le noyeu.
- ▶ Traite les courants électriques qui lui proviennent des endrites.
- ▶ Transmet le courant électrique résaultant aux neurones auxquels il est consulté par intermédiaire de son axone.

★ **L'axones** :
- ▶ C'est la fibre nerveuse.
- ▶ Sert de moyen de transport pour les signaux émis par le neurone.

★ **Des dendrites** :
- ▶ Forment une espèce d'arborescence autour de corps cellulaire.
- ▶ Soit les récepteurs principaux du neurone pour capter les signaux qui parviennent.

★ **Des synapses** :
- ▶ Les points de connexion avec les autres neurones, fibres nerveuses ou musculaires

1.2.2 Neurones Formels

Un neurone formel est une fonction algébrique non linéaire et bornée [2], dont la valeur dépend de paramètres appelés coefficients ou poids. Les variables de cette fonction sont habituellement appelées « entrées » du neurone, et la valeur de la fonction est appelée « sortie ».

Un neurone réalise une fonction non linéaire de ses entrées. Ces dernières peuvent être les sorties d'autres neurones : on réalise ainsi un réseau de neurones, qui n'est donc rien d'autre que la composition de fonctions non linéaires élémentaires constituées par les neurones individuels.

Un neurone formel est une représentation mathématique et informatique d'un neurone biologique. Le neurone formel possède généralement plusieurs entrées et une sortie comme il est illustré dans la figure 1.4.
Un neurone est donc avant tout un opérateur mathématique, dont on peut calculer la valeur numérique par quelques lignes de logiciel. On a pris l'habitude de représenter graphiquement [3] un neurone comme indiqué sur la figure 1.5 :

- Les x_i représentent les vecteurs d'entrées, ils proviennent soit des sorties d'autres neurones, soit de stimuli sensoriels (capteur visuel, sonore ...).

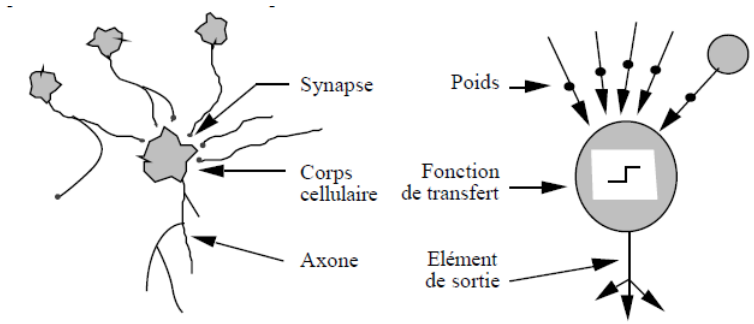


Figure 1.4 – Mise en correspondance neurone biologique / neurone artificiel.

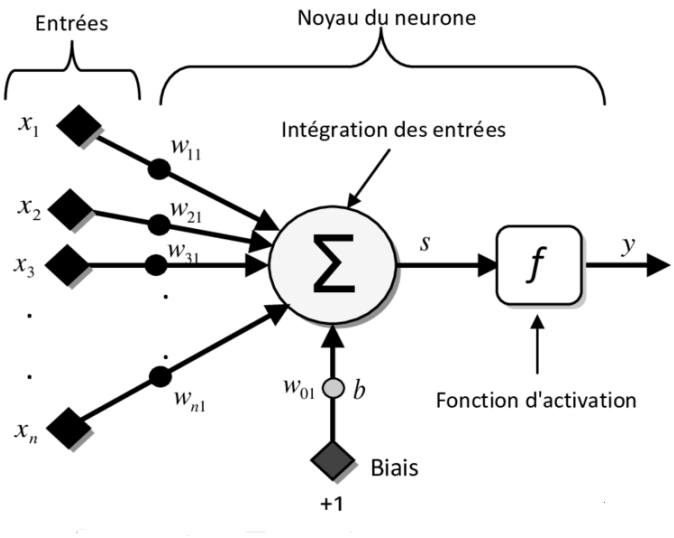


Figure 1.5 – Modèle d'un neurone artificiel.

- Les w_{ij} sont les poids synaptiques du neurone j. Ils correspondent à l'efficacité synaptique dans les neurones. biologiques ($w_{ij} > 0$: synapse excitatrice, $w_{ij} < 0$: synapse inhibitrice). Ces poids pondèrent les entrées et peuvent être modifiés par apprentissage.

- Biais : prend souvent les valeurs -1 ou +1 qui permet d'ajouter de la flexibilité au réseau en permettant de varier le seuil de déclenchement du neurone par l'ajustement des poids et du biais lors de l'apprentissage.

- Noyau : intègre toutes les entrées et le biais et calcule la sortie du neurone selon une fonction d'activation qui est souvent non linéaire pour donner une plus grande flexibilité d'apprentissage.

1.3 Structure des réseaux de neurones

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Les neurones sont organisés en couches successives qui sont reliées entre elles de différentes façons ce qui donne les différents types de réseaux artificiels.

1.3.1 Réseaux de neurones feedforward ou non bouclés

On dit qu'un réseau est feedforward, ou encore à sens unique, lorsqu'il ne contient pas de boucle interne, c'est à dire lorsque l'information qui le traverse ne circule que de l'entrée vers la sortie. Les neurones sont organisés en couches successives, les neurones d'une couche étant reliés à ceux de la suivante. Parmi ce type de réseaux, il existe les réseaux à deux couches. Ils sont composés d'un ensemble de neurones répartis sur deux couches distinctes appelées couche d'entrée et de sortie, les neurones de la première étant connectés à ceux de la seconde. Un exemple de ce type de réseau feedforward est celui de la figure 1.6.

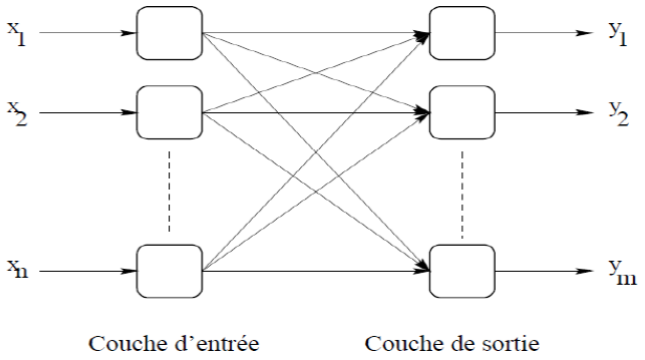


Figure 1.6 – Réseau de neurones feedforward à deux couches

Si l'on introduit entre la sortie et l'entrée une ou plusieurs couches(qui seront alors appelées couches cachées), on obtient un réseau dit **réseau feedforward multi-couches**. L'intérêt de rajouter au moins une couche cachée est d'augmenter le nombre de connexions, ce qui accroît la capacité d'un réseau à extraire l'information des données fournies en entrée.

1.3.2 Réseaux de neurones récurrents

On parle de réseau de neurones récurrent lorsqu'il existe une boucle au moins dans l'ensemble des connexions. Il ya alors au minimum un **feedback** d'une couche sur la précédente. La figure 1.7 présente un exemple très simple : un neurone de la couche de sortie renvoie sa sortie sur la couche d'entrée avec un délai. Cela permet d'utiliser une des sorties à l'instant t comme entrée à l'instant $t + 1$.

1.4 Propriété d'approximation universelle

Les réseaux de neurones à couches cachées ont la propriété générale d'être des approximateurs universels parcimonieux. Il s'agit en fait de deux propriétés distinctes détaillées ci-dessous.

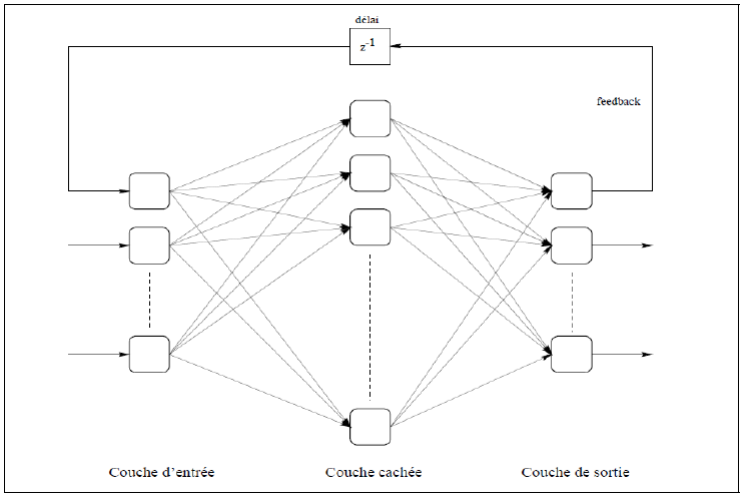


Figure 1.7 – Exemple de réseau récurrent simple

1.4.1 Propriété d'approximation universelle

Cybenko, et Funahashi 1989 démontrent la propriété d'approximation universelle et qui peut s'énoncer de la façon suivante [6] :

Propriété 1.1. *Toute fonction bornée suffisamment régulière peut être approchée uniformément, avec une précision arbitraire, dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche de neurones cachés en nombre fini, possédant tous la même fonction d'activation, et un neurone de sortie linéaire.*

1.4.2 Propriété de parcimonie

Lorsque l'on cherche à modéliser un processus à partir des données, on s'efforce toujours d'obtenir les résultats les plus satisfaisants possibles avec un nombre minimum de paramètres ajustables. Dans cette optique, Hornik et al [7], ont montré que :

Propriété 1.2. *Si le résultat de l'approximation (c'est-à-dire la sortie du réseau de neurones) est une fonction non linéaire des paramètres ajustables, elle est plus parcimonieuse que si elle est une fonction linéaire de ces paramètres.*
De plus, pour des réseaux de neurones à fonction d'activation sigmoide, l'erreur commise dans l'approximation varie comme l'inverse du nombre de neurones cachés, et elle est indépendante du nombre de variables de la fonction à approcher. Par conséquent, pour une précision donnée, donc pour un nombre de neurones cachés donné, le nombre de paramètres du réseau est proportionnel au nombre de variables de la fonction à approcher.

Ce résultat s'applique aux réseaux de neurones à fonction d'activation sigmoïdale puisque la sortie de ces neurones n'est pas linéaire par rapport aux poids synaptiques. Cette propriété montre l'intérêt des réseaux de neurones par rapport à d'autres approximateurs comme les polynômes dont la sortie est une fonction linéaire des paramètres ajustables : pour un même nombre d'entrées, le nombre de paramètres ajustables à déterminer est plus faible pour un réseau de neurones que pour un polynôme. Cette propriété devient d'autant plus intéressante dans le cas du filtrage des textes car le nombre d'entrées est typiquement de l'ordre de plusieurs dizaines.

1.5 Apprentissage des réseaux de neurones

L'apprentissage est une phase du développement d'un réseau de neurones, durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré.
L'apprentissage neuronal fait appel à des exemples de comportement. Il existe essentiellement deux sortes d'apprentissages(selon les exemples sont étiquetés ou non) :

- ★ L'apprentissage supervisé.
- ★ L'apprentissage non supervisé.

1.5.1 Apprentissage supervisé

On parle d'apprentissage supervisé si on dispose de données entrées-sorties :

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

où $x_i \in \mathbb{X}$, avec $\mathbb{X}$ est l'espace des entrées, et $y_i \in \mathbb{Y}$ telle que $\mathbb{Y}$ soit l'espace des sorties.

x_i désigne un stimulus(entrée) et y_i la cible pour ce stimulus, c'est-à-dire la sortie désirée du réseau. Chaque couple (x_i, y_i) correspond donc à un cas d'espèce de ce que le réseau devrait produire(la cible) pour un stimulus donné. Pour cette raison, l'apprentissage supervisé est aussi qualifié d'apprentissage par des exemples.

La distribution qui génèrent les données D est inconnue, ce qui fait la distribution jointe $p(x, y)$ est inconnue. L'apprentissage supervisé consiste alors à construire une fonction paramétrique $f(x)$ à partir des données D :

$$\begin{aligned} f : \mathbb{X} &\longmapsto \quad \mathbb{Y} \\ x &\longmapsto \hat{y} = f(x) \end{aligned}$$

On distingue deux types de problèmes selon la nature de $\mathbb{Y}$. Si $\mathbb{Y}$ est continue on parle de **régression** et on parle de **classification** lorsque $\mathbb{Y}$ est discret[8].

1.5.2 Apprentissage non supervisé

L'apprentissage non supervisé des réseaux de neurones consiste, comme dans le cas d'apprentissage supervisé, à modifier les poids des connections des neurones. Dans ce cas, les exemples de base d'apprentissage sont des données seules : il n'est pas possible de modifier les poids du réseau en fonction d'une erreur sur les réponses souhaitées, puisqu'aucune réponse n'est connue a priori. La figure 1.8 illustre les deux types d'apprentissage : supervisé et non supervisé.

1.6 Modélisation d'un neurone artificiel

La modélisation consiste à mettre en oeuvre un système de réseau de neurones sous un aspect non pas biologique mais artificiel, cela suppose que d'après le principe biologique, on aura une correspondance pour chaque élément composant le neurone biologique, donc une modélisation pour

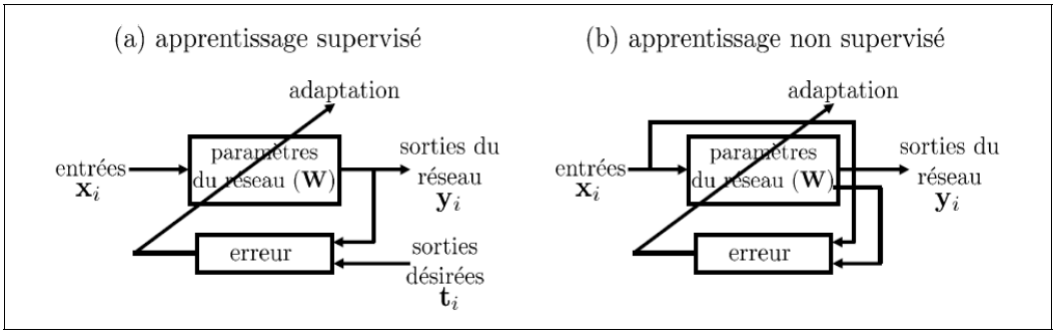


Figure 1.8 – Synoptique de l'apprentissage supervisé et non supervisé d'un réseau de neurones.

Neurone biologique	Neurone formel
Synapses	Poids des connexions
Axones	Signal de sortie
Dendrites	Signal d'entrée
Noyau ou Somma	Fonction d'activation

Table 1.1 – Analogie entre le neurone biologique et le neurone formel.

chacun d'entre eux. On pourra résumer cette modélisation par le tableau 1.1, qui nous permettra de voir clairement la transition entre le neurone biologique et le neurone formel.

Le modèle mathématique d'un neurone artificiel est illustré à la figure 1.5. Un neurone est essentiellement constitué d'un intégrateur qui effectue la somme pondérée de ses entrées. Le résultat s de cette somme est ensuite transformé par une fonction de transfert f qui produit la sortie y du neurone. En suivant les notations présentées à la section précédente, les n entrées du neurone correspondent au vecteur $x = [x_1, x_2, \ldots, x_n]^T$, alors que $w = [w_{1.1}, w_{2.1} \ldots, w_{n.1}]^T$ représente le vecteur des poids du neurone. La sortie s de l'intégrateur est donnée par l'équation suivante :

$$s = \sum_{i=1}^{n} w_{i1} x_i \pm b$$

Que l'on peut aussi écrire sous la forme matricielle :

$$s = w^T x \pm b$$

Pour obtenir la sortie de neurone, il faut ajouter une fonction d'activation :

$$y = y(s) = f\left(w^T x \pm b\right)$$

La fonction d'activation

La fonction d'activation, ou fonction de transfert, est une fonction qui doit renvoyer un réel proche de 1 quand les "bonnes" informations d'entrée sont données et un réel proche de 0 quand elles sont "mauvaises". On utilise généralement des fonctions à valeurs dans l'intervalle réel $[0; 1]$. Quand le réel est proche de 1, on dit que l'unité (le neurone) est active alors que quand le réel est proche de 0, on dit que l'unité est inactive. Elle peut prendre de nombreuses formes : binaire,

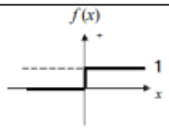
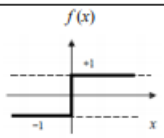
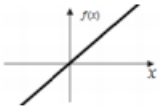
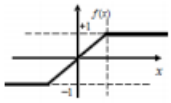
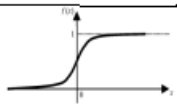
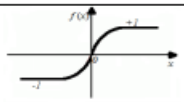
Catégories	Types	Equation	Allure
Seuil	Heaviside	$f(x) = \begin{cases} 0 \; si \; x < 0 \\ 1 \; si \; x \geq 0 \end{cases}$	
	Signe	$f(x) = \begin{cases} -1 \; si \; x \leq 0 \\ 1 \; si \; x > 0 \end{cases}$	
Linéaire	Identité	$f(x) = x$	
	Saturé symétrique	$f(x) = \begin{cases} -1 \; si \; x \leq -1 \\ 1 \; si \; x \geq 1 \\ x \; \text{sinon} \end{cases}$	
Non linéaire	Sigmoïde	$f(x) = \frac{1}{1 + e^{-x}}$	
	Tangente hyperbolique	$f(x) = \frac{2}{1 + e^{-x}} - 1$	

Figure 1.9 – les fonctions d'activations.

linéaire, sigmoïde, et bien d'autres encore. Le tableau ci-dessus dévoile les fonctions d'activations les plus utilisées (figure 1.9) :

Les modèles linéaires, sigmoïdaux sont bien adaptés aux algorithmes d'apprentissage impliquant une rétro-propagation du gradient car leur fonction d'activation est différentiable, ce sont les plus utilisés. Le modèle à seuil est sans doute plus conforme à la réalité biologique mais pose des problèmes d'apprentissage.

1.7 Modèles des réseaux de neurones

1.7.1 Le modèle de Perceptron

Le Perceptron est un réseau de neurones très simple. Il est linéaire et monocouche, acceptant uniquement des valeurs d'entrées et de sorties binaires (0 ou 1). Il est inventé par le psychologue Frank Rosenblatt [3] et inspiré du système visuel. Le procédé d'apprentissage est supervisé et le réseau est capable de résoudre des opérations logiques simples comme les opérations "ET logique" ou "OU logique". Ces réseaux sont efficaces pourvu que les données traitées soient des vecteurs orthogonaux ou linéairement indépendants. Ce que l'on entend par méthode d'apprentissage supervisé est que les sorties du réseau sont comparées avec des sorties cibles. Ce type de réseau est aussi utilisé pour la classification. Les opérations logiques plus compliquées comme un problème "XOR" (OU Exclusif) ne peuvent pas être résolues par un perceptron. Un réseau "feedforward" à

une seule couche (perceptron) peut uniquement représenter des fonctions linéairement séparables. C'est-à-dire celles pour lesquelles la surface de décision séparant les cas positifs des cas négatifs est un hyperplan.

1.7.2 Le modèle de Perceptron Multicouches

Un Réseau de Neurones Artificiels (RNA) est un ensemble de neurones formels associés en couches (ou sous-groupes) et fonctionnant 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). Les neurones sont arrangés par couches. Il n'y a pas de connexion entre neurones d'une même couche, et les connexions ne se font qu'avec les neurones des couches en avales (voir le figure 1.10)

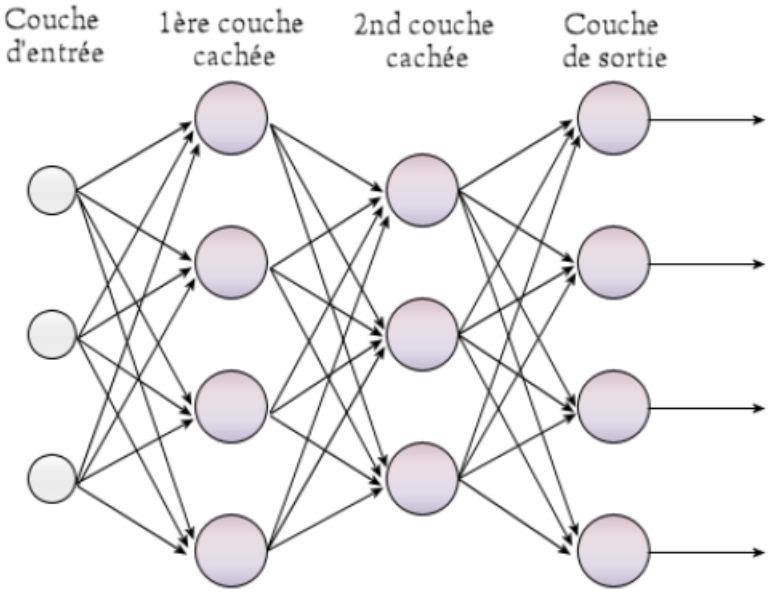


Figure 1.10 – Schéma d'un réseau de neurones non bouclé (Perceptron milticouches).

1.7.3 Le modèle de Adaline

L 'Adaline (Adaptatif Linear Neurone) de Widrow et Hoff est un réseau à trois couches : une d'entrée, une couche cachée et une couche de sortie. Ce modèle est similaire au modèle de Perceptron, seule la fonction de transfert change, mais reste toujours linéaire : $f(x) = x$.

Les modèles des neurones utilisés dans le perceptron et l'Adaline sont des modèles linéaires. Séparation linéaire : on dit que deux classes A et B, sont linéairement séparables si on arrive à les séparer par une droite coupant le plan en deux (voir la figure 1.11).

En 1985 Rumelhart et all [4] ont inventé l'algorithme de rétro-propagation pour l'apprentissage des réseaux multicouches, d'une manière générale nous définissons un réseaux de neurones à propagation avant comme suit :

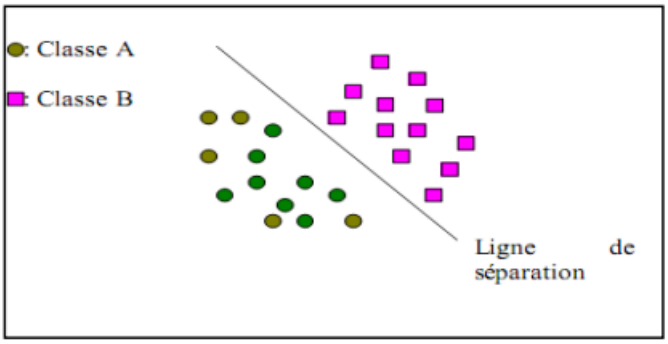


Figure 1.11 – La Séparation Linéaire Entre La Classe A et B.

Définition 1.1. *Soit $\mathbb{X}$ et $\mathbb{Y}$ deux espaces probabilisables et $\mathbb{W}$ un espace des paramètres. Un réseau de neurones artificiel est un ensemble de neurones interconnectés entre eux et organisé en couches. Mathématiquement c'est une application mesurable paramétré par un vecteur des paramètres W permettant d'approximer une relation fonctionnelle :*

$$\begin{aligned} f : \mathbb{X} &\longmapsto \mathbb{Y} \\ x &\longmapsto (f_1(x,w), \cdots, f_m(x,w)) \end{aligned}$$

Où

$$\begin{gathered} (f_k(x,w)) = w_{k0} + \sum_{j=1}^{J} w_{kj} \left(\tanh\left(w_{j0} + \sum_{i=1}^{n} w_{ji} x_i\right)\right) \\ k = 1, \cdots, m \end{gathered}$$

- **n :** le nombre de neurones dans la couche d'entré.
- **J :** le nombre de neurones dans la couche cachée.
- **m :** soit le nombre de neurones dans la couche de sortie.
- **tanh :** La fonction linéaire comme fonction d'activation des neurones de la couche de sortie.

1.8 L'apprentissage du Perceptron Multicouches : L'algorithme de Rétro-propagation du gradient

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 :

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

Avec :

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

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.

Algorithm 1 Algorithme de rétropropagation du gradient.

Entrées: une base de données $D = (X^h, d^h)_{1 \leqslant h \leqslant N}$ et une fonction d'activation f.
1. Initialiser : initialiser tous les poids et les biais du réseau de neurones.
2. Présenter un exemple en entrée : (x^n, d^n) (entrée, sortie désirée)
3. Calculer les sorties : calcules successivement les sorties des différentes couches pour l'entrée X^n.
4. La mise à jour des poids et des biais : modifier récursivement les poids et les biais du réseau suivant les règles détaillées avant :

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

avec :

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

et :

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

5. Critère d'arrêt : répéter les étapes **2**, **3** et **4** jusqu'au nombre maximum d'itérations ou jusqu'à ce que l'erreur quadratique moyenne soit inférieure à un certain seuil donnée.
Sortie : les poids optimaux : w_{ij}.

1.9 Problème de sur-ajustement

1.9.1 L'idée centrale du réseau de neurones

À partir d'exemples d'une relation entre un vecteur d'entrée X et une cible (ou étiquette) d, nous espérons en sortie que le réseau de neurones apprenne un modèle du relation entre X et d. Un réseau formé avec succès donnera, pour tout X donné, une sortie Y qui se rapproche (dans un certain sens) de la cible désirée d. L'apprentissage de réseau consiste à rechercher dans l'espace des pondérations du réseau une valeur de w qui produit une fonction qui s'adapte bien aux données d'entraînement fournies (base d'apprentissage). Dans le cas de régression, la procédure standard d'apprentissage de réseau de neurones consiste à l'ajustement des poids et les biaises du réseau afin de minimiser l'erreur quadratique E_{D_n} associé à une base d'apprentissage $D_n = \{(X_1, d_1); ...; (X_n, d_n)\}$, avec

$$E_{D_n}(w) = \frac{1}{2} \sum_{i=1}^{n} \|d_i - Y_i\|_2^2 \tag{1.10}$$

Avec Y_i est la sortie du réseau pour l'entrée X_i.
Le réseau peut continuer à améliorer ses performances sur les échantillons d'apprentissage même si les paramètres de réseau sont choisis de cette manière, mais il perd son pouvoir de prédiction sur ceux provenant de la validation, ce problème est nommé **sur-apprentissage** [10].

1.9.2 Méthodes pour limiter le sur-ajustement

On distingue deux familles de méthodes pour prévenir le surajustement : les méthodes passives et les méthodes actives. Les philosophies de ces deux familles de méthodes sont différentes.

— Les méthodes passives essayent de détecter le surajustement à posteriori pour supprimer les mauvais modèles. Parmi les méthodes les plus classiques figurent l'utilisation d'une base de validation pendant l'apprentissage, et les mesures de critère d'information.

— Les méthodes actives interviennent pendant la phase d'apprentissage pour empêcher le modèle de faire du surajustement. Les méthodes de régularisation comme l'arrêt prématuré ou la pénalisation entre dans ce cadre.

Les méthodes passives

Les méthodes passives, issues de la description du problème de surajustement en termes de biais-variance ne proposent comme solution qu'une limitation de la complexité du modèle par l'intermédiaire d'une limitation du nombre de neurones cachés.

Les méthodes actives (la méthode de régularisation)

Les méthodes de régularisation, par opposition, peuvent être qualifiés d'actives, car elles ne cherchent pas à limiter la complexité du réseau, mais elles contrôlent la valeur des poids pendant l'apprentissage. Il devient possible d'utiliser des modèles avec un nombre élevé de poids et donc un modèle complexe, même si le nombre d'exemples d'apprentissage est faible.

Une méthode permettant d'éviter le sur-apprentissage est d'utiliser une forme de **régularisation** [11].
Lorsque les poids du réseau sont grands en valeur absolue, les sigmoïdes des neurones cachés sont saturées, si bien que les fonctions modélisées peuvent avoir des variations brusques. Pour obtenir des fonctions régulières, il faut travailler avec la partie linéaire des sigmoïdes, ce qui implique d'avoir des poids dont la valeur absolue est faible.

Durant l'apprentissage, on pénalise les valeurs extrêmes des paramètres, car ces valeurs correspondent souvent au sur-apprentissage. Ainsi la fonction objectif devient :

$$R(w) = \beta E_D(w) + \alpha E_w(w) \tag{1.11}$$

Avec, par exemple $E_w(w) = \frac{1}{2}\sum_{i=1}^{n} w_i^2$ [12] et α et β sont des hyperparamètres de modèle. Ce terme supplémentaire privilégie les valeurs faibles de w et diminue la tendance d'un modèle à sur-ajuster le bruit dans les données d'apprentissage. Le problème principal lié à la régularisation est le choix des valeurs correctes pour les hyperparamètres α et β.

David MacKay [13] a beaucoup travaillé sur l'apprentissage bayésien des réseaux de neurones artificiels. L'approche Bayésienne pour l'apprentissage des réseaux de neurones, sous certaines hypothèses, est équivalent à l'apprentissage déterministe en minimisant une erreur régularisé. Le terme de pénalisation introduit vient du faite que dans cette perspective une information a priori

sur la distribution des poids des connexions est considéré.

En effet le terme de régularisation dépend de la nature de distribution a priori choisi. l'avantage de cette approche est qu'il nous permet de réestimer les paramètres de régularisation afin de réaliser un compromis entre l'erreur de régularisation et l'erreur résultant sur la base d'apprentissage. L'apprentissage via les simulations est une autre version plus robuste de l'approche bayésienne, plusieurs paramètres également doivent être traité.

Pour les RNA Bayésien nous pouvons utiliser un nombre infini des neurones cachés. Nous ne devons pas forcement réduire la taille du PMC en se basant sur la base d'apprentissage,[14] Mais en pratique on doit utiliser un nombre fini des neurones dans la couche cachée afin de limiter les calcules.

1.10 Conclusion

Les réseaux de neurones s'avèrent un outil très puissant que ça soit pour des problèmes de classification, de régression et de prédiction dans divers domaines d'applications grâce à leurs propriétés universelles. Dans ce chapitre nous avons présenté les concepts généraux d'un réseau de neurones, et en particulier le modèle perceptron multicouches ainsi que son algorithme d'apprentissage. Nous avons également énoncé le problème de sur-ajustement ainsi que quelques méthodes pour l'éviter.

CHAPITRE 2

MÉTHODES MCMC POUR LA SIMULATION DES VARIABLES ALÉATOIRES

2.1 Introduction

La statistique bayésienne est une approche statistique fondée sur l'inférence bayésienne. On utilise le terme de bayésienne pour la différencier de la statistique classique qui ne sait traiter que les grands échantillons. La statistique bayésienne est surtout utilisée lorsque l'on n'a que de petits échantillons, typiquement quand chaque observation est elle-même très coûteuse. Contrairement à la statistique classique, elle n'exige pas au départ qu'on se fixe une hypothèse précise à confirmer ou infirmer, ce qui la rend utile en data mining. Dans ce chapitre nous présentons les fondements de la statistique bayésienne ainsi que quelques méthodes de calcul bayésien.

2.2 Vision statistique et Bayésienne de l'apprentissage

2.2.1 Apprentissage statistique

La théorie de l'apprentissage statistique consiste a recherché une fonction prédictive à partir des données observées, elle englobe un ensemble de modèles capables d'extraire l'information pertinente à partir des données ou d'apprendre des comportements à partir d'exemples. Ce qui a permet de construire des outils très puissants pour plusieurs applications dans divers domaines tels que la vision par ordinateur, la reconnaissance de la parole, la bioinformatique, les prédictions, la reconnaissance de formes, etc.

Pour traiter un problème, l'apprentissage automatique passe par l'observation d'un phénomène puis la construction d'un modèle de ce phénomène et en fin les prévisions et l'analyse du phénomène grâce au modèle.

Considérons n observations $x_1, \ldots, x_n$ provenant d'un phénomène aléatoire. Ces observations sont supposées les réalisations des variables aléatoires $X_1, \ldots, X_n$ indépendantes et identiquement

distribuées(i.i.d.), de loi $\mathbb{P}$.

L'apprentissage statistique consiste à estimer la densité de probabilité sous-jacente à $\mathbb{P}$.

On dispose d'un ensemble D contenant un nombre fini n de données. L'objectif d'un algorithme d'apprentissage est d'apprendre une tâche à partir des n exemples représentatifs de cette tâche. On a vu dans le chapitre précédent (subsection 1.5.1), que l'on parle d'apprentissage supervisé si on dispose de données entrées-sorties $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, où $x_i \in \mathbb{X}$ avec $\mathbb{X}$ est l'espace des entrées et $y_i \in \mathbb{Y}$ telle que $\mathbb{Y}$ soit l'espace des sorties. Il consiste aussi à construire une fonction paramétrique $f(x)$ à partir des données D :

$$x \longmapsto \hat{y} = f(x)$$

• **<u>Fonction coût</u>**

$$\begin{aligned} \mathcal{C} : \mathbb{Y} \times \mathbb{Y} &\longmapsto \mathbb{R}^+ \\ (y, y') &\longmapsto \mathcal{C}(y, y') \end{aligned}$$

Différents types des coûts existant à savoir :

— **le coût quadratique** : $\mathcal{C}(y, f(x)) = (y - f(x))^2$
— **le coût absolu** $|y - f(x)|$.
— **le coût** $0 - 1$ pour la classification :

$$\mathcal{C}(y, f(x)) = \begin{cases} 1 & \text{si } y \neq f(x) \\ 0 & \text{si } y = f(x) \end{cases}$$

Il permet de mesurer le nombre d'exemples mal classés.

Théorème 2.1. ***(Risque réel)***
On définit le risque théorique comme étant l'espérance de $\mathcal{C}(Y, f(X))$ par rapport à la distribution jointe $p(X, Y)$:

$$R_{théorique} = E_{X,Y}(\mathcal{C}(Y, f(X))) = \iint_{\mathbb{X}\times\mathbb{Y}} \mathcal{C}(y, f(x))p(x, y)dxdy$$

Remarque 2.1. *Dans le cas des problèmes de classification le risque théorique s'écrit :*

$$R_{théorique} = \int_{\mathbb{X}} \sum_{y\in\mathbb{Y}} \mathcal{C}(y, f(x))p(x, y)dx$$

Dans ce cas l'apprentissage consiste à trouver une fonction minimisant le risque théorique.

Propriété 2.1. *Si les probabilités a postériori $p(y/X = x)$ sont connues alors*

$$\hat{y} = f(x) = E[Y/X = x] = \int_{\mathbb{Y}} yp(y/X = x)dy$$

Preuve
On considère le risque théorique $R_{\text{théorique}}$ On a :

$$\begin{aligned} R_{\text{théorique}} &= \iint_{\mathbb{X}\times\mathbb{Y}} \mathcal{C}(y, f(x))p(x, y)dxdy \\ &= \int_{\mathbb{X}} \left[\int_{\mathbb{Y}} (y - f(x))^2 p(y/X = x)dy\right] P_X(x)dx \end{aligned}$$

Pour tout $X = x$ fixé, Alors minimiser $R_{\text{théorique}}$ revient à minimiser le terme entre crochets

$$\hat{y} = f(x) = \text{argmin} \int_{\mathbb{Y}} (y - f(x))^2 p(y/X = x) dy$$

par dérivation on obtient le résultats de la proposition. ■

En général la distribution jointe $p(x, y)$ est inconnue, on dispose uniquement des données D qui sont les réalisations du couple de variables aléatoires (X, Y). Par conséquent la minimisation du risque théorique est intraitable. La solution proposée dans ce sens est d'utiliser les échantillons D pour construire un estimateur du risque théorique appelé **risque empirique**.
Sous l'hypothèse que la distribution des données est uniforme, le risque empirique s'écrit :

$$R_{emp} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{C}\left(y_i, f\left(x_i\right)\right)$$

• **Notion de Vraisemblance**

Étant donné un ensemble d'observations et une distribution hypothétique f, la vraisemblance mesure le degré de compatibilité de ces observations et la distribution ainsi postulée.
Soient $\mathcal{B}\left(\mathbb{R}^p\right)$ une tribu borélienne, μ une mesure $\sigma-$finie et $\{(\mathbb{R}^p, \mathcal{B}(\mathbb{R}^p)), \mathbb{P}_\theta, \theta \in \Theta\}$ un modèle statistique paramétrique. On suppose l'existence d'une fonction positive $p(., \theta)$ vérifiant

$$\frac{d\mathbb{P}_\theta}{d\mu}(x) = p(x, \theta)$$

appelée dérivée de *Randon-Nykodim* de $\mathbb{P}_\theta$ par rapport à la mesure σ -finie μ.

Définition 2.1. *On définit la fonction de vraisemblance L comme une fonction de densité conjointe des n variables aléatoires $X_1, \ldots, X_n$*

$$\begin{aligned} L : \Theta &\rightarrow \mathbb{R}^+ \\ \theta &\rightarrow p\left(X_1 = x_1, \ldots, X_n = x_n, \theta\right) \end{aligned}$$

Sous l'hypothèse que les variables aléatoires $X_1, \ldots, X_n$ soient indépendantes et identiquement distribuées, la fonction de vraisemblance L associée est donnée par :

$$L(\theta) = p\left(X_1 = x_1, \ldots, X_n = x_n, \theta\right) = \prod_{i=1}^{n} p\left(x_i, \theta\right) \qquad \forall \theta \in \Theta \tag{2.1}$$

Plusieurs estimateurs existent dans le but d' estimer les paramètres de L pour une meilleure approximation de la distribution naturelle f, le plus utilisé est l'estimateur par maximum de vraisemblance.

• **Vraisemblance dans le cas supervisé**

On suppose maintenant que les données sont étiquetés i.e à chaque exemple on dispose de son étiquette. On a alors un couple de variables aléatoires (X, Y). Dans ce cas la vraisemblance est une fonction des paramètres et on a

$$\begin{aligned} L : \Theta &\rightarrow \mathbb{R}^+ \\ \theta &\rightarrow p\left(Y/X_1 = x_1, \ldots, X_n = x_n, \theta\right) \end{aligned}$$

• **Cas des problèmes de régression**

Dans le cas des problèmes de régression, on suppose que $Y = M(X, W) + \epsilon$, où ϵ est une variable aléatoire de densité normal de moyenne 0 et de variance σ^2

$$\begin{aligned} L : \mathcal{W} &\rightarrow \mathbb{R}^+ \\ w &\rightarrow p\left(Y/X_1 = x^1, \ldots, X_n = x^n, w\right) \end{aligned}$$

Généralement $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ en effet : Soit $M(X, w)$ la moyenne empirique d'une suite de variables aléatoires $[Y/X]_i, avec \quad i = 1, \ldots, n$. Alors $Y \equiv \mathbb{E}(Y/X = x)$. Lorsque n tend vers l'infini, d'après le théorème de la limite central, on a :

$$\epsilon_i = \frac{\mathbb{E}(Y/X = x) - M}{\sqrt{n}} \longrightarrow Z$$

Où $Z \sim \mathcal{N}(0, \sigma_i^2)$, ce qui justifie l'hypothèse de normalité du bruit ϵ_i.
Nous allons voir, dans le cas des réseaux de neurones, que la vraisemblance dépend également de la fonction d'activation de couche de sortie.

2.2.2 Statistique Bayésienne

Théorème de bayes

Le théorème de Bayes est un résultat de base en théorie des probabilités, issu des travaux du révérend Thomas Bayes et retrouvé ensuite indépendamment par Laplace. Le théorème de Bayes est utilisé dans l'inférence statistique pour mettre à jour ou actualiser les estimations d'une probabilité ou d'un paramètre quelconque, à partir des observations et des lois de probabilité de ces observations.

Théorème 2.2. *Soit deux événements A et B, avec $P(B) \neq 0$. Si on connaît a priori la probabilité que l'événement B se produise et la probabilité que l'événement B se produise sachant que A s'est produit, alors la probabilité que A se produise sachant que B s'est produit est donnée par :*

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Dans le cadre de l'apprentissage machine, nous sommes intéressés par la probabilité d'avoir un ensemble de paramètres particulier θ, étant donné que nous avons à notre disposition l'ensemble d'apprentissage D. Alors le théorème de Bayes s'écrit :

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

Inférence bayésienne

L'approche Bayésienne repose sur le fait que les paramètres introduits dans le modèle statistique sont supposés aléatoires ce qui rend l'incertitude sur ces paramètres quantifiable. En revanche la modélisation probabiliste n'a de sens, pour l'analyse et les prévisions, que si elle fournit une représentation suffisamment proche du phénomène observé.
On distingue alors deux approches en statistique :

- **La statistique fréquentiste** : qui se base uniquement sur l'information fournie par les données observées.

- **La statistique Bayésienne** : qui met en cause une distribution dite a priori sur les paramètres inconnues. On "ré-actualise" cette distribution a priori au vu de l'observation en calculant la densité a posteriori $\pi(\theta/x)$, et c'est à partir de cette loi que l'on mène l'inférence. L'actualisation de cette distribution par la donné de la vraisemblance se fait à travers le théorème de Bayes pour obtenir une nouvelle forme dite a posteriori, c'est également la loi de θ conditionnellement à x. Elle s'exprime par :

$$\pi(\theta/x) = \frac{L(\theta/x)\pi(\theta)}{\int_\Theta L(\theta/x)\pi(\theta)}.$$

  Cette dernière est utilisée pour l'inférence Bayésienne, elle suffit de déterminer les procédures de décision et conduire toute inférence liée à θ.

Le modèle statistique paramétrique bayésien consiste en la donnée d'une loi a priori et de la loi des observations. On appelle loi des observations, la loi conditionnelle de X sachant θ, sa densité est notée $f(x|\theta)$. La loi a posteriori peut être interpréter comme un résumé (en un sens probabiliste) de l'information disponible sur θ, une fois x observé. L'approche bayésienne réalise en quelque sorte l'actualisation de l'information a priori par l'observation x, à travers de $\pi(\theta/x)$.

Définition 2.2. *On appelle modèle statistique Bayésien, la donnée d'un modèle statistique paramétré* $(\mathfrak{X}, \mathcal{A}, P_\theta, \theta \in \Theta)$ *avec* $f(x/\theta)$ *densité de* P_θ *et d'une loi* $\pi(\theta)$ *sur le paramètre* θ.

Le problème très général auquel on s'intéresse ici est celui d'un individu plongé dans un environnement donné (nature) et qui, sur la base d'observations, est conduit à mener des actions et à prendre des décisions qui auront un coût.

Les espaces intervenant dans l'écriture d'un modèle de décision sont :

— $\mathfrak{X}$: l'espace des observations.
— Θ : l' espace des paramètres dans le cas d'un problème statistique.
— $\mathcal{A}$: l'espace des actions ou décisions.
— $\mathcal{D}$: l'ensemble des règles de décisions δ, applications de $\mathfrak{X}$ dans $\mathcal{A}$ (les estimateurs possibles).

Soit un modèle paramétrique d'observation $x \sim f(x|\theta)$, où $\theta \in \Theta$, un espace de dimension finie, l'analyse statistique bayésienne vise à mieux exploiter l'information apportée par x sur le paramètre θ, pour ensuite construire des procédures d 'inférence sur θ. Bien que x ne soit qu'une réalisation aléatoire d'une loi gouvernée par θ, elle apporte une actualisation aux informations préalablement recueillies par l'expérimentateur.

L'information fournie par l'observation x est contenue dans la densité $f(x|\theta)$, que l'on représente sous la forme inversée de vraisemblance, $l(\theta|x) = f(x|\theta)$, la première expression est une fonction de θ, qui est inconnue, dépendant de la valeur observée x. L'inversion des rôles de x et de θ par rapport à la modélisation probabiliste reflète le premier but de la statistique qui est de reconstruire (avec un certain degré de précision) le paramètre θ au vu de la réalisation aléatoire x

[25].

Dans le paradigme Bayésien le paramètre inconnu θ n'est plus considéré comme inconnu et déterministe, mais comme une variable aléatoire. On considère ainsi que l'incertitude sur θ d'un modèle peut être décrite par une distribution de probabilité π sur Θ appelée distribution a priori, par opposition à la distribution a posteriori qui inclut l'information contenue dans l'observation x, ce qui revient à supposer que θ est distribué suivant $\pi(\theta)$, avant que x ne soit généré suivant $f(x|\theta)$.

D'un point de vue pratique, le choix de la loi a priori est souvent perçu comme une difficulté majeure de l'approche bayésienne en ce que l'interprétation de l'information a priori disponible est rarement assez précise pour conduire à la détermination d'une seule et unique loi.

D'autre part, il est aisé de constater sur des exemples classiques que des choix contrastés de lois a priori conduisent à des inférences divergentes. Il existe néanmoins des lois calibrées en fonction de la distribution des observations, dites lois conjuguées, et des lois à faible contenu informatif, dites lois non informatives, qui permettent d'évaluer l'influence d'une loi a priori donnée, des familles de lois paramétriques. Généralement l'exemple de loi a priori le plus utilisé est **la distribution normal**.

Performance et procédures d'estimation

On considère le modèle suivant $X \sim f(X/\theta)$, où $X \in \mathfrak{X}$ soit des observations et θ paramètre inconnu. Comment estimer une fonction $g(\theta)$? la réponse à cette question se base sur la définition d'une fonction coût $L(\theta, \delta)$ permettant de mesurer l'erreur résultant de l'emploi de δ (règle de décision) pour estimer $g(\theta)$.

On appelle estimation bayésienne du paramètre θ la moyenne de la loi a posteriori. Cette moyenne est notée :

$$\mathbb{E}[\theta|x] = \int_{\Theta} \theta\pi(\theta/x)d\theta$$

L'estimateur de Bayes de θ est noté $\hat{\theta}_B(x) = \mathbb{E}(\theta/x)$, il faut noter que le mode de la loi a posteriori est parfois retenu comme estimation bayésienne de θ. Si la vraisemblance est multimodale on a

$$\hat{\theta}_B(x) = \operatorname{argmax} \pi(\theta/x)$$

L'estimation bayésienne de $g(\theta)$ soit alors $\mathbb{E}[g(\theta)/x]$.
On définit le risque d'un estimateur δ par :

Risque fréquentiste :

$$R(\theta, \delta) = \mathbb{E}_\theta(L[\theta, \delta]) = \int_{\mathfrak{X}} L[\theta, \delta] f(x/\theta)dx$$

Risque de Bayes :

$$r(\pi, \delta) = \mathbb{E}^\pi[R(\theta, \delta)] = \int_{\Theta} \int_{\mathfrak{X}} L[\theta, \delta] f(x/\theta)dx \ \pi(\theta)d\theta$$

★ **Propriétés de l'estimateur de Bayes**

— 1) L'estimateur de Bayes est admissible
Sous quelques conditions, les plus souvent satisfaites, en pratique on a :
— 2) L'estimateur de Bayes converge en probabilité quand la taille de l'échantillon devient plus grande.
— 3) La loi a posteriori peut être asymptotiquement, pour de grandes valeurs de n, approximée par une loi normale $\mathcal{N}(E(\theta/x), \mathrm{Var}(\theta/x))$.

2.3 Méthodes classiques d'approximation

2.3.1 Intégration numérique

À partir de la simple méthode de Simpson, plusieurs approche sont été conçues en Mathématiques appliquées pour l'approximation numérique d'intégrales. Par exemple, la quadrature polynomiale est censée approcher les intégrales liées à des distributions proches de la loi normale, [Naylor et Smith, 1982]. L'approximation de base est donnée par :

$$\int_{-\infty}^{+\infty} e^{\frac{-t}{2}} f\left(t\right) dt \approx \sum_{i=1}^{n} w_i f(t_i)$$

où

$$w_i = \frac{2^{n-1} n\,! \sqrt{n}}{n^2 [H_{n-1}(t_i)]^2}$$

Et t_i est le i-ième zéro du n-ième polynôme d'Hermite, $H_n(t)$.

D'autres approximations d'intégrales reliées à la méthode précédente sont disponibles, qui reposent sur différentes bases orthogonales classiques, [Abramowitz et Stegun, 1964], ou les ondelettes [Muller et Vidakovic, 1999].

2.3.2 Méthodes de Monte Carlo

La solution de nombreux problèmes en mathématiques peut être exprimée en termes d'intégrale d'une fonction. On cherche souvent à obtenir une valeur numérique de cette intégrale, mais cela est souvent difficile de manière analytique [34].

Les intégrales de la forme

$$I = \int_{\Omega} f(x) dx$$

Avec $\Omega \in \mathbb{R}^d$ est le domaine d'intégration, peuvent être liées à une espérance d'une variable aléatoire.

Pour la mesure du probabilité d'une variable aléatoire X ayant une densité $p(x)$ on a :

$$E(f(X)) = \int_{\Omega} f(x) p(x) dx$$

Si de plus $p(x) > 0$ pour tout x tel que $f(x) \neq 0$, alors on a :

$$\begin{aligned} I &= \int_\Omega f(x)dx \\ &= \int_\Omega \frac{f(x)}{p(x)} p(x)dx \\ &= E\left(\frac{f(X)}{p(X)}\right) \\ &= E(g(X)) \qquad \text{avec} \quad g(x) = \frac{f(x)}{p(x)} \end{aligned}$$

Dans le cas où le domaine d'intégration Ω est borné, il suffit de choisir une variable aléatoire X uniformément distribuée sur Ω et de densité $p(x) = \frac{1}{|\Omega|}$, et donc :

$$\begin{aligned} I &= \int_\Omega f(x)dx \\ &= \int_\Omega \frac{f(x)}{\frac{1}{|\Omega|}} \frac{1}{|\Omega|} dx \\ &= |\Omega| E(f(X)) \end{aligned}$$

L'utilité d'exprimer l'intégrale I sous forme d'une espérance découle de **la loi forte des grands nombres** qui nous garantit que pour une suite de variables aléatoires indépendantes identiquement distribuées (i.i.d) $(X_i)_{i\in\mathbb{N}}$ on a :

$$E(g(X)) = \lim_{N\to+\infty} \frac{1}{N} \sum_{i=1}^{N} g\left(X_i\right)$$

Ceci donne une méthode pour estimer la valeur numérique de I, en particulier :

- Générer N variables aléatoires $(X_i)_{i=1,\dots,N}$ de densité $p(x)$ sur Ω.
- Approcher l'espérance en utilisant la loi forte des grands nombres $I \approx \frac{1}{N}\sum_{i=1}^{N} g\left(X_i\right)$.

Ainsi on obtient une approche probabiliste pour estimer la valeur de I.

2.4 Méthode de Monte Carlo par chaînes de Markov

Le rappel de quelques notions sur les chaines de Markov fait l'objet de cette section, permettant d'expliquer les bases nécessaires pour comprendre les algorithmes MCMC. Ensuite, nous définissons l'algorithme de Metropolis-Hastings en se concentrant sur les versions les plus utilisées.

Bien que l'algorithme de Metropolis-Hastings puisse être considéré comme l'un des algorithmes de Monte Carlo par chaines de Markov les plus généraux, il est aussi l'un des plus simples à comprendre et à expliquer, ce qui en fait un algorithme idéal pour débuter.

A la fin nous présentons L'échantillonnage de Gibbs qui est un cas particulier de l'algorithme de Metropolis qui a beaucoup de chance à être utilisé lorsque on connaît les distributions conditionnelles sachant les autres composantes.

2.4.1 Introduction aux chaînes de Markov

Soit $(\Omega, \mathcal{A}, P)$ un espace probabilisé. et $(E, \mathcal{B})$ un espace d'états dénombrable.

Notions sur les chaines de Markov

Définition 2.3. *Soit $(X_n)_{n\geq 0}$ un processus de variables aléatoires à valeurs dans $E = (e_1, e_2, \ldots, e_m)$. On dit que $(X_n)_{n\geq 0}$ est une chaine de Markov si,*
pour tout $e_0, \ldots, e_{n+1} \in E$, on a :

$$P(\underbrace{X_{n+1} = e_{n+1}}_{\textit{Le futur}} \,|\, \underbrace{X_0 = e_0, X_1 = e_1, \ldots, X_n = e_n}_{\textit{Le passé le présent}}) = P(\underbrace{X_{n+1} = e_{n+1}}_{\textit{Le futur}} \,|\, \underbrace{X_n = e_n}_{\textit{Le présent}})$$
$$= P_{n,n+1}$$

$P_{n,n+1}$ est la probabilité de passer de l'état e_n à l'état e_{n+1} en une étape. Autrement dit : une chaîne de Markov sur une space E est un processus aléatoire où l'état actuel de la chaîne étant donnée l'état précédent est indépendant de tous les états passés.

Cette propriété des chaines de Markov est aussi connue comme **propriété de Markov**.

Une chaîne de Markov peut être spécifiée en donnant :

- ★ La distribution marginale de X_0 (la distribution initiale des différents états).
- ★ Les distributions conditionnelles pour les X_{n+1} sachant X_n(les probabilités de transition pour un état de suivre un autre état).

On note par $p_0(x)$ la probabilité initial pour l'état x, c'est à dire $P(X_0 = x)$, et par $M_n(x, x')$ la probabilité de transition pour que l'état x' à l'instant $n+1$ suivre l'état x de l'instant n, c'est à dire $P(X_{n+1} = x'|X_n = x)$. Les probabilités de transition seront notées $M(x, x')$. Si les probabilités de transition ne dépendent pas de n, alors dans ce cas on dit que la chaîne de Markov est **homogène**.

En utilisant les probabilités de transition, on peut trouver facilement la probabilité d'avoir l'état x à l'instant $n+1$, noté par $p_{n+1}(x)$, en effet :

$$p_{n+1}(x) = \sum_{x'} p_n(x') M_n(x', x) \tag{2.2}$$

Définition 2.4. *On dit qu'une chaîne de Markov $(X_n)_{n\geq 0}$ est **homogène**, si pour tout $n \geq 0$, x et y dans E on a :*

$$P(X_{n+1} = y|X_n = x) = P(X_1 = y|X_0 = x)$$

Dans ce cas, on pose

$$M(x, y) = P(X_1 = y|X_0 = x), \qquad x, y \in E$$

*M est dite **la matrice de transition** de la chaîne $(X_n)_{n\geq 0}$, on dit aussi **noyau de transition** quand E est infini.*

Définition 2.5. *Étant donné une chaîne de Markov sur E de matrice de transition M, on dit que deux états e_i et e_j se communiquent si la chaîne partant de e_i atteint un jour e_j avec une probabilité strictement positive, et réciproquement, si partant de e_j , elle atteint un jour e_i avec une probabilité strictement positive.*

Remarque 2.2. — *La distribution de la chaîne à l'instant n (i.e la distribution de X^n) peut être vue comme un vecteur ligne, p_n.*
— *Les probabilités de transition à l'instant n comme une matrice, M_n, ou M dans le cas des chaînes de Markov homogènes.*
— *(Ce type des matrices dont tous les éléments sont positifs et la somme des lignes égale à 1, sont appelés **Les matrices stochastiques**).*

On peut écrire alors l'équation 2.2 comme le suivant :

$$p_{n+1} = p_n M_n$$

Et pour les chaînes homogènes on écrit

$$p_{n+1} = p_0 M$$

Dans ce qui suit on suppose que $(E, \mathcal{B})$ est un espace d'états infini non dénombrable.

Définition 2.6. *Un processus aléatoire est une chaîne de Markov à valeurs dans $(E, \mathcal{B})$ si et seulement si :*

$$\forall A \in \mathcal{B} \quad P(X_{n+1} \in A | X_0, X_1, \ldots, X_n) = P(X_{n+1} \in A | X_n)$$

Définition 2.7. *On appelle **noyau de transition** tout application $T_n : E \times \mathcal{B} \longrightarrow \mathbb{R}^+$, telle que*

- *$\forall x \in E$, l'application $A \longrightarrow T_n(x, A)$ est une probabilité,*
- *$\forall A \in \mathcal{B}$, l'application $x \longrightarrow T_n(x, A)$ est $\mathcal{B}$-mesurable.*

Dans le cas continu, le noyau de transition est aussi une densité conditionnelle $T_n(x, x')$ et on a :

$$P(X^{n+1} \in A | X^n = y) = \int_A T_n(y, x) dx, \ \forall A \in \mathcal{B} \tag{2.3}$$

Si E est discret le noyau de transition est simplement une matrice de transition.

Définition 2.8. *Si le noyau de transition $T_n(x_n, x_{n+1})$ est indépendant de l'indice courant n, la chaîne est dite homogène (dans ce cas on a $\forall n \in \mathbb{N}$: $T_n = T$), et sera dite stationnaire si $\forall h \in \mathbb{N}$, $\left(x_{t_1}, \ldots, x_{t_p}\right)$ et $\left(x_{t_1+h}, \ldots, x_{t_p+h}\right)$ ont la même loi.*

Définition 2.9. ***(Mesure invariante)***
*Soit T un noyau de transition sur $(E, \mathcal{B})$, et π une mesure sur $(E, \mathcal{B})$. π est dite **invariante** pour T ssi $\pi T = \pi$ i.e :*

$$\forall y \in E \ \sum_{x \in E} \pi(x) T(x, y) = \pi(y)$$

Ou

$$\forall A \in \mathcal{B} \ \int_E \pi(dx) T(x, A) = \pi(A)$$

Définition 2.10. ***(loi stationnaire)***
On dit que π est une loi stationnaire d'une chaîne de Markov de noyau de transition K si $\pi(x)$ est une densité de probabilité et si

$$\pi(B) = \int \pi(x) T(x, B) dx$$

Par analogie avec $\pi = \pi T$ dans le cas des chaînes à valeurs dans un espace d' état discret et T la matrice de transition, on parle aussi de loi invariante car si $x_n \sim \pi$, et que π est une loi stationnaire de la chaine de Markov alors $x_{n+1} \sim \pi$.

Condition d'équilibre

Une chaîne de Markov de noyau de transition T vérifie la condition d'équilibre s'il existe une fonction π telle que :

$$T(y,x)\pi(y) = T(x,y)\pi(x) \qquad \forall (x,y) \in E^2$$

C'est une condition suffisante pour que π soit une loi stationnaire de la chaine de Markov.

Bien que cette condition ne soit pas nécessaire pour que π soit une loi stationnaire associée au noyau de transition T, elle fournit une condition suffisante qui est souvent facile à vérifier et qui peut être utilisée pour la plupart des algorithmes MCMC. La condition d'équilibre exprime un équilibre dans le flux de la chaine de Markov, à savoir que la probabilité d'être dans x et de passer par y est la même que la probabilité d'être dans y et de revenir à x.

Chaine de Markov irréductible

La propriété d'**irréductibilité** est une première mesure de la sensibilité de la chaîne de Markov aux conditions initiales, x_0 ou μ(distribution invariante). Elle est crucial dans la mise en place des algorithmes de Monte Carlo par chaîne de Markov, car elle conduit à une garantie de convergence, évitant ainsi une étude détaillée de l'opérateur de transition, qui serait autrement nécessaire pour spécifier des conditions initiales acceptables.

Définition 2.11. *Une chaîne de Markov est π-**irréductible**, avec π mesure de probabilité si :*

$$\forall x \in E, \forall A \in \mathcal{B} \text{ avec } \pi(A) > 0, \exists n \in \mathbb{N} \text{ telle que } P^n(x,A) > 0.$$

Où P^n est le noyau de transition défini par $P^n(x,A) = P(X_n \in A / X_0 = x)$.

Dans le cas discret, la chaîne est irréductible si tous les états se communiquent, voir 2.5.

Ergodicité et Convergence

Il est naturel d'établir le comportement limite de la chaine X_n, c'est-à-dire, à quoi la converge-t-elle?. L'existence (et l'unicité) d'une distribution invariante μ fait de cette distribution un candidat naturel pour la distribution limite (lorsque la distribution de X_n converge vers la distribution invariant indépendamment du choix de distribution initiale, on dit que la chaîne est **Ergodique**), et nous allons maintenant trouver des conditions suffisantes sur X_n pour que la chaîne soit asymptotiquement distribuée selon μ. Le théorème suivant est l'un des résultats de convergence fondamentale pour les chaînes de Markov, et il est au coeur de la motivation pour les algorithmes de Monte Carlo par chaîne de Markov.

Théorème 2.3. *Si la chaîne est irréductible, il existe une unique loi stationnaire π qui est presque sûrement la loi limite de la chaîne de Markov.*

On considère une chaîne de Markov de distribution limite π. Pour toute fonction intégrable h

$$\lim_{n\to+\infty} \frac{1}{n}\sum_{i=1}^{n} h\left(X_i\right) = \int_E h(x)\pi(x)dx$$

2.4.2 Principe des méthodes MCMC

Les méthodes de Monte Carlo par Chaîne de Markov (MCMC) sont des stratégies pour générer des échantillons $x(i)$ tout en explorant l'espace d'état E en utilisant un mécanisme de chaîne de Markov. Ce mécanisme est construit de sorte que la chaine passe plus de temps dans les régions les plus importantes. En particulier, il est construit de sorte que les échantillons $x(i)$ imitent des échantillons tirés de la distribution cible $\pi(x)$. Nous rappelons que nous utilisons les MCMC quand nous ne pouvons pas tirer directement des échantillons de $\pi(x)$, mais nous pouvons évaluer $\pi(x)$ à une constante de normalisation près.

Définition 2.12. *Une méthode de Monte Carlo par Chaine de Markov (MCMC) pour la simulation d'une distribution π est une méthode produisant une chaine de Markov ergodique $(X(t))$ dont la distribution stationnaire est π.*

Les techniques MCMC sont souvent appliquées pour résoudre les problèmes d'intégration et d'optimisation sur des espaces en grande dimension. Ces deux types de problèmes jouent un rôle fondamental dans l'apprentissage automatique, la physique, la statistique, l'économétrie, l'analyse décisionnelle, l'inférence bayésienne et apprentissage, Mécanique statistique, sélection du modèle et vraisemblance pénalisé.

On souhaite échantillonner selon une loi cible de densité $\pi(x)$ connue à une constante multiplicative près. Soit une mesure de probabilité π sur un espace mesurable $(E, \mathcal{B})$. f est une fonction réelle mesurable et π-intégrable.

L'objectif général des méthodes MCMC, dans le cadre de l'intégration, est de proposer une approximation de la quantité suivante :

$$\pi(f) = \int_E f(x)dx$$

Si on arrive à construire une chaine de Markov (X_k) ergodique dont la mesure invariante est π Alors :

$$\frac{1}{N}\sum_{k=1}^{N} f\left(X_k\right) \longrightarrow \pi(f)$$

Il existe plusieurs algorithmes permettant de construire le noyau de transition précédent, **l'algorithme de Métropolis** [16] et **Hastings** [17] **l'échantillonnage de Gibs** [18] sont considérés comme des algorithmes de base pour les méthodes MCMC.

Algorithme de Metropolis-Hasting

L'algorithme de Metropolis est parmi les dix algorithmes qui ont influencé sur le développement et la pratique de la science et de l'ingénierie au 20eme siècle [19]. Cet algorithme est une instance

d'une grande classe d'algorithmes d'échantillonnage, connue sous le nom de Markov Chaîn Monte Carlo (MCMC). Ces algorithme sont joué un rôle important dans les domaines de la statistique, de l'économétrie, de la physique et de l'informatique au cours des deux dernières décennies. Il existe plusieurs problèmes en grande dimension, tels que le calcul du volume d'un corps convexe en dimensions d, pour lequel la simulation MCMC est la seule approche générale connue pour fournir une solution dans un délai raisonnable [20],[21].
On définit une loi de proposition (appelé aussi loi instrumentale) $q(y \mid \theta)$ et on initialise θ^0. À partir de θ^t, on génère y^t à l'aide de la loi de proposition et on accepte ou on rejette cette valeur de y^t à l'aide d'une procédure d'acceptation-rejet. La valeur retenue est notée θ^{t+1}.

Algorithm 2 Algorithme de Metropolis-Hastings

1-**Itération 0 :** Initialiser avec une valeur arbitraire θ^0.
2-**Itération n :** Mettre à jour θ^n par θ^{n+1} de façon suivante :

1. **Générer** $\psi \sim q(\psi \mid \theta^n)$
2. **Poser** :
$$\mathcal{E}(\theta^n, \psi) = min\left(\frac{\pi(\psi)q(\theta^n \mid \psi)}{\pi(\theta^n)q(\psi \mid \theta^n)}, 1\right)$$
3. **on définit** θ^{n+1} comme suit :
$$\theta^{n+1} = \begin{cases} \psi & \text{avec probabilité } \mathcal{E}(\theta^n, \psi). \\ \theta^n & \text{sinon.} \end{cases}$$

Remarque 2.3. — *Le choix de la distribution des propositions est important car il détermine la vitesse de convergence vers π et l'efficacité de l'échantillonnage.*
— *La loi de proposition doit être indépendante de l'état actuel.*
— *Il faut noter qu'il existent plusieurs choix de la loi de proposition :*

- *La loi de proposition indépendante $q(\theta^*/\theta^n) = q(\theta^*)$.*
- *La loi de proposition symétrique $q(\theta^*/\theta^n) = q(\theta^n/\theta^*)$.*
- *L'algorithme de Métropolis à marche aléatoire $q(\theta^*/\theta^n) \sim \mathcal{N}(\theta^n, \sigma^2 I)$.*

Echantillonage de Gibbs

L'algorithme de Gibbs [31] est une version de Métropolis-Hasting, il est utile lorsque la dimension de l'espace des paramètres augmente(parfois jusqu'à des centaines et des milliers de composantes), et il devient difficile de trouver des distributions de proposition appropriées pour l'algorithme de Metropolis.

L'échantillonneur de Gibbs [35] est une manière de générer des distributions de deux (ou plusieurs) variables à partir d'un modèle qui définit les distributions de probabilité conditionnelles.

Dans le cas d'un modèle à deux variables, la méthode consiste à prendre un élément de départ (x_0, y_0) et à générer à l'aide de nombres aléatoires les éléments d'un échantillon fictif (x_n, y_n) par

itération en choisissant dans l'ordre x_n d'une variable aléatoire de densité $f(X|Y = y_{n-1})$, et y_n d'une variable aléatoire de densité $g(Y|X = x_n)$, où $f(X|Y = y_{n-1})$ et $g(Y|X = x_n)$ sont les distributions de probabilité conditionnelles supposées connues ou modélisées.

Pour le cas d'un modèle à plusieurs variables où les distributions conditionnelles seraient connues, l'élément au départ serait $(x_1^0, x_2^0, \ldots, x_N^0)$ et l'échantillon est construit en simulant $(x_1^n, x_2^n, \ldots, x_N^n)$ en choisissant dans l'ordre

x_1^n d'une variable aléatoire de densité $f_1(X_1|X_2 = x_2^{n-1}, \ldots, X_N = x_N^{n-1})$

x_2^n d'une variable aléatoire de densité $f_2(X_2|X_1 = x_1^n, X_3 = x_3^{n-1}, \ldots, X_N = x_N^{n-1})$

x_3^n d'une variable aléatoire de densité $f_3(X_3|X_1 = x_1^n,\ X_2 = x_2^n,\ X_4 = x_4^{n-1}, \ldots)$

et ainsi de suite jusqu'à

x_N^n d'une variable aléatoire de densité $f_N(X_N|X_1 = x_1^n, \ldots, X_{N-1} = x_{N-1}^n)$.

L'échantillon ainsi généré sera une simulation conforme aux probabilités conditionnelles imposées et la distribution se conformera d'autant plus aux probabilités conditionnelles que le nombre d'itérations sera élevé.

Un point est donc généré à partir du point précédent, sans utiliser l'information sur les autres points générés : il s'agit donc d'une chaine de Markov. L'échantillonneur de Gibbs est un cas particulier de l'algorithme de Metropolis-Hastings (1953), qui génère une suite de réalisations et applique le principe de l'acceptation-rejet sous une forme plus générale.

L'algorithme d'échantillonnage de Gibbs (ou échantillonneur de Gibbs) est donné par les étapes suivants :

Algorithm 3 Échantillonneur de Gibbs

1. Initialiser $x^0 = (x_1^0, x_2^0, \cdots, x_N^0)$
2. à l'itération $t \quad (t = 1, 2, \cdots)$, étant donné $x^t = (x_1^t, x_2^t, \cdots, x_N^t)$,
générer

$$X_1^{t+1} \sim f_1(x_1|x_2^t, \cdots, x_N^t);$$
$$X_2^{t+1} \sim f_2(x_2|x_1^{t+1}, \cdots, x_N^t);$$
$$\vdots$$
$$X_N^{t+1} \sim f_N(x_N|x_1^{t+1}, \cdots, x_{N-1}^{t+1}).$$

3. $t \longleftarrow t + 1$. et aller à **2**

Exemple : Une distribution de points sur la portion de plan $Q = \{(x, y),\ \ x > 0,\ y > 0\}$, est telle que la distribution des x, pour y fixé suit une loi exponentielle de paramètre y et que la

distribution des y pour x fixé suit une loi exponentielle de paramètre x.

Un échantillonneur de Gibbs peut être utilisé pour simuler un échantillon de 10 points de Q conforme aux distributions conditionnelles données.

Les distributions de répartition conditionnelles peuvent s'écrire comme suit :

$$F(x|y) = \int_0^x ye^{-ty}dt; \quad G(y|x) = \int_0^y xe^{-tx}dt$$

Donc à l'aide d'un échantillonneur de Gibbs si u_n et v_n sont des nombres aléatoires issus d'une loi uniforme $\mathcal{U}_{(0,1)}$, l'échantillon peut être calculé à l'aide des relations

$$x_{n+1} = -\frac{1}{y_n}\log u_n; \quad y_{n+1} = -\frac{1}{x_{n+1}}\log v_n$$

Par exemple en utilisant les nombres aléatoires ci-dissous, à partir du point (1,1), on obtient les 10 points (0,145 ; 4,087), (1,309 ; 0,686), (0,036 ; 0,657), (3,788 ; 0,095), (14,308 ; 0,108), (16,938 ; 0,115), (4,499 ; 0,182), (0,575 ; 0,123), (0,487 ; 1,095), (0,931 ; 0,029).

u_n	v_n
0,864289	0,550899
0,082672	0,695312
0,004736	0,406955
0,253519	0,212330
0,159700	0,140474
0,593694	0,440392
0,900385	0,931119
0,941360	0,586161
0,360221	0,972795
0,975364	0,976381

Table 2.1 – Nombres aléatoires utilisés pour simuler la distribution bivariée avec un échantillonneur de Gibbs.

2.5 Conclusion

L'utilisation du théorème de Bayes permet de proposer une formulation bayésienne de l'apprentissage, qui est très générale, et qui peut s'appliquer à la recherche de tous types de modèles pour peu que ceux-ci soient représentés par un ensemble de paramètres sur lesquels on peut faire une hypothèse de distribution. Nous avons énoncé dans ce chapitre les notions de base du formalisme bayésien et nous avons détaillé quelques méthodes du calcul bayésien.

CHAPITRE 3

INTÉGRATION DES MÉTHODES MCMC DANS L'APPRENTISSAGE DES RNAS

3.1 Introduction

Les méthodes bayésiennes ont été appliquées ces dernières années aux réseaux de neurones par différents auteurs, notamment dans les travaux de MacKay 1992, Neal 1994,...

Les méthodes Monte Carlo par Chaînes de Markov (MCMC), s'inscrivent dans le cadre du formalisme Bayésien. On peut dire que l'application de ces méthodes à l'apprentissage des réseaux de neurones peut apporter plusieurs avantages, tout d'abord, l'introduction de connaissances a priori est susceptible d'améliorer l'estimation des paramètres. De plus, le formalisme Bayésien permet une analyse complète des incertitudes et des probabilités des données.

3.2 Introduction à l'approche bayésienne

3.2.1 Principe de l'approche bayésienne

L'apprentissage classique était effectué en trouvant une valeur du vecteur des poids qui minimise une fonction de coût ou d'erreur.

L'approche bayésienne pour les RNAs consiste à considérer tous les paramètres, notamment les poids du réseau notés $w = (w_1; \cdots; w_l)$, (l qui désigne la dimension de w) comme des variables aléatoires issues d'une distribution de probabilité.

L'apprentissage d'un réseau de neurones consiste donc à déterminer la distribution de probabilité des poids connaissant les données d'apprentissage. Dès lors, il y a un sens d'attribuer aux poids une distribution de probabilité a priori qui décrit le savoir actuel sur ce paramètre, cette distribution de probabilité est notée $P(w)$. Il faut bien comprendre que la distribution a priori

quantifie l'état de connaissance d'un expert (et donc son incertitude) sur le problème en main. Cela signifie que l'expert parie plus volontiers sur certaines valeurs de w que sur d'autres, cette information a d'autant plus de valeur que les données sont rares. Il doit être clair que le savoir de l'expert encodé dans la distribution a priori doit être tout à fait indépendant de l'échantillon en main, sinon la même source d'information interviendrait deux fois, ce qui ne serait pas cohérent.

Pour démarrer la démarche d'apprentissage bayésien, il faut d'abord définir la distribution a priori $P(w)$ des pondérations, qui exprime nos croyances initiales sur leurs valeurs avant que toute donnée soit arrivée. Quand on observe $D = \{(x_1, d_1), (x_2, d_2), \cdots, (x_n, d_n)\}$, nous mettons à jour cette distribution a priori à une distribution a posteriori en utilisant La règle de Bayes :

$$P(w|D) = \frac{P(D|w)\,P(w)}{P(D)} \propto P(D|w)\,P(w) \tag{3.1}$$

3.2.2 Avantages de l'approche bayésienne

D'un point de vue théorique, Neal[15] a montré que, lorsque les probabilités a priori des poids sont convenablement choisies, il n'est pas nécessaire de limiter la taille du réseau pour éviter le surajustement, et le nombre de neurones cachés peut tendre vers l'infini. Selon cette étude, le seul facteur qui doit limiter la taille du réseau est la capacité des ordinateurs utilisés et le temps disponible pour effectuer les calculs nécessaires.

D'un point de vue pratique, la théorie de l'approche bayesienne pour l'apprentissage des réseaux de neurones apporte d'importantes améliorations :

- ★ Le concept de régularisation peut être interprété de façon naturelle dans le contexte bayésien.
- ★ Les hyperparamètres intervenant dans la fonction de régularisation sont calculés lors de la phase d'apprentissage sans utiliser de base de validation.
- ★ Le calcul de l'évidence permet de sélectionner, parmi une famille de modèles, le meilleur modèle, uniquement grâce à la base d'apprentissage.
- ★ Comme tous les calculs se font à partir de la base d'apprentissage, il n'est plus nécessaire de disposer d'une base de validation. Il est donc possible d'utiliser toutes les données dont on dispose pour estimer les poids du réseau.
- ★ Des barres d'erreurs peuvent être calculées pour les problèmes de régression.
- ★ L'incertitude sur les poids peut être prise en considération pour corriger la probabilité calculée par un réseau dans un problème de classification.
- ★ Les entrées peuvent être sélectionnées grâce à la méthode **Automatic Relevance Determination** : un hyperparamètre est associé à chaque entrée et après l'apprentissage, les

hyperparamètres avec de grandes valeurs indiquent des entrées non pertinentes.

3.2.3 Inconvénients de l'approche bayésienne pour les RNA

Comme les paramètres utilisés sont maintenant issus de distributions de probabilité, il est nécessaire, pour connaître un paramètre, de calculer des intégrales faisant intervenir les distributions des autres paramètres. Il est, en général, impossible de calculer ces intégrales analytiquement, et plusieurs approches ont été proposées pour effectuer ces calculs. Mais soit ces méthodes sont très lourdes à implémenter, soit elles reposent sur des approximations qui peuvent fausser les résultats. Finalement les résultats théoriques proposés par l'approche bayésienne sont souvent inapplicables en l'état dans le cadre des réseaux de neurones.

Dans ses travaux, Neal[15] utilise des méthodes de Monte Carlo couplées à des modèles de Markov cachés pour calculer les différentes intégrales intervenant dans les différentes étapes.
Les calculs sont très lourds à mettre en place et nécessitent beaucoup de temps de calcul. Nous n'avons pas cherché dans cet ravail à utiliser cette approche.

MacKay [22], [23] et[24], a proposé des approximations reposant sur des hypothèses gaussiennes des probabilités a posteriori. Grâce à ces hypothèses, les calculs d'intégrales se trouvent simplifiés et peuvent être effectués plus ou moins simplement. Ces approximations sont parfois discutables, sur tout pour les problèmes de classification. Néanmoins, grâce à ces approximations, les calculs sont simplifiés de sorte que l'approche bayésienne devient utilisable en pratique. L'approche proposée par MacKay est connue sous le nom de **evidence framework.**

3.3 Procédure d'évidence

On propose d'utiliser des hypothèses d'approximations gaussiennes, on supposera pour ce faire que le bruit sur les données utilisées, est gaussien et que donc, la distribution des erreurs le sera aussi. Les FDP, apriori, des paramètres auront également une distribution gaussienne.

3.3.1 Distribution a priori des poids

On suppose que la distribution a priori des poids de réseau est Gaussienne de moyenne 0 et de variance σ_{priori}, c'est à dire que

$$P(w|\alpha) = \frac{1}{Z_w(\alpha)} \exp\left(-\alpha E_w\right) \tag{3.2}$$

Avec

- $\alpha = \frac{1}{\sigma^2_{\text{priori}}}$
- $Z_w(\alpha)$ représente la constante de normalisation de la FDP. Dans le cadre Bayésien, α est un hyperparamètre qui contrôle la distribution des autres paramètres. Le choix d'une probabilité a priori gaussienne pour les poids permet une interprétation probabilistede E_w ainsi que son interprétation comme étant le minimum du log de la distribution de la probabilité

a priori sur les paramètres (poids et biais du réseau).

On suppose que α est connu, et comme on a choisi une distribution gaussienne, on en déduit la constante de normalisation $Z_w(\alpha)$:

$$Z_w(\alpha) = \left(\frac{2\pi}{\alpha}\right)^{\frac{l}{2}}$$

Avec l représente le nombre total de paramètres du réseau de neurones.

3.3.2 Fonction de vraisemblance

$P(D/w)$ est la vraisemblance associé à la base de données D caractérisé par l'inverse de sa variance noté β appelé hyperparamètre. Si on suppose que le bruit dans le modèle de régression est Gaussien de moyenne 0 et de variance σ_{bruit} . Alors on obtient :

$$P\left(D|w,\beta\right) = \frac{1}{Z_D(\beta)} \exp\left(-\beta E_D\right) \tag{3.3}$$

En effet :

L'objectif de réseau de neurones est de trouver la fonction f entre les stimulus x_i et les sorties disirées d_i(cibles). Les données utilisées ayant de forte chances d'être bruitées, la relation se traduit par :

$$d_i = f(x_i) + \epsilon_i$$

Considérons que les erreurs ϵ_i ont une distribution gaussienne de moyenne 0 et de variance σ_{bruit}. La distribution du bruits s'écrit alors comme suit :

$$P(\epsilon_i/\beta) = \sqrt{\frac{\beta}{2\pi}} exp\left(-\frac{\beta}{2}\epsilon_i\right)$$

avec $\epsilon_i = d_i - f(x_i)$

En considérant que l'hyperparamètre β est connu. Alors la distribution gaussienne du bruit nous permet de calculer le facteur de normalisation $Z_D(\beta)$ qui est donné par :

$$Z_D(\beta) = \left(\frac{2\pi}{\beta}\right)^{\frac{n}{2}}$$

avec n est le nombre des éléments de la base de données.

3.3.3 Distribution a posteriori des paramètres

La distribution a posteriori combine la fonction de vraisemblance, $P(D|w,\beta)$ avec la distribution a priori $P(w|\alpha)$ qui sont définies précédemment comme suit :

$$P(w|\alpha) = \frac{1}{Z_w(\alpha)} \exp\left(-\alpha E_w\right)$$

$$P\left(D|w,\beta\right) = \frac{1}{Z_D(\beta)} \exp\left(-\beta E_D\right)$$

En Remplaçant ces distributions dans l'équation 3.1, on obtient :

$$P\left(w|D,\alpha,\beta\right) = \frac{\frac{1}{Z_D(\beta)Z_w(\alpha)}\exp\left(-\left(\beta E_D + \alpha E_w\right)\right)}{P\left(D|\alpha,\beta\right)}$$
$$= \frac{1}{Z_R(\alpha,\beta)}\exp(-R(w))$$

Avec

$$R(w) = \beta E_D + \alpha E_w$$

Et $Z_R(\alpha,\beta)$ est la constante de normalisation qui est donnée par :

$$Z_R(\alpha,\beta) = \int \exp(-R(w))dw \tag{3.4}$$

En général cette intégrale ne peut pas être calculer analytiquement.
Dans ce cadre Makay a proposé une estimation de la probabilité a postériori via l'hypothèse de l'approximation gaussienne.

3.3.4 Approximation gaussienne de la distribution a posteriori

La distribution a posteriori est multimodale, l'intégrale dans l'équation 3.4 peut être approcher en utilisant les approximations Gaussiennes introduite dans [24]. Soit w_{MP} le maximum de $P(w/D,\alpha,\beta)$ i.e. w_{MP} soit le minimum de $R(w)$. En utilisant le développement de Taylor de $R(w)$ autour de w_{MP} à l'ordre 2

$$R(w) \approx R\left(w_{MP}\right) + \frac{1}{2}\left(w - w_{MP}\right)^T A\left(w - w_{MP}\right) \qquad Avec \quad A = \nabla^2 R\left(w_{Mp}\right)$$

Par conséquent, on écrit :

$$Z_R(\alpha,\beta) = \int \exp(-R(w))dw$$
$$\approx \exp\left(-R\left(w_{MP}\right)\right)\int \exp\left(-\frac{1}{2}\left(w - w_{MP}\right)^T A\left(w - w_{MP}\right)\right)dw$$

Et comme $I = \int \exp\left(\frac{1}{2}\left(w - w_{MP}\right)^T A\left(w - w_{MP}\right)\right)dw$ est une intégrale de Gauss, alors :

$$I = \sqrt{(2\pi)^l \det A^{-1}}$$

D'où

$$Z_R(\alpha,\beta) \approx \exp\left(-R\left(w_{MP}\right)\right)\sqrt{(2\pi)^l \det A^{-1}}$$

3.3.5 Structure d'évidence

Dans ce qui précède, pour calculer facteur de normalisation $Z_R(\alpha, \beta)$, et la constante de normalisation $Z_w(\alpha)$, les hyperparamètres α et β sont supposés fixes, (où β est un paramètre associé à la distributions de bruit et α est associé à la distribution a priori des poids de réseau). Comme les différentes valeurs de ces paramètres agissent directement sur l'apprentissage des RNAs, alors il est raisonnable de déduire les valeurs de ces paramètres à partir des données collectées.

Dans cet axe, la règle de Bayes est appliquée pour la détermination des hyperparamètres α et β. On a :

$$P(\alpha, \beta|D) = \frac{P(D|\alpha, \beta)\, P(\alpha, \beta)}{P(D)} \tag{3.5}$$

Si on suppose que la distribution a priori de α et β est uniforme, alors la maximisation de la distribution a posteriori des hyperparamètres est équivalente à la maximisation de la vraisemblance $P(D|\alpha, \beta)$ (ou l'évidence). On remarque que la vraisemblance $P(D|\alpha, \beta)$ est exactement la constante de normalisation dans l'équation 3.1. Selon l'équation 3.1 et sous les mêmes hypothèses de section précédente (le bruit et la distribution a priori des poids sont normales), alors on obtient :

$$\begin{aligned}
P(D|\alpha, \beta) &= \frac{P(D|w, \beta)\, P(w|\alpha)}{P(w|D, \alpha, \beta)} \\
&= \frac{\frac{1}{Z_D(\beta)} \exp(-\beta E_D)\, \frac{1}{Z_w(\alpha)} \exp(-\alpha E_w)}{\frac{1}{Z_R(\alpha,\beta)} \exp(-R(w))} \\
&= \frac{\frac{1}{Z_D(\beta)} \frac{1}{Z_w(\alpha)} \exp(-(\beta E_D + \alpha E_w))}{\frac{1}{Z_R(\alpha,\beta)} \exp(-R(w))} \\
&= \frac{Z_R(\alpha, \beta)}{Z_D(\beta) Z_w(\alpha)}
\end{aligned}$$

Notant que les deux constantes $Z_D(\beta)$ et $Z_w(\alpha)$ sont connues, en effet $Z_D(\beta) = \left(\frac{2\pi}{\beta}\right)^{\frac{n}{2}}$ et $Z_w(\alpha) = \left(\frac{2\pi}{\alpha}\right)^{\frac{l}{2}}$.

Les valeurs optimales des hyperparamètres α_{MP} et β_{MP} correspondent à une maximisation de l'évidence. Pour ce faire, nous utilisons la fonction logarithmique log, ce qui équivaut à considérer que maximiser une probabilité revient à minimiser la fonction erreur qui s'y rattache.

$$\begin{aligned}
\log P(D|\alpha, \beta) &= -R(w_{MP}) - \frac{1}{2} \log \det A - \log Z_w(\alpha) - \log Z_D(\beta) + \frac{l}{2} \log 2\pi \\
&= -\beta E_D(w_{MP}) - \alpha E_w(w_{MP}) - \frac{1}{2} \log \det A - \frac{l}{2} \log(\alpha) - \frac{n}{2} \log(\beta) + c
\end{aligned}$$

Avec c est une constante. Pour caractériser les hyperparamètres les plus probables, α_{MP} et β_{MP}, on dérive la fonction $\log P(D|\alpha, \beta)$ par rapport à α et β, et on cherche le point où les deux

dérivées sont nulles. Tout d'abord on a :

$$\frac{\partial \log P\left(D|\alpha,\beta\right)}{\partial \alpha} = -E_w\left(w_{MP}\right) - \frac{1}{2}\frac{\partial \log \det A}{\partial \alpha} - \frac{l}{2\alpha}$$
$$\frac{\partial \log P\left(D|\alpha,\beta\right)}{\partial \beta} = -E_D\left(w_{MP}\right) - \frac{1}{2}\frac{\partial \log \det A}{\partial \beta} - \frac{n}{2\beta}$$

Dans le cas de régularisation par la méthode de weight decay [5], alors $A = \alpha I_l + \beta B$, avec I_l est la matrice unité et $B = \nabla^2 E_D\left(w_{MP}\right)$, et donc

$$\begin{aligned}\frac{\partial \log \det A}{\partial \alpha} &= \text{Trace}\left(A^{-1}\frac{\partial A}{\partial \alpha}\right)\\ &= \text{Trace}\left(A^{-1}I_l\right)\\ &= \text{Trace}\left(A^{-1}\right)\end{aligned}$$

Cette relation n'est pas exacte que dans le cas où E_D et E_w sont des fonctions quadratiques, dans le cas contraire, le dernier résultat est une approximation de $\frac{\partial \log \det A}{\partial \alpha}$.

En résolvant les deux équations suivantes :

$$\frac{\partial \log P\left(D|\alpha,\beta\right)}{\partial \alpha} = 0 \qquad \frac{\partial \log P\left(D|\alpha,\beta\right)}{\partial \beta} = 0$$

Alors on obtient

$$\alpha_{MP} = \frac{\gamma}{2E_w\left(w_{MP}\right)} \qquad \beta_{MP} = \frac{n-\gamma}{2E_D\left(w_{MP}\right)}$$

Avec $\gamma = l - \alpha_{Mp}$ Trace $\left(A^{-1}\right)$ s'appelle le nombre effectif des paramètres, il mesure le nombre des paramètres de réseau qui contribuent d'une manière effective à la réduction de risque considéré et vaut.

$$\gamma = \sum_{k=1}^{l} \frac{\lambda_k}{\lambda_k + \alpha}$$

λ_k représente les valeurs propres de la matrice hessienne H de l'erreur E_D, i.e :

$$H = \beta \nabla^2 E_D$$

Maintenant que les hyperparamètres peuvent être optimisés, résumons le processus conduisant au jeu de poids le plus probable.

Algorithm 4 Evidence framework

1. Initialisation
On choisit des valeurs de α et β. Ce qui permet d'initialiser les poids du réseau en utilisant les valeurs provenant de la distribution a priori.
À l'itération i on aura donc une estimation des poids w^i, des hyperparamètres α^i et β^i et de l'erreur $R^i(w)$.
2. déterminer les valeurs w_{MP}^{i+1} qui minimisent l'erreur $S^i(w)$ en utilisant, une méthode d'optimisation non linéaire, par exemple, l'algorithme du gradient conjugué. Après obtention du jeu de poids le plus probable à l'instant $i+1$ on calcule E_w^{i+1} et E_D^{i+1}
3. La ré-estimations des paramétres :
On calcule les nouvelles valeurs des hyperparamètres

a)

$$\gamma^{i+1} = \sum_{k=1}^{l} \frac{\lambda_k}{\lambda_k + \alpha^i}$$

b)

$$\alpha^{i+1} = \frac{\gamma^{i+1}}{2E_w^{i+1}}$$

c)

$$\beta^{i+1} = \frac{n - \gamma^{i+1}}{2E_D^{i+1}}$$

3. Répétez les étapes 2 et 3 en utilisant les nouvelles valeurs des hyperparamètres.

Le processus s'arrêtera lorsque l'erreur $R(w)$ sera égal à la moitié du nombre de points des données, soit $R(w) = \frac{n}{2}$, alors :

$$\alpha = \alpha_{MP}$$

et

$$\beta = \beta_{MP}$$

3.3.6 Sélection du modèle bayésien

D'un point de vue Bayésien, Nous pouvons sélectionner le modèle le plus probable en fonction de son évidence $P(D/M_i)$, Si cette dernière est plus grande alors le modèle associé est le meilleur.

$$P\left(M_i|D\right) = \frac{P\left(D|M_i\right)P(M_i)}{P\left(D\right)} \tag{3.6}$$

Où $P(M_i)$ est la probabilité a priori associé au modèle M_i.
Dans le cadre des réseaux de neurones on considère que chaque modèle depend que du nombre de neurones dans la couche caché. Graphiquement on peut représenté le logarithme de l'évidence en fonction du nombre de neurones cachées.

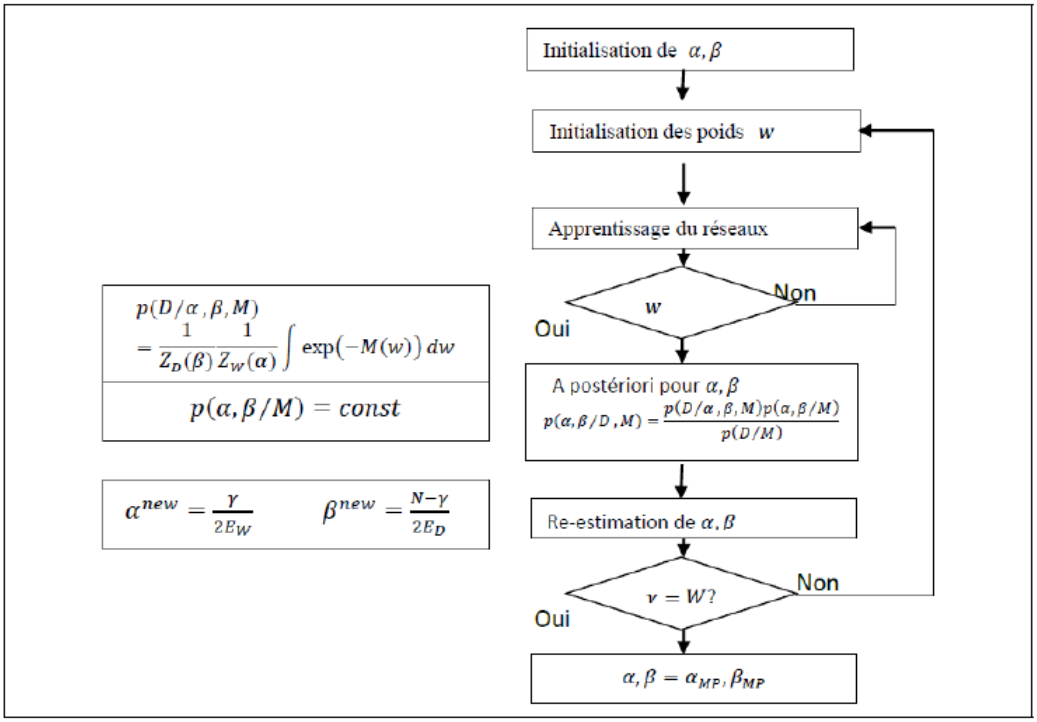


Figure 3.1 – L'architecture générale de la procédure d'évidence.

3.3.7 Procédure d'évidence pour un problème de classification

On considère un problème de classification avec c classes $\{C_1, \cdots, C_c\}$, et soit un Perceptron Multicouche, dont la couche de sortie est composée de c neurones.

Afin de discriminer les modèles appartenant à une classe C_k parmi les c classes, les unités de sortie $\{y_1, ..., y_c\}$, doivent être interprétées comme les probabilités a posteriori $p(C_k/x)$ que le modèle x appartient à la classe C_k . Ceci est obtenu en imposant deux conditions sur les fonctions de sortie :

$$y_k(x) \in [0,1] \quad et \quad \sum_{k=1}^{c} y_k(x) = 1$$

La fonction d'activation pour les neurones de sortie est donnée par la fonction softmax (softmax function) :

$$y_k(x) = \frac{\exp(a_k(x))}{\sum_{i=1}^{c} \exp(a_i(x))} \tag{3.7}$$

Avec $a_k(x)$ est donné par la somme pondérée des valeurs de sortie de toutes les neurones de la couche précédente.

La probabilité de mauvaise classification est minimisée en attribuant le modèle x à la classe C_k lorsque :

$$y_k(x) > y_l(x) \qquad \forall l = 1, \cdots, c \;\; l \neq k$$

La prédiction bayésienne pour une nouvelle entrée x est donnée par la marginalisation de la classification effectuée par le modèle $p(C_k/x, w)$ par rapport à la distribution a posteriori des paramètres :

$$p(C_k/x, D) = \int p(C_k/x, w)p(w, \alpha/D)dwd\alpha \tag{3.8}$$

Le deuxième terme de l'intégrale est donné par :

$$p(w, \alpha/D) = p(w/\alpha, D)p(\alpha/D) \tag{3.9}$$

Comme l'intégrale 3.8 ne peut pas être calculé analytiquement, nous devons utiliser des approximations numériques.

Procédure d'évidence

La procédure d'évidence donne une approximation pour l'équation (3.7) en supposant que la probabilité a posteriori de l'hyperparamètre $p(\alpha/D)$ est fortement creusé autour de son maximum α_{MP}. Pour refléter l'absence de connaissances a priori sur la meilleure valeur de α, la probabilité a priori $p(\alpha)$ est choisi pour avoir la forme du produit $p(\alpha) = \prod_g p(\alpha_g)$ où g indexe un groupe tel que chaque $p(\alpha_g)$ étant constant sur une échelle logarithmique. Ainsi, on peut trouver la valeur d'une maximisation du $\log p(\alpha/D)$ en maximisation $\log p(D/\alpha)$.
$p(D/\alpha)$ est donnée par la formule :

$$p(D/\alpha) = \int p(D/w)p(w/\alpha)dw$$

Dans la procédure d'évidence l'intégrale est approximée en utilisant l'approximation de Laplace autour de w_{MP}, la valeur de w qui maximise $p(w/\alpha, D)$. Il s'avère que les composants de la valeur optimale α_{MP} sont donnés par :

$$\alpha_g^{MP} = \frac{\gamma_g}{\sum_{i \in g}(w_i^{MP})^2}$$

Avec

$$\gamma_g = w_g - \alpha_g \sum_{i \in g}((\nabla\nabla G + \alpha I)^{-1})_{ii}$$

w_g est le nombre total des poids contrôlés par l'hyperparamètre α_g.

G est le négatif de la fonction vraisemblance pénalisée : $G = -\ln p(D/w) - \ln p(w/\alpha)$.

La somme prend en compte les éléments de la matrice $(\nabla\nabla G + \alpha I)^{-1}$ correspondant aux poids appartenant au groupe g.

3.4 Intégration des méthodes MCMC dans l'apprentissage des RNA

Les méthodes Monte Carlo par chaînes de Markov(MCMC), s'inscrivent dans le cadre du formalisme Bayésien. On peut dire que l'application de ces méthodes à l'apprentissage des réseaux de

neurones peut apporter plusieurs avantages, tout d'abord, l'introduction de connaissances a priori est susceptible d'améliorer l'estimation des paramètres. De plus, le formalisme Bayésien permet une analyse complète des incertitudes et des probabilités des données.

Le processus d'apprentissage bayésienne commence par la définition du modèle, et la distribution a priori $P(\theta)$ pour le modèle paramétrique. La distribution a priori exprime les connaissances sur les valeurs des paramètres avant que les données n'aient été observées. Après avoir observé de nouvelles données $D_n = \{(x_1, y_1), (x_2, y_2), .., (x_n, y_n)\}$, la distribution a priori est transformée en une distribution a posteriori via le théorème de Bayes :

$$P\left(\theta/D_n\right) = \frac{P\left(D_n/\theta\right)P(\theta)}{P\left(D_n\right)} \propto L\left(\theta/D_n\right)P(\theta) \tag{3.10}$$

Où la fonction vraisemblance $L(\theta|D_n)$ donne la probabilité que les données observées fonctionnent selon les paramètres inconnu du modèle. Dans le cas de points de données indépendants et échangeables, nous obtenons :

$$L\left(\theta/D_n\right) = \prod_{i=1}^{n} P(y_i/x_i, \theta)$$

Où n est le nombre des données.

Nous considérons un réseau PMC avec des poids w. PMC prend le vecteur d'entrée x et génère une sortie $d = f\left(x, w\right)$ qui représente la fonction de régression. Dans les problèmes de régression, on suppose généralement que la distribution de les données cibles peuvent être décrite par une fonction déterministe des entrées, corrompue par un bruit Gaussien additif avec une matrice de covariance constante. La densité de probabilité pour une cible y est alors :

$$P\left(y/x, w, A\right) = (2\pi)^{-\frac{d}{2}}|A|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(x-\mu\right)^T A^{-1}\left(x-\mu\right)\right)$$

Où A est la matrice variance-covariance.

En utilisant différentes paramétrisations de la matrice de covariance, nous faisons différentes hypothèses sur le bruit. Les paramètres qui ont été généralement utilisés sont des matrices de covariance diagonales constantes (σI) et diagonales $(\mathrm{diag}\left(\sigma_1^2, ..., \sigma_1^2\right))$. Ceux-ci correspondent à l'hypothèse d'un bruit indépendant entre les sorties, avec un niveau de bruit commun ou des niveaux de bruit indépendants, respectivement. Les densités de probabilité pour une cible y sont données respectivement par :

$$P\left(y/x, w, \sigma^2\right) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}\left(y-d\right)^2\right) \tag{3.11}$$

$$P\left(y_j/x, w, \sigma_j^2\right) = (2\pi\sigma_j^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\sigma_j^2}\left(y_j-d_j\right)^2\right) \tag{3.12}$$

Notez que l'utilisation d'une covariance diagonale constante avec une approche de maximum de vraisemblance correspond à la minimisation de l'erreur de somme des carrés. Un préalable conjugué

pratique, qui nous permet de définir facilement notre connaissance préalable (ou ignorance) de la variance du bruit, est donné en général par :

$$\begin{aligned} P\left(\sigma^2\right) &= Inv - Gamma\left(\sigma^2/\nu_0, \sigma_0^2\right) \\ &\propto \left(\sigma^2\right)^{\left(\left(-\frac{\nu_0}{2}\right)+1\right)} \exp\left(-\frac{1}{2}\nu_0\sigma_0^2\sigma^{-2}\right) \end{aligned} \tag{3.13}$$

Où Inv-Gamma est l'inverse de la distribution de la loi Gamma avec ν degré de liberté.

De cette manière, la connaissance a priori à propos de σ est équivalente à ν_0 mesures a priori ayant une variance commune σ_0. Parfois, les distributions antérieures sont spécifiées en termes de précisions correspondantes $\tau = \sigma^{-2}$, avec des distributions Gamma données. La distribution gamma dans l'analyse bayésienne a été discutée, par plusieurs mathématiciens, par exemple, par Box et Tiao en 1973. La distribution gamma en rapport avec les PML a été discutée par Neal en 1996 [15].

Pour prédire une nouvelle sortie y^{n+1} pour une nouvelle entrée x^{n+1}, la distribution prédictive est obtenue en intégrant les prédictions du modèle par rapport à la distribution a posteriori des paramètres du modèle :

$$P\left(y^{n+1}/x^{n+1}, D_n\right) = \int P\left(y^{n+1}/x^{n+1}, \theta\right) P\left(\theta/D_n\right) d\theta \tag{3.14}$$

Les attentes peuvent être évaluées par rapport à la distribution a posteriori des paramètres, par exemple pour y^{n+1} :

$$\hat{y}^{n+1} = \int f\left(x^{n+1}, w\right) P\left(\theta/D_n\right) d\theta \tag{3.15}$$

Les distributions a posteriori dans le cas des PMC sont généralement très complexes et les intégrales sont très difficiles à évaluer. Neal[15] a introduit la mise en oeuvre de l'implantation bayésienne pour les réseaux de neurones par Les méthodes Monte Carlo par chaînes de Markov (MCMC). Dans MCMC, les intégrations requises par l'approche bayésiennes ont approchées numériquement à l'aide d'un échantillon de valeurs tirées de la distribution a posteriori des paramètres. Par exemple (3.6) est approximé par :

$$\hat{y}^{n+1} \approx \sum_{t=1}^{N} f\left(x^{n+1}, w^t\right) \tag{3.16}$$

Où w^t sont des échantillons des poids.

Dans les méthodes MCMC, les échantillons sont générés à l'aide d'une chaîne de Markov qui a la distribution a posteriori souhaitée comme distribution limite.

3.5 Conclusion

Les méthodes bayésiennes pour l'apprentissage des réseaux de neurones peuvent apporter plusieurs avantages, car il n'est pas nécessaire de limiter la taille du réseau pour éviter le surajustement, et que le nombre de neurones cachés peut tendre vers l'infini, le seul facteur qui doit limiter

la taille du réseau est la capacité des ordinateurs utilisés et le temps disponible pour effectuer les calculs nécessaires, mais comme les paramètres utilisés sont issus d'une distribution de probabilité, il est nécessaire pour connaître un paramètre de calculer des intégrales faisant intervenir les distributions des autres paramètres. Il est, en général, impossible de calculer ces intégrales analytiquement, et plusieurs approche sont été proposées pour effectuer ces calculs.

CHAPITRE 4

APPLICATION À LA RECONNAISSANCE DES VISAGES

4.1 Introduction

Le système de la reconnaissance faciale est devenu un besoin très demandé parce que c'est un système qui assure la sécurité et la protection des individus, un système de sécurité basé sur la reconnaissance de visage est fiable par rapport aux autres systèmes qui nécessitent un code PIN ou d'autres techniques d'accès qui peuvent être volées ou pirater facilement, car il se base sur un compensant principale d'identification chez l'être humain qui est le visage, en revanche la reconnaissance faciale assure pour les individus que personne ne peut accéder à leur compte, de plus ce sont des systèmes très facile à utiliser.

4.2 La biométrie

La biométrie est une technique qui permet de reconnaître des personnes à partir de leurs caractéristiques physiques et comportementales :

- **La biométrie physiologique :** est basée sur l'identification de traits physiques particuliers qui, pour toute personne, sont uniques et permanents. Cette catégorie regroupe la reconnaissance des empreintes digitales, la forme de la main, la forme du visage, la rétine et l'iris de l'œil et les veines de la main [36].

- **La biométrie comportementale :** se base sur l'analyse de certains comportements d'une personne comme le tracé de sa signature, l'empreinte de sa voix, son mouvement de lèvre ou encore sa façon de taper sur un clavier.

En théorie, on dit qu'un critère physiologique ou comportemental est exploitable en reconnaissance d'individus s'il satisfait aux conditions ci-après[37] :

- **Universalité :** toutes les personnes d'une population à identifier doivent la posséder.
- **Unicité :** deux personnes ne doivent pas posséder exactement la même caractéristique.
- **Permanence :** cela signifie qu'elle ne doit pas varier au cours du temps.
- **Mesurabilité :** qui justifie de la possibilité de le quantifier.

De nombreuses modalités biométriques sont utilisées dans diverses applications (voir figure 4.1). Chaque modalité biométrique a ses forces et ses faiblesses et le choix dépend généralement de l'application à traiter. Aucune modalité biométrique ne répond effectivement aux exigences de toutes les applications. Ainsi, l'utilisation de plusieurs modalités biométriques pour l'identification d'un individu augmente aussi bien la précision que la couverture de la population.

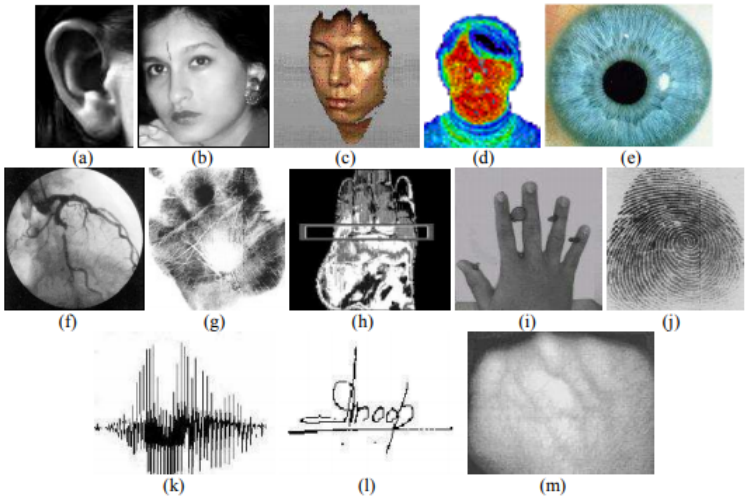


Figure 4.1 – Les principales caractéristiques biométriques pouvant servir comme moyen d'identification : a) forme de l'oreille, b) visage 2D, c) visage 3D, d) visage infrarouge, e) iris, f) rétine, g) empreinte de la main, h) thermogramme de la main, i) forme de la main, j) empreinte digitale, k) voix, l) signature et m) réseau veineux de la main.

4.2.1 Biométrie du visage

La reconnaissance de visages humains est considérée actuellement comme l'une des tâches les plus importantes pour relever le défi en reconnaissance des formes. La facilité et la précision avec lesquelles nous pouvons identifier nos amis et nos ennemis, même dans des conditions défavorables, sont des capacités des plus étonnantes du système visuel humain.

Un système de reconnaissance de visage est un système d'identification et de vérification d'individus, qui permet de vérifier si une personne appartient à la base de donnée du système, et de l'identifier si c'est le cas. On peut représenter les systèmes de reconnaissance par le diagramme suivant (voir figure 4.2).

- **Détection puis localisation :**

  Les systèmes de reconnaissance de visages sont complexes. La difficulté réside notamment dans la partie détection automatique du visage, bien que nous développerons surtout la

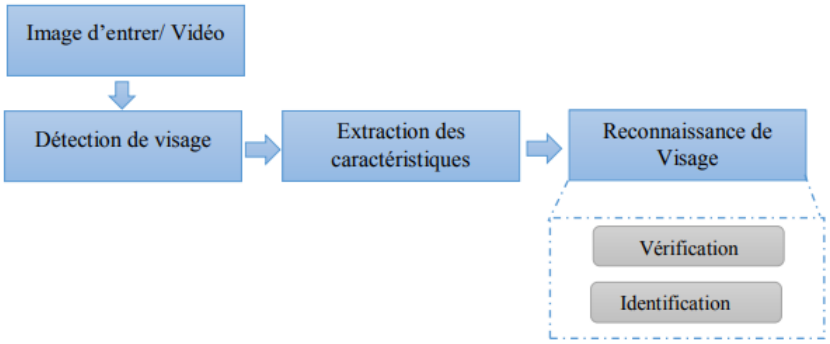


Figure 4.2 – Système de reconnaissance de visage.

partie reconnaissance, il est intéressant de parler de l'étape de détection automatique du visage qui est très importante dans un système de reconnaissance. Ce qui rend la détection de visages dans une image très difficile, c'est surtout la complexité du décor, les variations de poses, les conditions de lumières généralement inconnues, etc. Il existe plusieurs méthodes qui peuvent être appliquées à la détection automatique des visages [40]. Il faut détecter la présence d'un visage dans l'image, ensuite le localiser en vue d'extraire les traits pour le caractériser et le différentier des autres. Le résultat de cette étape est l'obtention de la partie d'image à traiter.

Les approches de détection peuvent être divisées en deux catégories : les approches locales et les approches globales [45] :

▷ **Les approches locales**

Les approches locales, les plus anciennes, reposent sur une analyse bas niveau de l'image par l'étude des coins, de l'intensité, de la couleur ou du mouvement. D'autres approches utilisent la mise en correspondance de modèles locaux, statistiques ou déformables. Ces modèles sont généralement localisés autour des caractéristiques faciales (yeux, nez, ...) et nécessitent donc la détection de celles-ci. Une fois les caractéristiques détectées, elles sont organisées de manière à définir un modèle plus global des visages, en tenant compte d'un ensemble de contraintes géométriques. Ces techniques reposent sur le choix d'un bon compromis entre influence de l'information globale et des modèles locaux, de manière à fournir au système, par exemple, une certaine robustesse aux occultations partielles des visages. La recherche des caractéristiques faciales étant menée dans l'intégralité de l'image, elle est fortement influencée par le bruit de l'image. Par conséquent, l'imprécision en termes de détection de caractéristiques faciales.

▷ **Les approches globales**

Les méthodes globales, c'est-à-dire basées sur la détection du visage dans sa globalité, ont été introduites dans le but d'être appliquées à des images contenant plusieurs visages et/ou en présence d'un fond complexe. Ces techniques évitent les problèmes d'imprécision dans la détection des caractéristiques faciales par le biais de l'apprentissage des règles intrinsèques des visages. Pour cela, il est nécessaire de disposer d'un volume de

données important et présentant une grande variabilité dans les conditions d'éclairage, de prise de vue, ... Ces dernières approches sont en général plus robustes au bruit et à des déformations de la région faciale que les approches locales.

- **Extraction des caractéristiques :**

  Dans cette étape on extrait de l'image les informations qui seront sauvegardées en mémoire pour être utilisées plus tard dans la phase de Décision. Le choix de ces informations utiles revient à établir un modèle pour le visage, elles doivent être discriminantes et non redondantes. Ces informations seront ensuite classées, en d'autre terme, affectées à la classe la plus proche, les individus ayant des similarités sont regroupés dans la même classe.

- **Reconnaissance du visage :**

  Le module de reconnaissance exploite les caractéristiques du visage ainsi extraites pour créer une signature numérique qu'il stocke dans une base de données. Ainsi, à chaque visage de la base est associée une signature unique qui caractérise la personne correspondante. La reconnaissance d'un visage requête est obtenue par l'extraction de la signature requête correspondante et sa mise en correspondance avec la signature la plus proche dans la base de données. La reconnaissance dépend du mode de comparaison utilisé : vérification ou identification.

4.2.2 Principales difficultés de la reconnaissance de visage

La reconnaissance de visage est la méthode la plus naturelle car c'est la méthode utilisée inconsciemment par tous les individus grâce à l'interaction visuelle. Elle est surtout utilisée dans des casinos, des stades, mais également dans les aéroports.

Le problème de ces méthodes est le manque d'efficacité si les images en entrée sont de faible luminosité ou si certaines parties d'un visage sont masquées. De plus, le fait qu'un visage soit ombragé rend l'extraction des caractéristiques faciales très difficiles. Il existe deux types de variations associées aux images de visages : inter et intra sujet. La variation inter-sujet est limitée à cause de la ressemblance physique entre les individus. Par contre la variation intra-sujet est plus vaste. Elle peut être attribuée à plusieurs facteurs que nous analysons ci-dessous [38] :

Variation de pose

Un changement de l'angle d'inclinaison du visage engendre, pour une position fixe du capteur, de nombreux changements d'apparence dans l'image collectée (voir figure 4.3). Nous nous intéressons ici aux rotations du visage en profondeur tels que les mouvements de type hochement de tête ou négation. En effet, la phase préliminaire de normalisation du visage permet de corriger d'éventuelles rotations dans le plan de l'image. Les rotations en profondeur engendrent l'occultation de certaines parties du visage. D'autre part, elles amènent des différences de profondeur qui, projetées sur le plan 2D de l'image, résultent en des déformations qui font varier la forme globale du visage. Ces déformations qui consistent à l'étirement de certaines parties du visage et le compactage d'autres régions font varier aussi les distances entre les caractéristiques faciales.

Figure 4.3 – Exemples de variation de poses (Images recueillies à partir d'Internet).

Expressions faciales

Les visages sont des objets non rigides. L'expression faciale de l'émotion, combinée avec la parole, peut produire des changements d'apparence importants des visages (voir figure 4.4). Le nombre de configurations possibles est incalculable. L'influence de l'expression faciale sur la reconnaissance est donc difficile à évaluer. Puisque l'expression faciale affecte la forme géométrique et les positions des caractéristiques faciales, il semble logique que les techniques globales ou hybrides y soient plus robustes que la plupart des techniques géométriques. Gross et al [45] soutiennent que les expressions faciales n'ont pas une grande influence sur les algorithmes de reconnaissance, pour autant qu'elles restent raisonnables. Alors que les cas extrêmes qui engendrent d'importantes déformations de la bouche (tel que le cri) et le rétrécissement ou la fermeture complète des yeux dégradent considérablement les performances de la reconnaissance automatique.

Figure 4.4 – Exemples de variation d'expressions.

Changement d'illumination

L'intensité et la direction d'éclairage lors de la prise de vue influent énormément sur l'apparence du visage dans l'image (voir figure 4.5). Dans la plupart des applications réelles, des changements dans les conditions d'éclairage sont néanmoins inévitables, notamment lorsque les vues sont collectées à des dates différentes, en intérieur ou en extérieur. Etant donné la forme intrinsèque d'un visage humain, des changements d'éclairage peuvent faire apparaître sur le visage des ombres accentuant ou masquant certaines caractéristiques faciales. L'évaluation FRVT conclue que des changements importants dans les conditions d'éclairage peuvent mener à des baisses considérables dans les taux de reconnaissance.

Présence ou absence des composants structurels

La présence des composants structurels telle que la barbe, la moustache, ou bien les lunettes peut modifier énormément les caractéristiques faciales telles que la forme, la couleur, ou la taille du visage. De plus, ces composants peuvent cacher les caractéristiques faciales de base causant ainsi

Figure 4.5 – Exemple de variation d'éclairage.

une défaillance du système de reconnaissance. Par exemple, des lunettes opaques ne permettent pas de bien distinguer la forme et la couleur des yeux, et une moustache ou une barbe modifie la forme du visage.

Occultations partielles

Le visage peut être partiellement masqué par des objets dans la scène, ou par le port d'accessoires tels que les lunettes de soleil, une écharpe, un chapeau (voir figure 4.6), ... Les occultations peuvent être intentionnelles ou non. Dans le contexte de la vidéo surveillance, par exemple, il peut s'agir d'une volonté délibérée de contrecarrer la reconnaissance. La plupart des systèmes déployés doivent être non intrusifs, i. e : ne comptant pas sur une coopération du sujet. Par conséquent, il est important de savoir reconnaître des visages partiellement occultés. Gross et Al ont évalué l'impact du port de lunettes de soleil, et d'un cache nez occultant la partie inférieure du visage.

Figure 4.6 – Exemples d'occlusion du visage (Image recueillie à partir d'Internet).

4.2.3 Applications des systèmes biométriques

Les applications de la biométrie peuvent être divisées en trois groupes principaux [38] :

- **Applications commerciales :**

  telles que l'ouverture de réseau informatique, la sécurité de données électroniques, l'accès Internet, la carte de crédit, le contrôle d'accès physique, la gestion des registres médicaux, l'étude à distance, etc.

- **Applications gouvernementales :**

  telles que la carte d'identité nationale, le permis de conduire, la sécurité sociale, le contrôle des passeports, etc.

- **Applications légales :** telles que l'identification de corps, la recherche criminelle, l'identification de terroriste, etc.

4.2.4 L'extraction des paramètres

L'étape de l'extraction des paramètres représente le cœur du système de reconnaissance, elle consiste à effectuer le traitement de l'image dans un autre espace de travail plus simple et qui assure une meilleure exploitation de données, et donc permettre l'utilisation, seulement, des informations utiles, discriminantes et non redondantes.

4.3 Traitement d'image et Description des visages

4.3.1 Traitements des images

Définition d'une image

C'est la représentation d'une personne ou d'un objet par la peinture, la sculpture, la photographie, etc. C'est aussi un ensemble structuré d'informations qui après affichage sur écran, ont une signification pour l'oeil humain. Elle peut être écrite sous forme de fonction $I(x, y)$, où I est une fonction d'intensité lumineuse ou de couleur aux coordonnées spatiales (x, y). De cette façon l'image est exploitable par la machine, d'où la nécessité de sa numérisation.

L'image numérique

Il est clair que les images manipulées par l'ordinateur sont numériques (série de bits). L'image numérique est l'image dont la surface est divisée en éléments de taille fixe appelés pixels, ayant comme caractéristique le niveau de gris ou de couleur. La numérisation d'une image est la conversion de celle-ci en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques $I(x, y)$ qui sont les niveaux de gris des coordonnées réelles (x, y). L'image numérique se caractérise par :

- **Pixel :** C'est le plus petit point de l'image. Chaque pixel a une valeur numérique qui représente le niveau de gris ou de couleur selon la nature de l'image.

- **Dimension :** C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentant les intensités lumineuses (pixel).

- **Résolution :** C'est la clarté ou la finesse des détails atteinte par un moniteur ou une imprimante dans la production d'image, sur les moniteurs d'ordinateurs, la résolution est exprimée en nombre de pixel par unité de mesure (pouce ou centimètre). On utilise aussi le mot résolution pour désigner le nombre total de pixels affichable horizontalement ou

verticalement sur un moniteur, plus grand est ce nombre, meilleure est la résolution.

- **Voisinage d'un pixel** [41] : Le voisinage d'un pixel est l'ensemble des pixels qui lui sont adjacents.

- **Niveaux de gris :** C'est la valeur numérique qui reflète l'intensité de la luminosité d'un point. Pour 256 niveaux de gris compris entre 0 et 255, chaque pixel sera codé sur 8 bits, et ses niveaux de gris seront obtenus après dégradation de la couleur noire, 0 représente le blanc et 255 représente le noir.

4.3.2 Description des visages par Matrice de co-occurrence

La texture est un ensemble de pixels (niveaux de gris) spatialement agencés selon un certain nombre de relations spatiales, ainsi créant une région homogène. Il y a plusieurs approches et modèles qui ont été proposées pour la caractérisation de la texture, parmi les plus connue on a **la matrice de co-occurrence.** Qui est largement utilisée dans le monde du traitement d'image. Elle présente une grande simplicité de mise en œuvre et donne de bons résultats sur la plupart des types d'images.

Matrice de co-occurrence mesure la probabilité d'apparition des paires de valeurs de pixels situés à une certaine distance d dans l'image. Elle est basée sur le calcul de la probabilité $P(i, j, d, \theta)$ qui représente le nombre de fois qu'un pixel de niveau de gris i apparaît à une distance relative d d'un pixel de niveau de gris j et selon une orientation θ donnée (voir la figure 4.7).

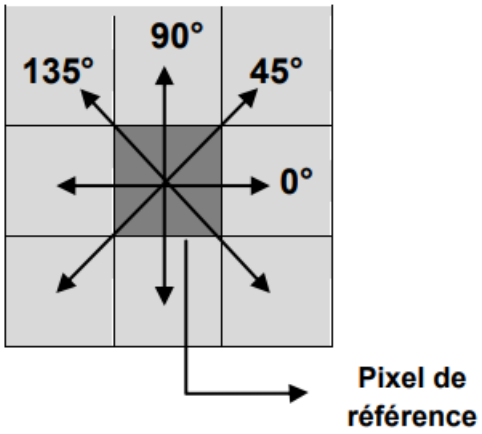


Figure 4.7 – Orientations principales considérées pour le calcul des GLCMs.

Supposons qu'une image à analyser soit rectangulaire et ait N_x cellules de résolution dans le sens horizontal et N_y cellules de résolution dans le sens vertical. Supposons que le niveau de gris apparaissant dans chaque cellule de résolution est quantifié à N_g niveaux. Soit $L_x = \{1, 2, \ldots, N_x\}$ le domaine spatial horizontal, $L_y = \{1, 2, \ldots, N_y\}$ le domaine spatial vertical, et $G = \{1, 2, \ldots, N_g\}$ représente l'ensemble des N_g niveaux de gris quantifiés. L'ensemble $L_y \times L_x$ est l'ensemble des cellules de résolution du image ordonnée par leurs désignations de ligne-colonne. L'image peut être représentée comme une fonction I qui attribue du niveau de gris à chaque cellule de résolution ou

paire de coordonnées dans $L_y \times L_x$.

$$I : L_y \times L_x \longrightarrow G.$$

Les matrices obtenues selon ces quatre directions (voir figure 4.7) sont alors calculées par les formules suivantes [42] :

Pour $\theta = 0$:

$$P(i,j,d,0) = Card\{((k,l),(m,n)) \in (L_y \times L_x)^2 / k-m=0, |l-n|=d, I(k,l)=i, I(m,n)=j\}$$

Pour $\theta = 45$:

$$\begin{aligned} P(i,j,d,45) = Card\{((k,l),(m,n)) \in (L_y \times L_x)^2 / (k-m=d, l-n=-d) \\ ou\ (k-m=-d, l-n=d), I(k,l)=i, I(m,n)=j\} \end{aligned}$$

Pour $\theta = 90$:

$$P(i,j,d,90) = Card\{((k,l),(m,n)) \in (L_y \times L_x)^2 / |k-m|=d, l-n=0, I(k,l)=i, I(m,n)=j\}$$

Pour $\theta = 135$:

$$\begin{aligned} P(i,j,d,135) = Card\{((k,l),(m,n)) \in (L_y \times L_x)^2 / (k-m=d, l-n=d) \\ ou\ (k-m=-d, l-n=-d), I(k,l)=i, I(m,n)=j\} \end{aligned}$$

Les matrices de co-occurrence contiennent beaucoup d'éléments (elles sont de taille 256×256 éléments pour une image sur 256 niveaux de gris) et sont donc difficilement manipulables. Il est à noter d'ailleurs que, dans la pratique, les images sont souvent altérées afin de ne considérer qu'une partie des niveaux de gris (8, 16, 32 ...). C'est pourquoi, un certain nombre d'auteurs comme Zucker [43] ont essayé d'extraire de l'information de ces matrices à fin de mettre en évidence la structure des textures. Mais c'est Haralick et Al [44] qui ont proposé les premiers 14 paramètres, caractérisant les textures, issus de ces matrices. Voici 6 paramètres considérés comme étant les plus utilisés et les plus pertinents :

1. **L'énergie :** mesure l'uniformité de la texture. Elle atteint de fortes valeurs lorsque la distribution des niveaux de gris est constante ou de forme périodique. Le déplacement correspond alors à la période.

$$ENE = \sum_i \sum_j (P(i,j,d,\theta))^2$$

2. **Le contraste :** est d'autant plus élevé qu'il y a des passages d'un pixel très clair à un pixel très foncé dans l'image (notion usuelle de contraste). Il est fortement non-corrélé à l'énergie.

$$CST = \sum_i \sum_j \Big((i-j)^2 P(i,j,d,\theta)\Big)$$

3. **L'entropie :** mesure le désordre dans l'image. Elle atteint de fortes valeurs pour une texture aléatoire. Elle est fortement corrélée (à l'inverse) à l'énergie.

$$ENT = \sum_i \sum_j (\log(P(i,j,d,\theta))P(i,j,d,\theta))$$

4. **La variance :** mesure l'hétérogénéité de la texture. Elle augmente lorsque les niveaux de gris différent de leur moyenne. La variance est indépendante du contraste.

$$VAR = \sum_i \sum_j \Big((i-\mu)^2 P(i,j,d,\theta)\Big)$$

Avec μ représente la moyenne des niveaux de gris.

5. **La corrélation :** mesure la dépendance linéaire (relativement à θ et d) des niveaux de gris dans l'image. Elle n'est corrélée ni à l'énergie, ni à l'entropie.

$$COR = \sum_i \sum_j \left(\frac{(i-\mu)(j-\mu)P(i,j,d,\theta)}{\sigma}\right)$$

6. **Le moment inverse :** IDM (Inverse Différence Moment) mesure l'homogénéité de l'image. Ce paramètre est corrélé à une combinaison linéaire des variables ENE et CST.

$$IDM = \sum_i \sum_j \left(\frac{(P(i,j,d,\theta)}{(i-j)^2+1}\right)$$

De nombreuses études ont été menées en vue de caractériser, classifier, modéliser les textures à l'aide de ces paramètres.

La méthode d'extraction de ces paramètres basée sur le calcul des matrices de co-occurrence est une des méthodes les plus proches de la notion de texture. Elles mettent effectivement en évidence les relations existant entre les pixels de l'image en faisant intervenir l'aspect local (les niveaux de gris) et l'aspect spatial ((d,θ)).

4.4 Généralité sur les bases de données

Plusieurs bases de visages ont été développées pour l'évaluation des algorithmes de reconnaissance faciale. Chacune comporte des conditions de prises de vues différentes. Les bases les plus anciennes sont celles de ORL et Yale [45].

4.4.1 Base de données ORL

Les expériences, menées dans ce mémoire, utilisent une base d'images, très populaire dans ce domaine de recherche, qui est textbfl'ORL (Olivetti Research Laboratory). Cette base contient un ensemble de visage pris entre avril 1992 et 1994 au laboratoire d'Olivetti Research à Cambridge. La base de visages ORL (disponible gratuitement voir [40]) contient 40 personnes distinctes, chaque personne ayant 10 images différentes de son visage (voir figure 4.8), ce qui donne 400 images de visages au total, avec 256 niveaux de gris et de taille 112×92. Toute les images sont obtenues dans différentes situations, telles que temps différent, différents angles, différentes expressions (fermer les yeux, ouvrir les yeux, sourire, surpris, heureux. . .) et différentes détails (avec lunette, sans lunette, barbe, modèle de cheveux), et une inclinaison de plus ou moins 20 degrés. L'ensemble de ces images va être divisé en deux sous ensemble. Un ensemble représentant la base d'apprentissage pour entraîner le réseau de neurones, soit au total : 240 images. Le deuxième ensemble est l'ensemble des images de test (des images non utilisées dans la phase d'apprentissage), soit au total : 160 images.

Figure 4.8 – Extrait de la base ORL. Pour chacune des 40 personnes enregistrées, on dispose de 10 vues avec des changements de pose, d'expression et d'éclairage.

4.4.2 Base des données Yale

Yale est une base qui contient 15 sujets avec 11 images pour chaque sujet, de type gif avec résolution de 320×243 pixels, ces images présentent des variations d'expression et de l'illumination puisqu'elles sont prises sur trois angles d'éclairage(Figure4.9).

4.5 Apprentissage par les réseaux de neurones bayésiens

Notons qu'on va faire la reconnaissance des visages des 40 personnes représentant la base des données ORL.

Après avoir lire l'image qui contient le visage, et la transformer en une matrice tel que chaque composante de la matrice représente le niveau de gris du pixel associé dans l'image, on calcule la matrice de co-occurrence associée à cette matrice, et à partir de cette matrice de co-occurrence, on

Figure 4.9 – Quelques exemples extraits de la base de test Yale.

calcule les 6 caractéristiques pertinentes du visage, qui seront par la suite l'entrée du Perceptron multicouche. Les étapes avant apprentissage sont résumées dans la figure 4.10 :

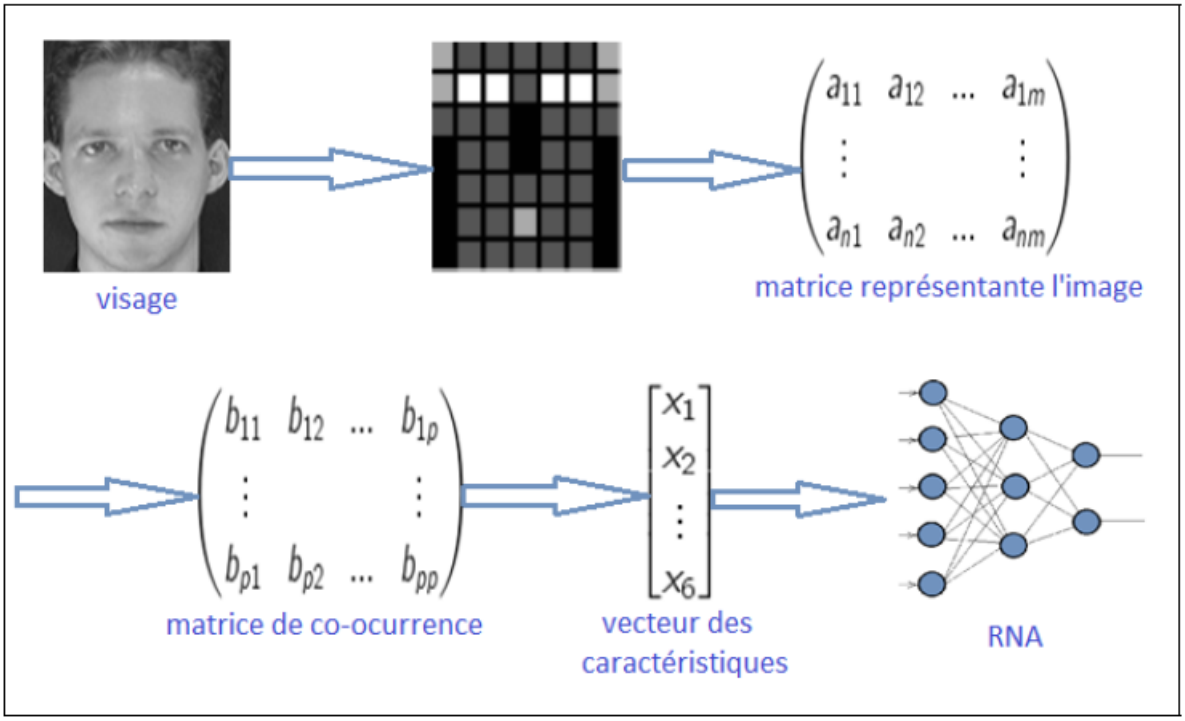


Figure 4.10 – Les étapes et la procédure avant apprentissage.

4.5.1 Codage

- La base ORL contient 40 personnes différentes, donc on aura besoin de 40 neurones dans la couche de sortie, de telle sorte que chaque neurone représente une personne.

- La sortie désirée associée à un vecteur des caractéristiques d'un visage du personne i est le vecteur nulle sauf dans la case i contient 1.

- Le vecteur des caractéristiques est de taille 6, alors on aura besoin de 6 neurones dans la couche d'entrée, de telle sorte que chaque neurone représente une caractéristique :
  — x_1 : représente l'énergie.
  — x_2 : représente le contraste.
  — x_3 : représente l'entropie.
  — x_4 : représente la variance.
  — x_5 : représente la corrélation.
  — x_6 : représente le moment inverse.

4.5.2 Structure du réseau neuronal

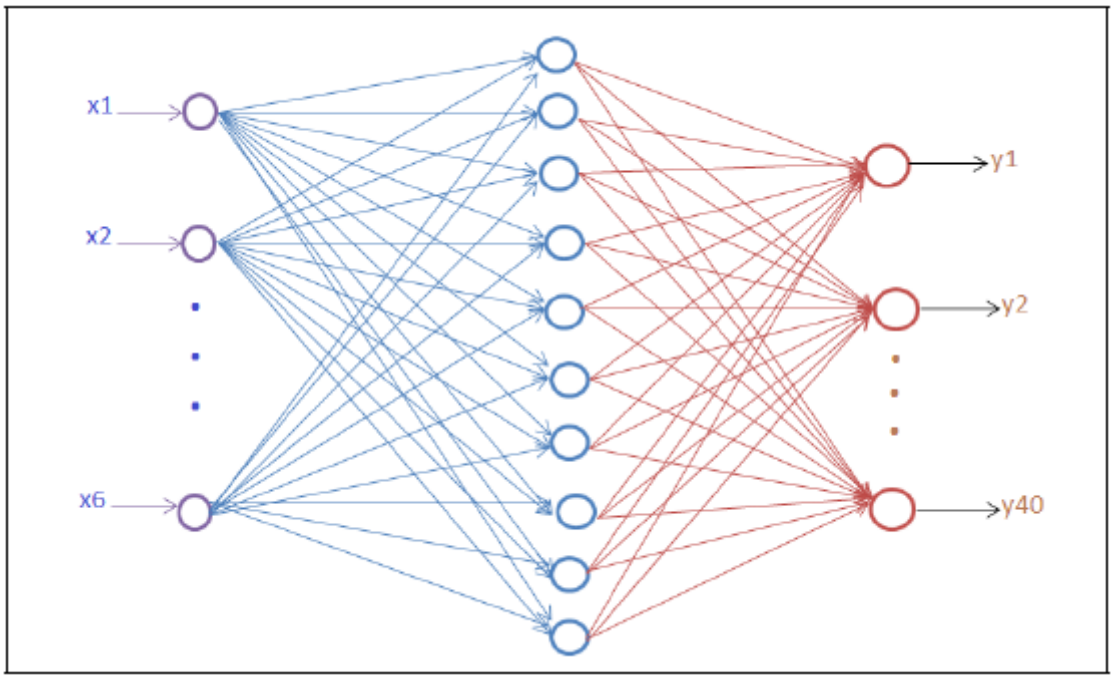


Figure 4.11 – Structure du réseau neuronal

4.5.3 Résultats numériques

⁕ **Taux de reconnaissance :**

Dans cette partie de notre programme on fait le test sur un ensemble des images qui n'appartiennent pas à l'ensemble des images d'apprentissage. Pour calculer le taux de reconnaissance, on divise le nombre des images du test reconnue sur le nombre total des images de test. On refait ces calcules pour des pourcentages différents des images d'apprentissage pour voir leurs influences sur le taux de reconnaissance.

Après avoir tourné l'algorithme de rétro-propagation du gradient dans un premier temps et la procédure d'évidence après, pour le Perceptron Multicouche dont l'architecture est décrite avant. On a aboutit à les résultats représentés dans le tableau suivant :

Pourcentages des images d'apprentissage (%)	10	20	30	40	50	60	70	80	90	95
Taux de reconnaissance (%) (Rétro-propagation de gradient)	20	32.6	40.8	50	52	53.5	60	67.74	71.98	80.13
Taux de reconnaissance (%) (Réseaux de neurones bayésiens)	19.35	35	40.11	53	55.13	58	63	69	75.42	85

Table 4.1 – Comparaison entre deux algorithmes par leur taux de reconnaissance sous l'influence du pourcentage des images.

⁕ **Commentaires :**

- Le pourcentage des images d'apprentissages influence sur le taux de reconnaissance, généralement lors que on augmentent le pourcentage des images d'apprentissages le taux de reconnaissance augmente aussi, ce qui montre l'importance d'augmenter le nombre d'image d'apprentissage.

- D'après le tableau nous remarquons que dans la plupart des cas, le taux de reconnaissance pour les réseaux de neurones bayésiens est plus élevé que celui-ci pour l'apprentissage par rétro-propagation du gradient, ce qui montre l'importance de l'intégration des méthodes bayésiennes dans l'apprentissage des réseaux de neurones.

4.6 Conclusion

La reconnaissance faciale comme étant une technique biométrique faite l'objectif de ce chapitre, ainsi que les différentes difficultés de cette méthode et quelques domaines de son application. Sans oublier à mentionner à la méthode de description des visages qui est la description par Matrice de co-occurrence. à la fin de ce chapitre, on a appliqué les réseaux des neurones bayésiens dans la reconnaissance des visages.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Les réseaux de neurones artificiels est une technologie alternative pour simuler des complexes problèmes. On les utilisent dans divers applications, à savoir : la reconnaissance des visages, la reconnaissance de parole, traitement d'image et reconnaissance des locuteurs..

Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste, en particulier bayésien. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes permettant de créer des classifications rapides, et d'autre part dans la famille des méthodes de l'intelligence artificielle auxquelles ils fournissent un mécanisme perceptif indépendant des idées propres de l'implémenteur, et fournissant des informations d'entrée au raisonnement logique formel.

L'utilisation du théorème de Bayes permet de proposer une formulation bayésienne de l'apprentissage, qui est très générale, et qui peut s'appliquer à la recherche de tous types de modèles pour peu que ceux-ci soient représentés par un ensemble de paramètres sur lesquels on peut faire une hypothèse de distribution. Nous avons décrit dans ce travail le principe de façon intuitive, et nous avons montré ensuite l'application dans le cadre des réseaux de neurones.

Ce travail nous conduit à développer des perspectives très intéressantes qui s'inscrivent dans la continuité de nos recherches, dans un premier temps, nous souhaitons appliquer les réseaux de neurones bayésiens pour le maximum possible des problèmes de la vie réelle, surtout le problème de reconnaissance des caractères manuscrites Amazighs et le problème de reconnaissance de parole. Dans un deuxième temps, nous souhaitons améliorer l'algorithme de réseaux de neurones bayésiens et intégrer d'autre méthodes bayésiennes dans l'apprentissage des réseaux de neurones en vue de faire une comparaison entre ces méthodes. En fin nous souhaitons améliorer le module d'extraction automatique de visage et de ses caractéristiques.

BIBLIOGRAPHIE

[1] Claude Touzet, “les réseaux de neurones artificiels : introduction au connexionnisme”, juillet 1992.

[2] Youcef DJERIRI. Réseaux de nreurones artificiels. 2017.

[3] Frank Rosenblatt. the perceptron : a probabilistic model. for information storage and organisation in the brain. in : psychological review 65.6(1958) page 386.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors .In :*nature* 323.6088(1986), *p*.533.

[5] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In Advances in neural information processing systems, pages $950 - 957$, 1992.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforwarc networks are universal approximators". In : Neural networks 2.5(1989), *pp*.359366

[7] Kurt Hornik et al. "Degree of approximation results for feedforward network approximating unknown mappings and their derivatives. In : Neural Computation 6.6(1994), pp. $1262 - 1275$

[8] László Györfi and I Vajda. Asymptotic distributions for goodness-of-fit statistics in asequence of multinomial models. Statistics and probability letters, $56(1) : 57 - 67, 2002$.

[9] Thomas P Vogl, JK Mangis, AK Rigler, WT Zink, and DL Alkon. Accelerating the convergence of the back-propagation method. Biological cybernetics, $59(4 - 5) : 257 - 263, 1988$.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. The Journal of

Machine Learning Research, 15(1) : $1929 - 1958, 2014$.

[11] Seán Mc Loone and George Irwin. Improving neural network training solutions using regularisation. Neurocomputing, $37(1-4) : 71 - 90, 2001$.

[12] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In Advances in neural information processing systems, pages $950 - 957, 1992$.

[13] David JC MacKay. Bayesian interpolation. Neural computation, $4(3) : 415 - 447, 1992$.

[14] Aki Vehtari and Jouko Lampinen. "Bayesian MLP neural networks for image analysis". In : Pattern Recognition Letters $21.13 - 14(2000)$, pp. $1183 - 1191$.

[15] R.M. Neal, "Bayesian learning for neural networks", Lecture Notes in Statistics. New York : Springer ; 1996.

[16] Nicholas Métropolis et al. "Equation of state calculations by fast computing machines".bIn : The journal of chemical physics $21.6(1953)$, pp. $1087 - 1092$.

[17] W Keith Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In : (1970).

[18] Stuart Geman and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". In : IEEE Transactions on pattern analysis and machine intelligence $6(1984)$, pp.$721 - 741$.

[19] Isabel Beichl and Francis Sullivan. “The metropolis algorithm”.In : Computing in Science et Engineering $2.1(2000), pp.65\breve{}69$.

[20] Martin Dyer, Alan Frieze, and Ravi Kannan. “A random polynomial-time algorithm for approximating the volume of convex bodies”. In : Journal of the ACM(JACM) $38.1(1991), pp.1\breve{}17$.

[21] Mark Jerrum and Alistair Sinclair. “The Markov chain Monte Carlo method : an approach to approximate counting and integration”.In : Approximation algorithms for NP-hard problems $(1996), pp.482\breve{}520$.

[22] D.J.C. MacKay, "A practical Bayesian framework for back-propagation networks", Neural Computation, $vol.4, pp.448 - 472, 1992$.

[23] D.J.C.MacKay, " Bayesian interpolation", Neural Computation, $vol.4, pp.415 - 447, 1992$.

[24] D.J.C. MacKay, "The evidence framework applied to classification networks", Neural Computation, $Vol.5, pp.720 - 736, 1992$.

[25] Jean-Michel Marin and Christian P Robert."Les bases de la statistique bayésienne". In : *Techniques de l'ingénieur* (2009), pp. $1 - 25$.

[26] H. Ramchoun, M.A.Janati Idrissi, Y.Ghanou and M.Ettaouil, "Multilayer Perceptron : Architecture Optimization and Training", International Journal of Interactive Multimedia and Artificial Intelligence, Vol.4, pp.$26 - 30, 2016$.

[27] M. ETTAOUIL and Y.GHANOU, "Neural architectures optimization and Genetic algorithms", Wseas Transactions On Computer, Vol.8, pp.$526 - 537, 2009$.

[28] M. ETTAOUIL, M.LAZAAR and Y.GHANOU, "Architecture optimization model for the multilayer perceptron and clustering", Journal of Theoretical and Applied Information Technology, Vol.47,pp.$64 - 72, 2013$.

[29] Z. EN-NAIMANI and M.ETTAOUIL, "Architecture Optimization Model of Probabilistic Neural Network", International Journal of Computer Science Issues, V.13, pp.$1 - 9$, 2016.

[30] Z. EN-NAIMANIM. LAZAAR and M.ETTAOUIL, "Architecture Optimization Model for the Probabilistic Self-Organizing Maps and Speech Compression", International Journal of Computational Intelligence and Applications, V.15, 2016.

[31] Stuart Geman and Donald Geman."Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". In : IEEE Transactions on pattern analysis and machine intelligence 6 (1984), pp.721˘741.

[32] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images. In Readings in computer vision, pages 564-584. Elsevier, 1987.

[33] Nedjem. Eddine. Ayat,"sélection de modèle automatique des machines à vecteurs de suports : application à la reconnaissance d'images de chiffres manuscrits", thèse doctorat en génie. P. H. D. Université du Québec, Montréal, Canada, 2004.

[34] John von Neumann and Stanislav Ulam. Monte carlo method. National Bureau of Standards Applied Mathematics Series, 12(1951) : $36, 1951$.

[35] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In Readings in computer vision, pages $564 - 584$. Elsevier, 1987.

[36] M. Chassé. " La biométrie au Québec : Les enjeux ". Commission d'accès à l'information au Québec, Juillet 2002.

[37] A. Ouamane. " Reconnaissance Biométrique par Fusion Multimodale du Visage 2D et 3D". Thèse de doctorat en sciences en Electronique, Université de Mohamed Khider Biskra, 11 Juin 2015.

[38] S. G. Ababsa. "Authentification d'individus par reconnaissance de caractéristiques biométriques liées aux visages $2D/3D$ ". Thèse de doctorat, Université Evry Val d'Essonne. 03 octobre 2008.
'

[39] Anouar Mellakh. "Reconnaissance des visages en conditions dégradées". Thèse de doctorat préparée au Département Électronique et Physique de l'Institut National des Télécommunications dans le cadre de l'École Doctorale SITEVERY en co-accrédidation avec l'université d'Evry-Val d'Essonne. 07 Avril 2009.

[40] H. Bredin "Vérification biométrique d'identité basée sur les visages parlantes. Apport de la mesure de synchronie audiovisuelle face aux tentatives d'imposture élaborées." Juin 2007.

[41] Gonzalez, Rafael C « Digital image processing » volume2 1987.

[42] Haralick, R. M. Shanmugam, K. Dinstein, I. (1973). Textural Features for Image Classification. IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6), 610–621.

[43] S.W. Zucker and D. Terzopoulos. "Finding structure cooccurrence matrices for texture analysis". Computer Vision Graphics and Image Processing. Vol.12, pp. 286-308. 1980.

[44] R.M. Haralick. "Statistical and structural approachs to texture". IEEE. Vol.67.

[45] Anis Chaari. Nouvelle approche d'identification dans les bases de données biométriques basée sur une classification non supervisée. Modélisation et simulation. Université d'Evry-Val d'Essonne, 2009. Français. tel-00549395.