MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET DE L'INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de fin d'études

Présenté en vue de l'obtention
du Diplôme d'Ingénieur d'Etat en Electronique

Option : Communication - Instrumentation

Thème :

Application de l'algorithme de rétro propagation neuronale et du filtre de Kalman à la segmentation d'images

Proposé et dirigé par :

M^lle^ AIT OUAZZOU H.

Présenté par :

M^elle^ RAHMANI Sadia
M^elle^ OULD AREZKI Samira

Année universitaire 2008/2009

Soutenu le : 06/10/2009

Remerciements

Nous tenons à remercier dans un premier temps le bon DIEU qui nous a accordé la force et le courage durant toutes nos études , et en deuxième lieu nos profonds remerciements s'adressent à notre promotrice M^{lle} Hayet AIT OUAZZOU pour sa présence, sa disponibilité et sa précieuse aide qu'elle nous a apporté durant toute la réalisation de ce travail.

Nos remerciements s'adressent aux Messieurs /Mesdames, les membres du jury pour avoir accepté de juger ce modeste travail.

Enfin nous exprimons notre gratitude à toute personne ayant contribué de près ou de loin à l'aboutissement de ce travail (M^{elle} HADJ SAID Malika , M^{r} LAZRI Mourad et M^{r} MAIDI), ainsi les personnes qui travaillent au service des thèses.

Sommaire

Chapitre IV ***Résultats de simulation***

Liste des figures

Liste des tables

Introduction

Les progrès des moyens informatiques et des techniques de traitement ont permis d'ouvrir une voie de développement très prometteuse vers le traitement d'images.

Le traitement d'images suscite un intérêt de plus en plus croissant à mesure que l'image s'impose comme un support et une source d'information privilégiée.

Il est né de l'idée et de la nécessité de remplacer l'observateur humain par une machine, et possède l'aspect multidisciplinaire. On le trouve donc dans des domaines très variés tel que les télécommunications, (TV, vidéo….), la médecine (radiographie, ultrasons…), la météorologie, l'armement (applications militaires).

En traitement d'image, la segmentation est une étape très importante dans l'analyse des images .En effet, elle a pour objectif l'extraction des éléments pertinents et permet également la description de l'information contenue dans celle-ci, en donnant une représentation plus condensée et facilement exploitable.

À cet effet, de nombreuses méthodes de segmentation ont été proposés durant les dernières décennies reposantes sur les différentes approches, contour, région.

Le choix de celle-ci est lié :

- A la nature de l'image.
- Aux opérations situées en aval de segmentation (reconnaissance de forme, interprétation).
- A la contrainte d'exploitation (complexité de l'algorithme, la taille de la mémoire disponible).

Ce mémoire comporte quatre chapitres. Le premier chapitre introductif rappelle brièvement les notions de bases de la segmentation d'image et ces différents types.

Le second chapitre porte sur l'algorithme de la rétro propagation neuronale du gradient de l'erreur, dont on étudie les réseaux de neurones artificiels qui sont une nouvelle technique de traitement de l'information. Ils se traduisent par des algorithmes qui mettent en jeu des concepts associés à la nature du cerveau dont la notion d'apprentissage. Ces réseaux sont utilisés dans diverses applications comme la compression d'image et des données, la reconnaissance de la parole, l'estimation …etc.

L'un des outils les plus utilisés actuellement dans les recherches est le filtre de Kalman , et pour cela on consacre le troisième chapitre a étudié ce filtre d'une manière générale. Et comme le filtre de Kalman utilise une forme modifiée de l'algorithme rétro propagation pour réduire l'erreur quadratique moyenne entre la sortie désirée et la sortie actuelle , alors on l'utilise comme un moyenne de correction.

Le dernier chapitre est consacré à la présentation de quelques résultats obtenus en simulant avec le logiciel Matlab.

Enfin, nous terminons par une conclusion, tout en envisageant des perspectives éventuelles à notre travail.

I.1 Introduction

La segmentation joue un rôle prépondérant dans le traitement d'image. Elle est réalisée avant les étapes d'analyse et de prise de décision dans plusieurs processus d'analyse d'image, tel que la détection des objets. Elle aide à localiser et à délimiter les entités présentes dans l'image.

De nombreuses méthodes de segmentation d'image en niveau du gris existent dans la littérature et la plupart de ces méthodes sont basées sur l'une des propriétés fondamentales : la discontinuité(approche par contour) et la similarité (approche par région). Il est évident que ces deux approches de la segmentation sont duelles, cependant l'information qu'elles mettent en évidence est différente. Les contours possèdent l'essentiel des caractéristiques de formes de la région, par contre la segmentation en région privilégiée les caractéristiques non géométriques liées au critère de segmentation, et s'intéresse au contenu de la région plus qu'à sa forme.

Ce chapitre est consacré à la description des différentes approches de la segmentation d'image en niveau du gris et l'image couleur de type région.

I.2 Définition [1]

La segmentation est une opération de traitement d'image de bas niveau qui vise séparer une image en régions dont les pixels présentent des caractéristiques semblables. Dans le cas de la segmentation d'une image couleur, les caractéristiques utilisées sont les composantes colorimétriques des pixels (à savoir par exemple , R ,G et B si l'image est codée dans le système RGB) .

I.3 Techniques de la segmentation

Dans le but d'avoir une meilleure segmentation d'une image, de nombreuse méthodes ont été proposées suivant les paramètres caractérisant cette image. La différence entre ces méthodes réside dans leurs façons de classer les différents pixels de l'image.

I.3.1 Approche par classification des pixels

La méthode de classification ne prend pas en compte la disposition spatiale des pixel et ne considèrent que la distribution des niveaux du gris ou couleurs dans l'espace de représentation utilisé. Elles identifient les classes de pixels en présence dans l'image et affectent à chaque pixel une étiquette indiquant la classe à laquelle il appartient. La formation des régions n'est obtenue qu'après l'analyse de la connexité des pixels appartenant à la même classe.

Les méthodes ne prenant en compte qu'un seul attribut (le niveau de gris on une seule composante colorimétrique), sont qualifiées des méthodes monodimensionnelles, par contre, les méthodes exploitant plusieurs attributs (couleur) sont qualifiées de multidimensionnelles.

I.3.1.1 Méthodes monodimensionnelles

Ces méthodes reposent sur l'exploitation de l'histogramme, qui caractérise la distribution des niveaux du gris.

Les méthodes monodimensionnelles déterminent des seuils qui constitueront les limites des différentes classes, ces seuils peuvent être pour toute l'image (seuils globaux). Dans ce cas, ils sont déterminés à partir de l'histogramme de l'image complète.

Dans une manière précise, la segmentation par méthode de classification monodimensionnelle se réalise en trois étapes :

- Identifications des seuils interclassent.
- Affectation des points aux différentes classes.
- Extraction des composantes connexes de chaque classe.

I.3.1.2 Méthodes multidimensionnelles

L'approche la plus classique pour construire des classes de pixels sans connaitre leur nombre à priori est fondée sur l'estimation des densités de probabilité des couleurs. Comme l'information couleur est tridimensionnelle, l'outil d'estimation qui semble le mieux adapté est l'histogramme 3D.

Nous ne considérons qu'un histogramme multidimensionnel soit composé de cellules. Chaque cellule d'un histogramme multidimensionnel est repérée par des composantes colorimétriques. Elle contient le nombre de pixels dans les images qui ont des composantes colorimétriques égales à ces coordonnées. Comme la couleur d'un pixel est codée sur 256^3

valeurs, un histogramme 3D d'une image couleur occupe alors une place mémoire très importante. C'est pour cette raison qu'en pratique, la couleur de pixel est requantifiée avec un nombre de valeurs (souvent 32 par composante).

La requantification de la couleur des pixels consiste à effectuer une première segmentation . Si cette requantification ne fournit pas une représentation assez fidèle de l'image initiale, l'analyse de l'histogramme multidimensionnel correspondant ne donne lieu alors à une mauvaise segmentation.

La méthode des K_moyennes (K_means)

Cette méthode consiste a diviser l'espace en sous espaces homogène selon un critère de vraisemblance entre les pixels, sans connaissance apriori de l'image, a part le nombre de classes !; Elles sont basées sur la notion de la distance telle que la règle de décision pour l'affectation d'un pixel à une classe est la distance minimale entre la valeur du pixel et le centre d'un nuage de points représentant une classe . Cette proximité spatiale est souvent mesurée par la distance euclidienne. **[4]**

La méthode des *k*-means a été introduite par J. McQueen en 1971 et mise en œuvre sous sa forme actuelle par E. Forgy. De nombreuses variantes se sont succédées depuis afin d'étendre ses capacités de classification (séparations non linéaires) : *Fernel k*-means (*k*-means basée sur des méthodes à noyaux), améliorer ses performances : global-*k*-means *k*-Harmonic means, automatiser le choix du nombre de clusters : Gaussian-means,X-means.**[5]**

La méthode des K_means constitue l'un des algorithmes les plus simples et les plus répondus dans la classification de données multidimensionnelles . Elle considère initialement un centre de chaque classe , chaque pixel est ensuite affecté à la classe la plus proche . Le centre de chaque classe réactualisé à chaque fois qu'un pixel est affecté à sa classe respectif .ce processus est répété jusqu'à ce que les centres ne changent pas. La Figure I.3.3 représente les classes selon les k_means.

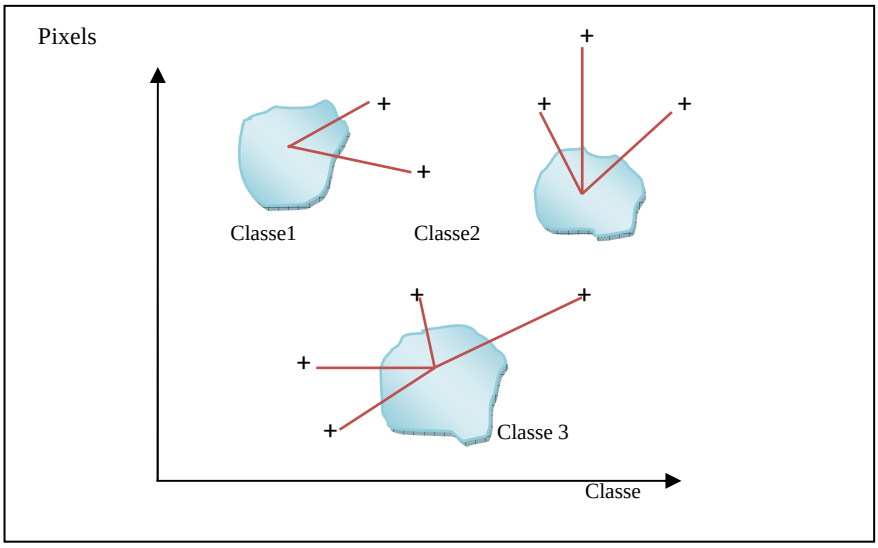


Figure I.3 Représentation des classes selon K_means

On résume l'algorithme des K_means par les étapes suivantes:

- ***Etape 1* :** Fixer le Nre de classes K.
- ***Etape 2* :** Initialiser les centres de classes W_k.
- ***Etape 3* :** Tirer aléatoirement une observation X_n.
- ***Etape 4* :** Calculer les K distances Euclidiennes par rapport aux centres des classes utilisant l'équation suivante :

$$d(X_n, W_j) = ||X_n - W_j||$$

- ***Etape 5* :** Affecter le vecteur paramétrique à la classe qui réalise la plus petite distance.

$$||X_n - W_k|| = \min||X_n - W_k|| \quad j=1\ldots K$$

- ***Etape 6* :** Une fois les classes formées, on calcule les nouveaux centres de classes suivant les affectations effectuées à l'étape précédente.

$$W_j(t+1) = W_j(t) + \frac{1}{\sum_n^{t+1} u_{jn}}[X(t+1)u_{j,t+1} - W_j(t)]$$

Avec $u_{jn}(t)$: est la fonction indiquant l' appartenance d'une observation X_n à la classe C_k

$$u_{jn} = \begin{cases} 1 & \text{Si } X_n \in C_k \\ 0 & \text{Sinon} \end{cases}$$

- ***Etape 7* :** Répéter la procédure jusqu'à ce que la position des centres n'évolue plus.

I.3.2 Approche par région

Les méthodes appartenant à cette famille manipulent directement des régions. Soit elles partent d'une première partition de l'image, qui est ensuite modifiée en divisant ou regroupant des régions, et on parle alors de méthodes de type décomposition /fusion ; soit elles partent de quelques régions, qui sont amenées à croitre par incorporation de pixels jusqu'à ce que toute l'image soit couverte, et on parle alors de méthodes par *croissance de régions*. Des méthodes fondées sur la modélisation statistique conjointe de la régularité et des niveaux de gris de chaque région existent également.

Dans cette partie nous allons essayer de parler des méthodes les plus connues :

- La croissance par région.
- L'approche par division et fusion.

I.3.2.1 Croissance par région

La décomposition de l'image en primitive région est un fait et un pixel peut construire une région .Ces régions sont ensuite regroupées itérativement selon un critère de similarité jusqu'à ce qu'il y ait de fusion possible.

I.3.2.2 Approche par division fusion

Pour ce qui est cette méthode, l'image est prise pour une seule région .Elle sera partagée itérativement en région de plus en plus petite selon un critère d'homogénéité.

I.3.3 La segmentation par contour

Dans le but de reproduire le système visuel humain qui inconsciemment détermine les frontières des objets, le traitement par ordinateur par le biais de la segmentation par contour tente lui aussi de déterminer les formes d'une image.

I.3.3.1 Définition [2]

L'ensemble des points séparant les régions d'une image entre elles appelé contour d'une image désignent les changements importants de la fonction du niveau de gris de celle-ci.

I.3.3.2 Méthodes de détection de contour

Aussi nombreuses qu'elles puissent être, les méthodes de détection de contour peuvent être classées en trois grandes catégories : les méthodes dérivatives, surfaciques, et morphologiques.

I.3.3.2.1 Méthodes surfaciques

L'image de niveau de gris est considérée comme une surface et la transition entre deux régions est modélisée par un gabarit utilisable à deux fins :

- Le contour est présent quand la mise en correspondance entre le gabarit et une zone de l'image est bonne.
- L'approximation par des facettes de la surface fournit une équation analytique locale qui permet de calculer de manière très précise la position du contour et ses caractéristiques.

I.3.3.2.2 Méthodes morphologiques

La morphologie mathématique est applicable à une image numérique avec l'hypothèse que cette dernière soit un espace de représentation d'objet ayant des propriétés ensemblistes.

Par sa définition, qui signifie étude des formes, la morphologie mathématique transformera l'image par le biais d'un objet de formes connues. Cet objet est appelé «élément structurant».

I .3.4 Approche par seuillage

Le seuillage est une des plus veilles et les plus simples techniques de segmentation. Il consiste à diviser l'image en niveaux de gris en deux régions homogènes qui sont le fond et l'objet. Son intérêt apparait en reconnaissance de forme car il permet de calculer des paramètres tels que la surface, le périmètre, le centre de gravité,… ; d'un objet.

I.3.4.1 Définition

Le seuillage est partitionnement de l'image en de classes par le choix d'une valeur S dite seuil. Cette dernière classera chaque pixel comme pixel de fond ou pixel d'objet par la méthode suivante :

- F(x,y)> S alors f(x,y) appartient aux fond.
- Sinon f(x,y) est un objet.

La détermination de S a l'origine de différents types de seuillage.

Ainsi :

- Si S ne dépond que de f(x, y), le seuillage est globale.
- Si S dépond de f(x, y) et de p, on parle de seuillage local.
- Si S dépond de f(x, y) et de p et de (x, y), on parle de seuillage dynamique.

Avec p qui est une propriété locale.

I.5 Conclusion

La segmentation est un élément de grande importance en traitement d'image. Pour parvenir à ce résultat plusieurs méthodes ont été développées. La diversité de ces méthodes est due au fait qu'en segmentation le choix d'une méthode par rapport a une autre suit des caractéristiques particulières du problème considéré, mais toutes ces méthodes peuvent introduire des erreurs qui s'accumulent et détériorent la qualité de la décision finale.

Nous introduisons dans le chapitre suivant les réseaux de neurones , algorithmes d'apprentissage du perceptron multicouches et l'étude expérimentale nous permettra de mieux connaître les limites et les capacités de chaque algorithme ainsi que les performances de notre classificateur, aussi nous allons représenter le protocole d'apprentissage supervisé ainsi les équations des deux algorithmes utilisés dans notre étude.

II.1 Introduction

Le cerveau est sans doute la partie la plus complexe et la plus mystérieuse du corps humain, jusqu'à présent , les plus avides des chercheurs n'ont pas encore expliqué parfaitement sont fonctionnement .

L'étude du cerveau humain a montré qu'il se compose de milliard d'unités appelées neurones, celles-ci fortement interconnectées entres elles , forment une architecture orientée appelée réseau de neurones. Le cerveau est capable d'organiser ces neurones selon un assemblage complexe, de manière a pouvoir réaliser des taches très élaborées, c'est la tentative d'imiter son comportement et ces performances qui a conduit à une modélisation mathématique du neurone ce qui a donner naissance aux réseaux de neurones artificiels.

II.2 Historique [6]

En s'inspirant de leurs travaux sur les neurones biologiques W.Pitts et MC Culloch proposent en 1943 la première modélisation d'un neurone formel.
En 1949, D. Hebb, publie un livre sur l'organisation du comportement et les relations existantes entre la physiologie et la psychologie, on lui doit la règle de Hebb toujours utilisée dans les techniques d'apprentissage des réseaux neuronaux.

En 1958, F.ROSENBLATT commence à travailler sur le perceptron, un des premiers modèles spécifiés de manière suffisamment complexe pour en espérer des résultats intéressants .

En 1960, B.RINDROW , chercheur Américain à Stanford , développe le modèle adaline (adaptative linear element). C e réseau est la base des réseaux multicouches.

En 1969, M.MINSKY et S.PAPERT publient un ouvrage démontrant les limites du perceptron, puis ils étendent ces limitations à tous les modèles du réseau de neurones. Par conséquent, plusieurs recherches dans ce domaine sont abandonnées.

Beaucoup de chercheurs se tournent alors vers d'autres domines plus prometteurs tels que l'intelligence artificielle, toutefois certain d'entre eux persistent et les réseaux de neurones demeurent dans l'ombre jusqu'en 1982, date ou J.J.HOPFIELD, à qui on

doit le renouveau d'intérêt pour les réseaux de neurones, présente un article clair et convaincant développant son modèle de réseau de Hopfield.
En 1985, apparition de l'apprentissage par l'algorithme de rétro propagation pour les réseaux multicouches.

Aujourd'hui , l'étude des réseaux de neurones est une voie prometteuse de l'intelligence artificielle, ils ont des applications dans plusieurs domaines tels que l'industrie, les finances et les télécommunications …..etc.

II.3 Le neurone biologique

II.3.1 Définition

Les neurones sont les cellules de base de système nerveux. Ces cellules nerveuses sont responsables de la réception et la transmission des influx nerveux et forment pour cela de langue fibres reliés entre elles. **[6]**
Elles sont composées d'un corps cellulaire qui contient un noyau d'un axone et d'une ou plusieurs dendrites. Le neurone biologique est représenté par la Figure II.1 :

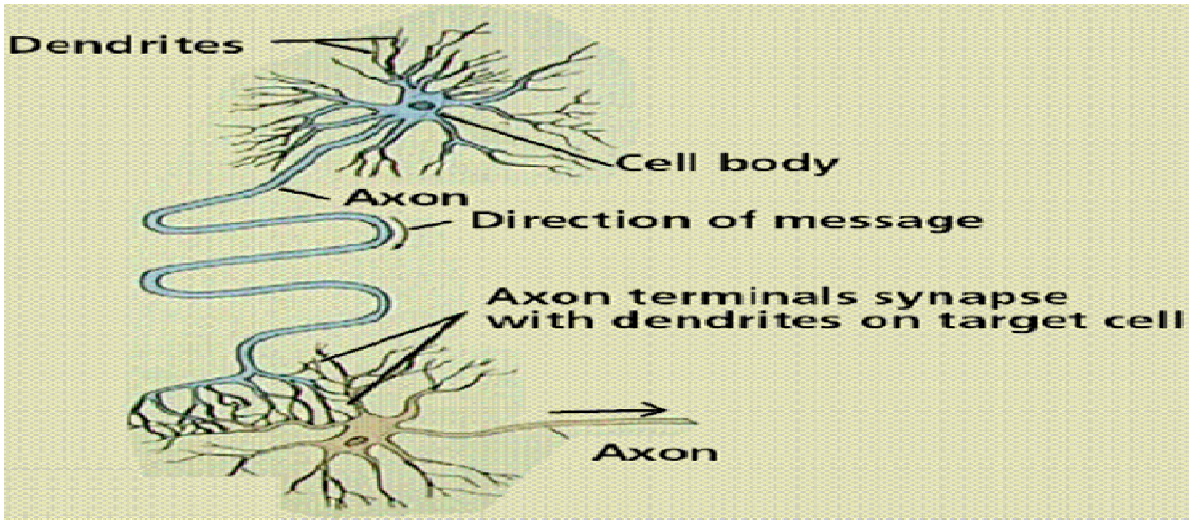


Figure II.1 Le neurone biologique.

II.3.2 Composition de neurone biologique [6]

Un neurone est composé de quatre parties :

- **Le corps cellulaire** : le corps cellulaire contient le noyau du neurone et effectue les transformations biochimiques nécessaires à la vie du neurone.

- **Les dendrites** : sont des petites expansions du neurone qui transportent l'influx nerveux vers le corps cellulaire à partir de la synapse.
- **La synapse** : est une région spécialisée où un signal nerveux saute d'une cellule nerveuse à une autre. C'est le site de communication entre deux cellules nerveuses.
- **Les axones** : sont la longue expansion du neurone qui transporte les flux nerveux du corps cellulaire jusqu'à la synapse.

La connexion entre deux neurones se fait en des endroits appelés synapses. On peut les trouver entre deux dendrites, entre un axone et un corps cellulaire, ou entre deux axones . C'est le lieu où le signal électrique de l'impulsion nerveuse est converti en un signal biochimique.[7]

D'une façon simple, on peut dire que le corps cellulaire traite les courants électriques qui lui proviennent de ces dendrites . Il transmet le courent de ce traitement aux neurones auxquels il est connecté par l'intermédiaire de son axone ; puis effectue une sommation des influx nerveux.

- Si la somme dépasse un seuil, le neurone répond par un potentiel d'action qui se propage le long de son axone.
- Si la sommation est inferieur à ce seuil neurone reste inactif.

II.4 Le neurone formel

II.4.1 Définition [7]

Le neurone formel est un model mathématique s'inspirant du neurone biologique, qui fait une sommation pondérée de ces entiers (chacune de ces entiers est une valeur numérique qui représente l'état du neurone qui l'a émis), puis s'activant la valeur de cette somme. Si cette somme dépasse un certain seuil, le neurone est activé et transmet une repense (sous forme de potentiel d'action) dont la valeur est celle de son activation. Si le neurone n'est pas activé, il ne transmet rien.

Le neurone formé peut être représente une cellule possédant plusieurs entrées et une sortie et peut être modélisé par deux opérations.

- un opérateur de sommation qui élabore un potentiel « post-synaptique »,p est égal à la somme pondérée des entrées de la cellule :

$$\mathbf{P = \sum i(wi * xi)}$$

Avec : w_i est le poids, et x_i est l'entrée (l'état du neurone connecté à l'entrée).

- un opérateur de décision qui calcul l'état de la sortie y du neurone en fonction de son potentiel p.

Cet opérateur est appelé « fonction d'activation » : $y = f(p)$

L'architecture du neurone est représentée par la Figure II.2

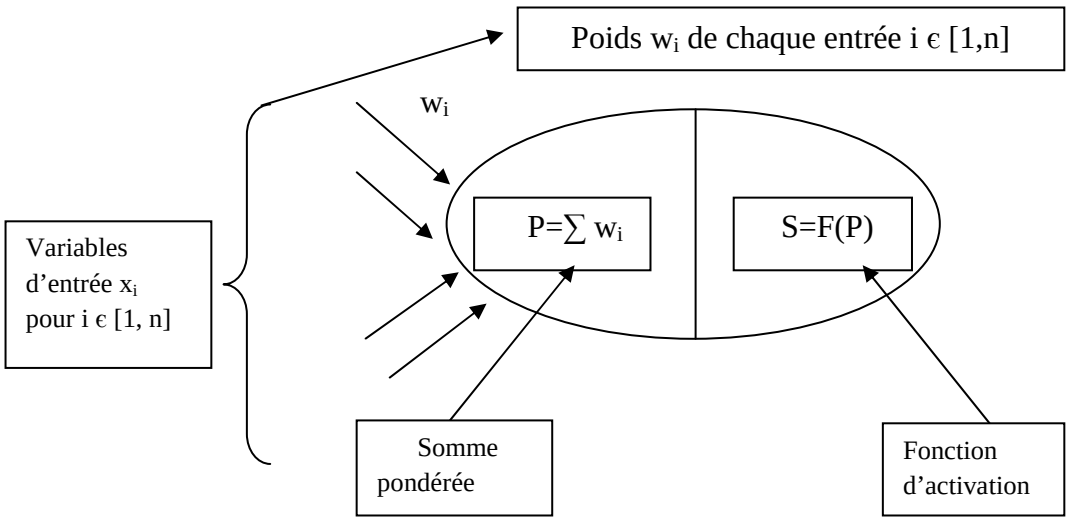


Figure II.4 Architecture du neurone formel.

L'état que peut prendre un neurone est en général binaire mais peut aussi bien prendre une valeur discrète ou bien continue.

- **Transition entre neurone biologique et neurone formel [8]**

Neurone biologique	**Neurone formel**
Synapse	Poids des connexions
Axone	Signal de sortie
Dendrite	Signal d'entrée
Corps cellulaire	Fonction d'activation

II.4.2 La fonction d'activation

La fonction d'activation définit l'état interne du neurone en fonction de son entrée totale. Selon le type de cet état on aura donc différents types pour la fonction d'activation qui sont représentés ci-dessous :

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

Table I I.4.2 ***Table de la fonction***

II.5 Les réseaux de neurones

II.5.1 Définition

Les réseaux de neurones sont composés d'éléments simples (neurones) fonctionnant en parallèle. Ces éléments ont été fortement inspirés par le système nerveux biologique. Comme dans la nature, le fonctionnement du réseau de neurone est fortement influencé par la connexion des éléments entre eux. On peut entrainer un

réseau de neurone pour une tache spécifique (reconnaissance de caractères par exemple), en ajustant les valeurs de connexion (ou poids) entre les éléments (neurones).

L'information transportée par ces connexions est de type numérique ; elle peut être codée de diverses manières ; de plus les liens (connexions) sont ajustables.

La structure d'un réseau de neurones formels à connexions directes de poids w_{ij} de vecteurs d'entrées x_i est illustrée par la Figure II.3.1 :

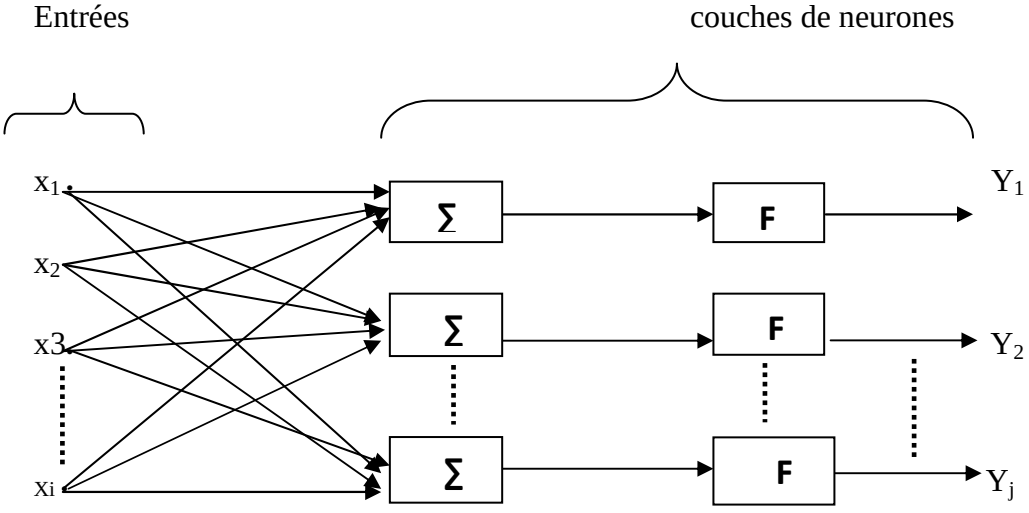


Figure II.5.1 Structure d'un réseau de neurone formel.

Ou : i est le nombre d'element du vecteur d´entree.

j est nombre d'élément du vecteur de la couche.

II.5.2 L'architecture des réseaux de neurones

L'architecture d'un réseau de neurone joue un rôle très important dans son comportement et constitue la connaissance du système, on peut distinguer deux types du réseau :

- **Réseaux non récurrents (a couche**) :

Ce type de réseaux est souvent utilisé pour des problèmes de classification ou d'approximation de fonction non linéaires et complexes. Les réseaux à couches comprennent une couche d'entrée, de sortie et une ou plusieurs couches cachées .

Dans ce réseau, les neurones d'une couche sont reliés par ceux de la couche suivante par des connexions qui sont interdites entre neurones d'une même couche.

- **Réseaux récurrents** :

Ce sont les réseaux qui possèdent des connexions sous forme de boucles. Chaque neurone est connecté a tous les neurones du réseau.

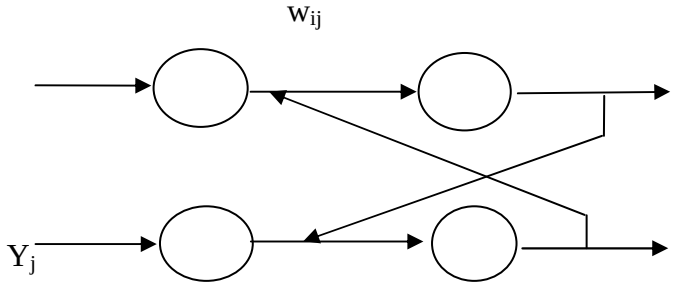


Figure II.5.2 Réseau récurrent

II.5.3 L'apprentissage

La phase d'apprentissage dépond beaucoup plus de la structure du réseau.
L'apprentissage est un processus de simulation continu au cours du quelque paramètres libres du réseau s'adapte a l'environnement de ce réseau.
L'apprentissage se traduit généralement par la modification des poids de connexions.
Selon le degré de contrôle donné a l'utilisateur, on distingue trois modes d'apprentissages :

- ✓ **L'apprentissage supervisé** : c'est le mode d'apprentissage le plus couramment utilisé. Son principe est élémentaire, il consiste à comparer le

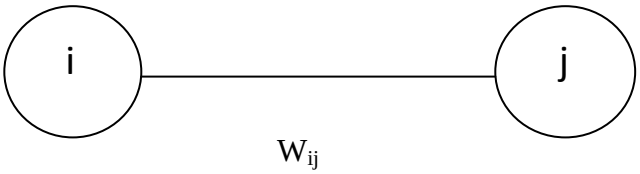
résultat obtenu avec le résultat désiré puis a ajuster les poids de connexions pour minimiser la différence entre les deux .

Dans cet apprentissage l'algorithme le plus répondu est celui de la retro propagation du gradient d'erreur, qui utilise la dérivée de la fonction du transfert des neurones pour calculer l'erreur.

✓ **L'apprentissage non supervisé** : contrairement aux modes supervisés, ce mode utilise un minimum d'informations (seule une base d'entrée est fournie au réseau). Celui-ci doit donc déterminer lui-même ses sorties en fonction des similarités détectés entre les différentes entrées, c.à.d. en fonction d'une règle d'auto-organisation (règle de Hebb).

La règle de Hebb :

La règle de Hebb a donné naissance a de nombreux algorithmes d'apprentissages supervisés ou non supervisés, comme ci -après :



« Si deux cellules sont activées en même temps alors la force de la connexion augmente » **[7].**

La modification de poids dépend de la coactivation des neurones présynoptique et post-synaptique. X_i et X_j sont respectivement les valeurs d'activation des neurones des i et j , ∂w_{ij} (dérivé partielle du poids) correspond à la modification des de poids réalisée

La loi de Hebb peut être modélisée par les équations suivantes (W(t+1) est le nouveau poids , w_{ij} l'ancien) :

$$wij(t+1) = wij(t) + \partial wij$$

$\partial w_{ij} = X_i + X_j$ (la coactivité est modélisée comme le produit des deux valeurs d'activation).

- ✓ **L'apprentissage renforcé (semi supervisé)** : L'apprentissage renforcé fait l'objet de nombreuses études en associations avec les modèles neuromimetiques ou avec les modèles symboliques pour lesquels des méthodes générales ont été mises aux points . Il est utile lorsqu'une information du retour sur la qualité de la performance est fournie.

Les algorithmes d'apprentissage par renforcement essentiellement développé depuis le début des années 80, permettant d'entrainer le réseau pour qu'ils fournissent a chaque stimulus entrant la sortie plus adéquate, cette forme d'apprentissage est de plus en plus utilisée.

II.5.4 Les modèles du réseau de neurones

Il n'est pas possibles d'énumérer l'ensemble des types de réseaux de neurones disponibles à ce jours. Cependant à titre illustratif sont présentés famille parmi les plus populaires. Les chercheurs n'ont cessé que d'inventer de nouveaux types de réseaux toujours mieux adaptés à la recherche de solution de problèmes particuliers.

II.5.4.1 Le modèle des perceptrons multicouches

Le perceptron multicouche est sans doute le plus simple et le plus connu des réseaux de neurones. La structure est relativement simple : une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées. Chaque neurone n'est relié qu'aux neurones des couches précédentes, mais à tous les neurones de la couche précédente. Sa fonction d'activation est de type sigmoïdale en général, son apprentissage est basé sur la rétro propagation du gradient de l'erreur . Les domaines d'application du perceptron sont essentiellement : l'approximation ,la prévision , la classification. Le perceptron multicouche est représenté par la Figure II.5.4.1 :

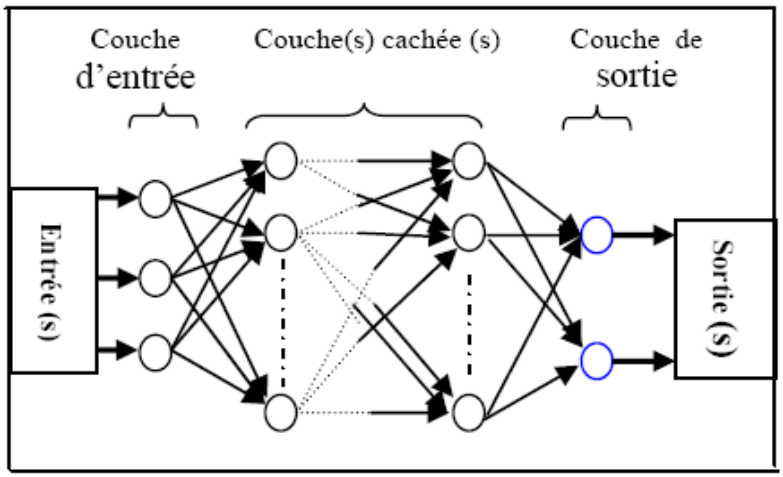


Figure II.5.4.1 Illustration du perceptron multicouche.

II.5.4.2 L'architecture

Un perceptron multicouche est composé de plusieurs couches de neurones et de connexions. Ce nombre est moins égal à deux, signifiant ainsi que le réseau possède deux couches de poids connexionnistes :

- La couche d'entrée : Elle recevra les données source que l'on veut utiliser pour l'analyse.

- La couche cachée : Elle n'a qu'une utilité intrinsèque pour le réseau de neurone, et n'a pas de contacte directe avec l'extérieur.

- La couche de sortie : Elle donne le résultat obtenu après compilation par le réseau des données entrée dans la première couche.

II.5.5 Le modèle de Hopfield

Ce réseau est un réseau récursif, un peu plus complexe que les perceptrons multicouches. Chaque cellule est connectée à toutes les autres et les changements de valeurs de cellules s'enchainent en cascade jusqu'à un état stable. Ce réseau est bien adapté à la reconnaissance de formes.

II.5.6 Le modèle de Kohonen

Le réseau de Kohonen est essentiellement composé d'une couche d'entrée et d'une couche compétitive (ensemble de neurones entièrement connectés dans un plan). La structuration topologique de la carte prévient d'une mesure de distance d définie

sur les neurones du réseau. On défini ainsi un voisinage du neurone. on peut modifier les poids des neurones voisins par la même occasion. Ce modèle est représenté par la Figure II.5.6.

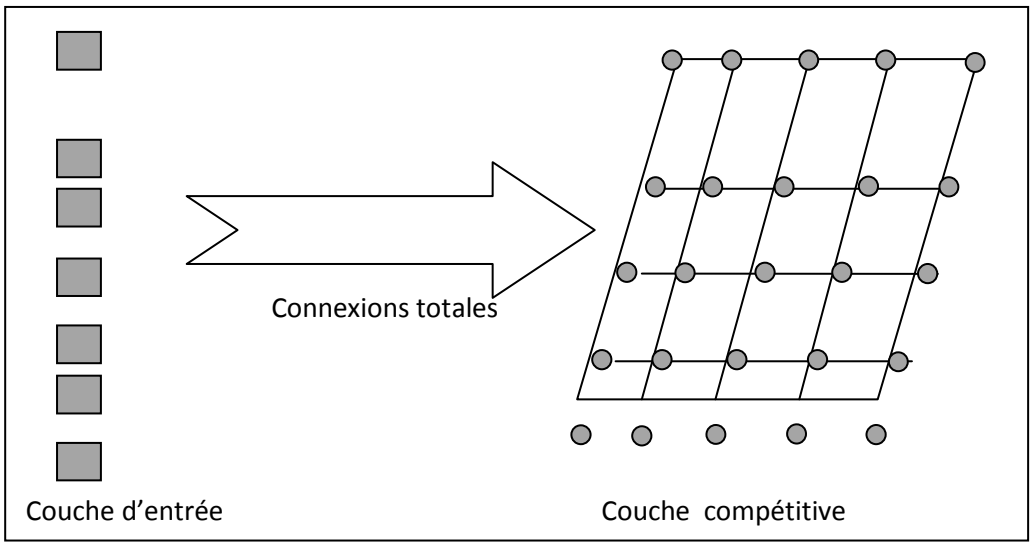


Figure II.5.6.3 Modèle de Kohonen [4]

II.6 Avantages et application des réseaux de neurones [9]

II.6.1 Avantages

L'avantge des réseaux de neurones est qu'ils acceptent des données incomplètes, incertaines ou bruitées. De plus ils sont simples beaucoup et moins de travail personnel que dans l'analyse statique classique. Les réseaux de neurones offrent la capacité de représenter n'importe quelle dépendance fonctionnelle. C'est à dire que le réseau découvre la dépendance lui meme sans avoir besoin qu'on lui « souffle » quoi que ce soit.

II.6.2 Domaines d'application

Les principales applications des réseux de neurones sont diverses, on retrouve essentiellement :

- **La reconnissance des formes :** La reconnaissance des formes est un domaine privilégie d'application pour les réseaux de neurones. Elles désigne les processus qui traitent des informations afin de parvenir à leur classification.

- **Le traitement du signal :** dans **ce** domaine , les applications réalisées traitent essentiellement de la reconnaissance de signatures radar . La société NESTOR a développé un réseau de neurone qui identifie une cible à coup sur (100% de réussite) et reconnait du bruit avec un taux de réussite de 95%/
- **Le contrôle adaptatif :** Ce domaine recouvre entre , la robotique et le contrôle de qualité . On peut citer pour exemple d'application le réseau qui construit par A.BARTO , R.SUTTON , et C.ANDERSON qui contrôle la stabilité d'un balancier sur un petit mobile.
- **La vision et la parole :** Des applications ont été dévloppées aussi bien pour la compéhension de la parole que pour sa synthèse.

Dans le cadre d'une classification, la nécessité de pouvoir évaluer un taux de succès pour qualifier la performance du réseau.

La classification avec les réseaux de neurones se fait en deux étapes :
La première est appelée phase d'apprentissage ; dont on calcule sa sortie et on modifié ces poids, on minimisant l'erreur quadratique entre la sortie désirée et la sortie réelle du réseau , en suivant la règle d'apprentissage.

La deuxième étape est la phase d'exploitation , qu'est la mise en application directe du réseau, on modifiant plus les poids.

L'apprentissage d'un réseau de neurones , est la détermination des poids du réseau au cours du temps.

Les règles d'apprentissage reposent sur la minimisation d'un critère d'erreur quadratique instantanée, durant la phase d'apprentissage sont dit supervisé .

On associe alors à chaque vecteur d'entrée X un vecteur de sortie désirée Y_d et l'apprentissage se déroule comme suite :

a- L'échantillon à apprendre est présenté aux neurones d'entrées, il est propagé vers l'avant à travers le réseau. La réponse des neurones de sortie sera comparée aux réponses désirées et compte tenu des valeurs des poids des connexions, la sortie observée est différente de celle désirée. Cette différence est associée à l'erreur quadratique notée E.

b- La minimisation de cette erreur quadratique permet de vérifier la convergence du réseau. Plusieurs itérations sont nécessaires pour atteindre la stabilité du réseau.

La phase d'apprentissage se termine après plusieurs itérations ainsi le réseau est prêt à être utilisé.

II.7 La mise en œuvre des réseaux de neurones

La mise en œuvre est assez simple.

Nous allons suivre une démarche qu'est composée de quatre étapes principales :

Etape 1 **: Fixer le nombre de couches cachées [10]**

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

Etape 2 : **Déterminer le nombre de neurones par couches cachées [10]**

Chaque neurone supplémentaire permet de prendre en compte des profils spécifiques des neurones d'entrée. Un nombre plus important permet donc de mieux coller aux données présentées mais diminue la capacité de généralisation du réseau. Ici non plus il n'existe pas de règle générale mais des règles empiriques. La taille de la couche cachée doit être :

- Soit égale à celle de la couche d'entrée (Wierenga et Kluytmans, 1994).
- Soit égale à 75% de celle-ci (Venugopal et Baets, 1994).
- Soit égale à la racine carrée du produit du nombre de neurones dans la couche d'entrée et de sortie (Shepard, 1990).

Notons que le dernier choix réduit le nombre de degrés de liberté laisses au réseau, et donc la capacité d'adaptation sur l'échantillon d'apprentissage, au profit d'une plus grande stabilité/capacité de généralisation.

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

Etape 3: **Choisir la fonction d'activation [10]**

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

Etape 4: **Choisir l'apprentissage [10]**

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

II.8 Algorithme de rétro propagation du gradient

L'algorithme de rétro propagation du gradient est certainement à la base des premiers succès des réseaux de neurones . Sa mise en application a permis au domaine du connexion de sortir de la période de silence.

Nous allons représenter cette algorithme qu'est le plus connu pour réaliser l'adaptation des réseaux multicouches. Il s'agit d'une méthode d'apprentissage supervisé ; fondée sur la modification des poids du réseau dans le sens contraire à ces poids.

Nous allons présenter brièvement la méthode d'obtention de ce gradient qui se base sur le calcul des dérivées partielles successives de fonctions composées.

II.8.1 Calcul détaillé

On considère un réseau de neurones recevant des vecteurs à P composantes. Les P entrées, X_k du réseaux sont distribuées sur tous les neurones la sortie du neurone i vaut :

$$Yi = \sigma(pi) = \sigma[\sum_{j=1}^{p} Wij * Xj] \qquad (II.1)$$

Or : σ est la fonction sigmoïde.

$$\sigma = \frac{1}{1+\exp(-x)}$$

On associé au vecteur d'entrée un vecteur de sortie désiré Y_d.

L'erreur quadratique est donnée par l'équation suivante :

$$E = \frac{1}{2}\sum_{j=1}^{n}(Ydj - Yj)^2 \qquad \textit{(II.2)}$$

L'algorithme de rétro propagation consiste à effectuer une descente du gradient sur le critère E. Le gradient est calculé pour tous les poids de manière suivante :

- **1er cas : si le neurone est une unité de sortie** :

Calculons le gradient de cette erreur par rapport à Wik :

$$\frac{\partial E}{\partial W_{ik}} = \sum_{j=1}^{n}\left(Y_{dj} - Y_j\right).\frac{\partial\left(Y_{dj} - Y_j\right)}{\partial W_{ik}}$$

Or: $Yk = \sum(Ydj - Yj)$

$$Yi = \sigma(\sum_{j=1}^{n} wij * xj)$$

$$Xk = \sigma(Yk)$$

$$Pi = \sum_{k=1}^{n} wik * xk$$

$$\frac{\partial E}{\partial wik} = \frac{\partial E}{\partial Yi}\frac{\partial Yi}{\partial pi}\frac{\partial pi}{\partial wik}$$

$$\frac{\partial E}{\partial wik} = (Yi - Yid)\frac{\partial(Yi - Yid)}{\partial Yi}\sigma i`(pi)\,xk$$

$$\frac{\partial E}{\partial W_{ik}} = -\left(Y_{di} - Y_i\right)\frac{\partial Y_i}{\partial W_{ik}} = -\left(Y_{di} - Y_i\right).X_k\sigma'\left(P_i\right)$$

Ou : σ_i` est la dérivée de fonction sigmoïde σ.

Donc : $\frac{\partial E}{\partial wik} = \delta i * xk$ **(*II.3)***

Et la mise à jour du poids, selon le principe du gradient s'écrit :

$$\Delta wik = -a\frac{\partial E}{\partial wik} \quad \textit{(II.4)}$$

$\Delta wik = -a\,\delta i * xk$. Avec a est le gain d'apprentissage positif à la (n^{eme} +1) itération , la règle de modification des poids de la couche de sortie sera :

$$wik(n+1) = wik(n) - a\,\delta i * xk$$

$$\mathbf{wik(n+1) = wik(n) + a(Yid - Yi)\,Yk\,(1 - Yk)\,xk}$$

- **2^{em} cas : si le neurone est unité de couche cachée :**

$$\frac{\partial E}{\partial wkj} = \frac{\partial E}{\partial Yi}\frac{\partial Yi}{\partial xk}\frac{\partial xk}{\partial Yk}\frac{\partial Yk}{\partial wkj}$$

$$\frac{\partial E}{\partial wkj} = (Y_i - Y_{id})\frac{\partial (Yi - Yid)}{\partial Yi} wik\,\sigma i'(pi)\,\sigma_i'(Y_k)X_j$$

$$\frac{\partial E}{\partial wkj} = (Y_i - Y_{id})\,wik\,\sigma_i'(pi)\,\sigma_i'(Y_k)X_j$$

$$\frac{\partial E}{\partial wkj} = (Y_i - Y_{id})\,wik\,Y_k(1 - Y_k)X_k(1-x_k)\,X_j$$

$$\mathbf{\frac{\partial E}{\partial wkj} = (Y_i - Y_{id})\,wik\,X_k\,Y_k(1 - Y_k)\,(1-x_k)\,X_j}$$

Et la mise à jour du poids, selon le principe du gradient s'écrit :

$$\Delta wik = -a\frac{\partial E}{\partial wkj} \qquad (II.5)$$

La (n^{eme} +1) itération , la règle de modification des poids de la couche de sortie sera :

$$\mathbf{wik(n+1) = wik(n) + \alpha(Y_i - Y_{id})\, wik\, X_k\, Y_k(1 - Y_k)(1 - x_k)\, X_j} \quad (II.6)$$

II.8.2 Lissage de la règle d'apprentissage

L'algorithme de rétro propagation présenté précédemment est rarement utilisé tel quel en pratique. En effet, cet algorithme est fondé sur la minimisation d'une erreur instantanée et non pas d'une erreur calculée sur toute la base d'apprentissage. La vitesse de convergence est donc assez lente et nécessite un pas d'adaptation petit pour éviter l'instabilité.

L'amélioration la plus couramment utilisée consiste à ajouter un terme de filtrage sur les incréments d'adaptation. Ce terme est appelé "momentum".

Son adjonction correspond à la minimisation d'un critère E approximativement égal à la somme des erreurs quadratiques pondérer exponentiellement. En effet, soit :

$$E(n) = \frac{1}{2}\sum_{i=1}^{n} \gamma^{(n-i)} . \| Y_d - Y \|^2$$

Et de cette relation on obtient la règle d'apprentissage suivante :

$$W_{jk}(n+1) = W_{jk}(n) + a.\delta_j(n).X_k(n) + \gamma.\Delta W_{jk}(n) \qquad (II.7)$$

II.8.3 L'algorithme

L'algorithme de rétro propagation du gradient se résume finalement aux étapes suivantes :

1. Choix de la taille du réseau :

- Nombre de nœuds cachés,
- Valeurs des poids et seuils du réseau,

2. Choix aléatoire d'une paire **(X,Yd)** dans la base d'apprentissage,

3. Calcule des sorties des différentes couches pour l'entrée X, selon la relation **(II.1)**

4. Mise à jour des poids

- ✓ Mise à jour les poids de la dernière couche en utilisant la relation **(II.4)** et **(II.6).**
- ✓ Mise à jour les poids des couches précédentes selon les relations **(II.5) et (II.7)** .

5. Si le test d'arrêt n'est pas satisfait retourner en **2.**

II.8.4 Un exemple numérique

II.8.4.1 Les conditions initiales

Le pas d'apprentissage : a= 0.1

Le momentum : mom=0.999

Initialisation des poids de la couche d'entrée : W1= (1e-10)

Delw1=0

Initialisation des poids de la couche de sortie : W2= (1e-10)

Delw2=0

II.8.4.2 Les équations utilisées

- L'entrée du neurone de la couche cachée (la somme pondérée) :

$$\mathbf{S1=\sum W_{jk}*}Xj$$

- ✓ La sortie du neurone de la couche cachée (l'activation) :

$$YY1=\sigma(S1) \Longrightarrow YY1=1/1+\exp(-S1)$$

- L'entrée du neurone de la couche de sortie (la somme pondérée)

$$S2=\sum Wik*YY1$$

- ✓ La sortie du neurone de la couche de sortie (l'activation):

$$YY2=\sigma(S2) \Longrightarrow YY1=1/1+\exp(-S2)$$

Calcul des mises à jour des poids:

- Pour les poids de la couche de sortie:

delt2=e_i*YY2 (1-YY2) avec : e_i= YY2 - Y_d

DelW2= delt2*YY1*(-a)

- Pour la couche cachée :

S3=delt2*W2

delt1=S3*YY1 (1-YY1) et YY1 (1-YY1) =dYY1

DelW1= delt1*X*(-a) avec X : la matrice de l'image

Les nouveaux poids pour la couche de sortie (la mise à jour des poids):

- Pour les poids de la couche de sortie :

w2=w2+delw2+ (mom*Delw2)

- Pour les poids de la couche cachée :

W1=W1+delW1+mom*delW1

II.8.4.3 Les résultats numériques

1^{ere} iteration :

S1= 0.0006

YY1=0.5002

S2=(5.068 5.068 5.068)

YY2= (0.9937 0.9937 0.9937)

- Pour la couche de sortie :

Calcul des erreurs:

delW2= (0 ; 0 ; 0)

Calcul des mises à jour :

W2= (10.1327 ; 10.1327; 10.1327)

- Pour la couche cachée :

Calcul des erreurs:

delW1= (0 0 0)

Calcul des mises à jour :

W1= (0.1473 0.1427 0.1549)

Calcul de E :

E= 0.000117520941777

2eme iteration:

S1= 0.0007

YY1=0.5002

S2=(5.4202 5.4202 5.4202)

YY2= (0.9956 0.9956 0.9956)

- Pour la couche de sortie :

Calcul des erreurs:

delW2= (0 ;0 ;0)

Calcul des mises à jour :

W2= (10.8364 ; 10.8364 ; 10.8364)

- Pour la couche cachée :

Calcul des erreurs:

delW1= (0 0 0)

Calcul des mises à jour :

W1= (0.1686 0.1634 0.1763)

3eme itération :

S1= 0.0008

YY1= 0.5002

S2= (5.7707 5.7707 5.7707)

YY2= (0.9969 0.9969 0.9969)

- Pour la couche de sortie :

Calcul des erreurs:

delW2= (0 ; 0 ; 0)

Calcul des mises à jour :

W2= (11.2464; 11.2464 ; 11.2464)

- Pour la couche cachée :

Calcul des erreurs:

delW1= (0 0 0)

Calcul des mises à jour :

W1= (0.191 0.1852 0.1988)

Calcul de E:

E= 0.000117520941777

4eme iteration :

S1= 0.0008

YY1= 0.5002

S2=(5.6254 5.6254 5.6254)

YY2= (0.9964 0.9964 0.9964)

- Pour la couche de sortie :

Calcul des erreurs:

delW2= (0 ;0 ;0)

Calcul des mises à jour :

W2= (11.2464; 11.2464 ; 11.2464)

- Pour la couche cachée :

Calcul des erreurs:

delW1= (0 0 0)

Calcul des mises à jour :

W1= (0.1816 0.176 0.1893)

Calcul de E :

E= 0.000117520941777

II.8.5 Problèmes d'apprentissage

Il est clair qu'un grand nombre de choix s'impose à celui qui voudrait entraîner un réseau de neurones.

L'algorithme de rétro propagation de gradient est très sensible à ces choix. Les problèmes qui peuvent surgir se résument en :

. Minimums locaux

Ce problème concerne la possibilité de converger vers un minimum local dans l'espace des poids.

Une fois que le réseau se stabilise sur un minimum, l'apprentissage s'arrête ainsi.

Si un minimum local est atteint, l'erreur au niveau des sorties du réseau peut être inacceptable.

Si le réseau s'arrête d'apprendre avant l'atteinte d'une solution acceptable, on peut régler ce problème en changeant le nombre de neurones cachés ou bien les paramètres d'apprentissage avec un ensemble diffèrent des poids initiaux.

. Mauvais choix de paramètres

Le comportement du réseau est gouverné par un ensemble de paramètres architecturaux tels que le nombre de neurones cachés, la valeur du pas du gradient, l'initialisation des poids. Un mauvais choix de ces valeurs peut conduire à une non convergence.

II.8.6 Considérations pratiques [8]

L'apprentissage d'un réseau de neurones implique un choix de plusieurs paramètres d'initialisation. Les différentes étapes sont les suivantes :

- Déterminer les données d'apprentissage avec les sorties désirées correspondantes
- Définir la structure du réseau
- Initialiser les poids par des valeurs aléatoires petites car il est impossible d'obtenir des valeurs des poids contenant des connaissances du réseau si tous les poids sont initialisés à des valeurs égales (J.D.Paola et R.A. Schowengerdt, 1994)
- Le taux d'apprentissage (pas du gradient) doit être positif, compris entre 0 et 1.

En effet, on veut une vitesse de convergence suffisante, sans toutefois entraîner des risques d'instabilité de l'algorithme. Plus le pas du gradient est grand, plus les poids changent vite, ce qui permet un apprentissage rapide.
La meilleure valeur du pas est la plus grande que l'on peut atteindre sans qu'il y ait d'oscillations.

- Le momentum doit être compris entre 0 et 1 et très proche de 1.

II.9 Conclusion

Les réseaux de neurones , qui sont une modélisation mathématique du serveau humain, sont souvent utilisés dans les applications d'analyse d'images. Nous retenons , essentiellement dans l'etude des différents modèles de réseaux, l'importance de la phase d'apprentissage qui à pour but de déterminer les poids des connexions et le seuil des neurones.

III .1 Introduction

La méthode classique pour l'apprentissage du perceptron multicouche et l'algorithme de rétro propagation qu'est un algorithme itératif qui minimise l'erreur entre la sortie désirée et la sortie calculée par une entrée particulière du réseau. Malgré son utilisation dans plusieurs domaines en particulier dans la classification d'image , cet algorithme souffre de plusieurs imperfections inhérentes d'une part, à sa convergence , et d'autre part , au choix des paramètres d'initialisations. Pour palier à cette contrainte , nous proposant une modification à cet algorithme en utilisant une méthode issue du traitement du signal ;

L'estimation de l'évolution d'un processus est compliquée dans un environnement bruité : les effecteurs ne sont pas fiables à 100% et les capteurs non plus. Lorsque les erreurs de mesure et d'action sont des bruits blancs dont on est capable d'estimer la covariance, il est possible de « rattraper » les erreurs de manière itérative avec les **filtres de Kalman**.**[10]**

Le Filtre de Kalman est une approche statistique, dont le principe est d'estimer les paramètres d'un modèle (position, vitesse, accélération, température, pression, ...) en combinant les observations avec l'information fournie par le modèle de façon à minimiser l'erreur entre la valeur réelle du paramètre et sa valeur estimée. Autrement dit, ce filtre nous permet d'obtenir, à partir des mesures bruitées (les mesures obtenues via un capteur), une estimation optimale du paramètre inconnu du processus**.[11]**

III.1 Historique [12]

Le **filtre de Kalman** doit son nom à Rudolf Kalman bien que Thorvald Nicolai Thiele et Peter Swerling aient développé un algorithme similaire avant lui. La paternité du filtre fait l'objet d'une petite controverse dans la communauté scientifique. Le filtre a été décrit dans diverses publications par Swerling (1958), Kalman (1960) et Kalman-Bucy (1961).

Stanley Schmidt est reconnu comme ayant réalisé la première implémentation du filtre. C'était lors d'une visite de Rudolf Kalman au NASA Ames Research Center qu'il vit le

potentiel de son filtre pour l'estimation de la trajectoire pour le programme Apollo. Ceci conduisit à l'utilisation du filtre dans l'ordinateur de navigation.

Une grande variété de filtres de Kalman ont été, depuis, développés à partir de la formulation originale dite filtre de Kalman *simple*. Schmidt développa le filtre de Kalman *étendu*, Bierman, Thornton et bien d'autres développèrent toute une gamme de filtres *racine carré*. Le filtre le plus utilisé est vraisemblablement la *phase-locked loop*, largement répandue dans les radios, ordinateurs, équipement de communication, etc.

III.2 Le principe du filtre de Kalman

Le filtrage de Kalman présente l'intérêt d'avoir une formulation faisant appel au concept de variable d'état. La version classique du filtre de Kalman est optimale pour les problèmes linéaires, avec ***une approche temporelle discrète*** , pour l'autre version qu' est le problème non linéaire (processus non-linéaires). **[13]**

III.2.1 L'approche d'état discrète

Un processus (système ou signal) peut être modélisé par sa représentation d'état discrète constituée de deux équations :

Equation d' evolution:

$$X(k+1) = A.X(k) + B.U(k) + W(k) \qquad \textit{(III.1)}$$

Et **:** **L**'équation de mesure :

$$Y(k) = C.X(k) + V(k) \qquad \textit{(III.2)}$$

Tel que :

- **A** et **B** sont les ***matrices du système***.
- **C** : est la ***matrice de mesure***.
- **Y(k)** : est ***l'observation (mesure).***
- **U(k)** : est la ***commande*** (l'entrée du système) à l'instant k.

- **X(k)** : est ***l'état (inconnu)*** du système à l'instant k.

- **W(K) :** est le ***bruit de modélisation*** lié à l'incertitude que l'on a sur le modèle de processus. C'est sans doute le paramètre le plus difficile à quantifier. On le considère blanc, centré **($E[W(k)] = 0$)** et gaussien, de **variance** supposée connue :

$$Q = E[W(k) * W(k)T] \quad .$$

- **V(k) :** est le bruit de mesure également considéré : blanc, gaussien et centré **($E[V(k)] = 0$)** et de **variance** supposée aussi connue

$$(R = E[V(k) * V(k)T]).$$

III.2.2 Hypothèses

Hypo 1 : Les matrices du système A ,B et C supposées connues et constantes ; donc le système est dit stationnaire .

Hypo 2 : W(k) et V(k) se sont des bruits supposés non corrélés entre eux (l'un est indépendant de l'autre) et non corrélés a l'état initial du système. Se sont aussi des bruits blancs, gaussiens et centrés.

Hypo 3 : V(k) est inversible (il y a autant de source de bruits indépendantes que de mesure dans l'équation de mesure)

III.3 Equation du filtre de Kalman

III.3.1 Cycle d'estimation de Kalman

L'estimation optimale consiste à chercher à chaque instant une estimation $\hat{x}(k)$ de l'état $x(k)$ en minimisant un critère qui est la variance de l'erreur d'estimation :

$$\tilde{x}(k) = x(k) - \hat{x}(k) \qquad \text{et} \qquad P(k) = E\left(\tilde{x}(k).\tilde{x}(k)^T\right)$$

On peut séparer le cycle en deux étapes :

- ***Etape de prédiction*** : La phase de prédiction utilise l'état estimé de l'instant précédent pour produire une estimation de l'état courant.

Après avoir exploité la mesure $y(k)$ à l'instant k, nous obtenons une estimation $\hat{x}(k/k)$ de l'état $x(k/k)$, elle donne lieu à une erreur d'estimation $\tilde{x}(k/k)$.

Les équations d'estimation de l'étape de prédiction sont :

$$\hat{x}(k+1/k) = \mathbf{A}\,.\,\hat{x}(k/k) + \mathbf{B}\,.\,\mathbf{U(k)} \quad \textbf{\textit{(Etat prédit)}} \qquad \textit{(III.3)}$$

$$P(k+1/k) = \mathbf{A}\,.\,P(k/k)\,.\,\mathbf{A^T} + \mathbf{Q} \quad \textbf{\textit{(Estimation prédite de la covariance)}}$$

- ***Etape de correction*** **:** les observations de l'instant courant sont utilisées pour corriger l'état prédit dans le but d'obtenir une estimation plus précise.

Nous disposons à l'instant $k+1$, immédiatement avant l'exploitation de la mesure $y(k+1)$ d'une estimation d'état $\hat{x}(k+1/k)$, elle donne lieu à une erreur d'estimation $\tilde{x}(k+1/k)$ de variance $P(k+1/k)$.

Nous utilisons la mesure $y(k+1)$ pour améliorer cette estimation de façon optimale selon les équations de correction suivantes :

$$\hat{x}(k+1/k+1) = \mathrm{A}\,.\,\hat{x}(k+1/k) + \mathrm{K}\,.(\,y(k+1) - \mathrm{C}\,.\,\hat{x}(k+1/k)\,) \qquad \textit{(III.4)}$$

$$P(k+1/k+1) = (\,\mathrm{I} - \mathrm{K}\,.\mathrm{C}\,)\;P(k+1/k) \qquad \textit{(III.5)}$$

$$K = P(k+1/k)\,.\mathrm{C}^{\mathrm{T}}\,\Sigma^{-1}$$

$$\Sigma^{-1} = (\,\mathrm{C}\,.\,P(k+1/k)\,.\mathrm{C}^{\mathrm{T}} + \mathrm{R}\,)^{-1}$$

NB:

La formule de la mise à jour de la covariance est valide uniquement pour un gain de Kalman optimal. L'utilisation d'autres valeurs de gains nécessite des formules plus complexes.

III.4 L'algorithme du filtre de Kalman [11]

Initialement x_0 et p_0 sont supposés gaussiens.

Afin d'obtenir l'état optimal du système, on doit combiner les observations $y(k)$ avec l'information fournie par le modèle $x(k)$.

Les étapes de l'algorithme :

- ✓ Initialisation de l'état du système et de sa matrice de covariance

$$x(0/0) = x_0$$
$$P(0/0) = p_0$$

- ✓ Calcul de l'estimation $\hat{x}(k+1/k)$ de l'état du système à l'instant $k+1$ à partir des mesures disponibles à l'instant k.

$$\hat{x}(k+1/k) = \mathbf{A} \cdot \hat{x}(k/k) + \mathbf{B} \cdot \mathbf{U(k)}$$

- ✓ Mise à jour intermédiaire de la matrice de covariance de l'état en tenant compte de l'évolution prévue par l'équation d'évolution de l'état ;

$$P(k+1/k) = \mathrm{A} \cdot P(k/k) \cdot \mathrm{A}^{\mathrm{T}} + \mathrm{Q}$$

- ✓ Calcul du gain du filtre optimal (qui peut être calculé à priori)

$$\mathrm{K} = P(k+1/k) \cdot \mathrm{C}^{\mathrm{T}} \cdot (\mathrm{C} \cdot P(k+1/k) \cdot \mathrm{C}^{\mathrm{T}} + \mathrm{R})^{-1}$$

- ✓ Mise à jour de la matrice de covariance de l'état

$$P(k+1/k+1) = (\mathrm{I} - \mathrm{K} \cdot \mathrm{C}) \; P(k+1/k).$$

- ✓ Réactualisation de l'estimation de l'état

$$\hat{x}(k+1/k+1) = \mathrm{A} \cdot \hat{x}(k+1/k) + \mathrm{K} \cdot (y(k+1) - \mathrm{C} \cdot \hat{x}(k+1/k))$$

III.5 Domaines d'application [12]

Le **filtre de Kalman** est utilisé dans une large gamme de domaines technologiques (radar, vision électronique, communication ...). C'est un thème majeur de l'**automatique** et du **traitement du signal**. Un exemple d'utilisation peut être la mise à disposition, en continu, d'informations telles que la position ou la vitesse d'un objet à partir d'une série d'observations relative à sa position, incluant éventuellement des erreurs de mesures.

Par exemple, pour le cas des radars où l'on désire suivre une cible, des données sur sa position, sa vitesse et son accélération sont mesurées à chaque instant mais avec énormément de perturbations dues au bruit ou aux erreurs de mesure. Le filtre de Kalman fait appel à la dynamique de la cible qui définit son évolution dans le temps pour obtenir de meilleures données, éliminant ainsi l'effet du bruit. Ces données peuvent être calculées pour l'instant présent (filtrage), dans le passé (lissage), ou sur un horizon futur (prédiction).
Le filtrage de Kalman est aussi de plus en plus utilisé en dehors du domaine du traitement du signal, par exemple en météorologie et en océanographie, pour l'assimilation de données dans un modèle numérique, en finance ou en navigation.

III.6 Le filtrage de Kalman avec les réseaux de neurones

Le filtrage de Kalman utilise une forme modifiée de l'algorithme rétro propagation pour réduire l'erreur quadratique moyenne entre la sortie désirée et la sortie actuelle du réseau.

Les inconvénients de l'algorithme de la rétro propagation sont induits aux choix arbitraire du pas du gradient et le momentum (a,γ) . Cela signifie que la convergence du réseau est fondée uniquement sur l'expérimentation. Une solution à ce problème consisterait a utiliser le filtrage de Kalman qui permet de calculer les poids. Le pas du gradient et le momentum (a,γ) sont remplacés par un paramètre K appelé gain de Kalman. Ce gain est calculé à partir du vecteur entré et du vecteur poids pour chaque couche du réseau. Son avantage provient de sa particularité d'être beaucoup plus rapide et d'être moins sensible au choix des valeurs initiales des poids du réseau contrairement à l'algorithme de la rétro propagation, le filtrage de Kalman consiste à résoudre un système d'équations linéaires pour faire l'adaptation des poids.

III.6.1 Principe

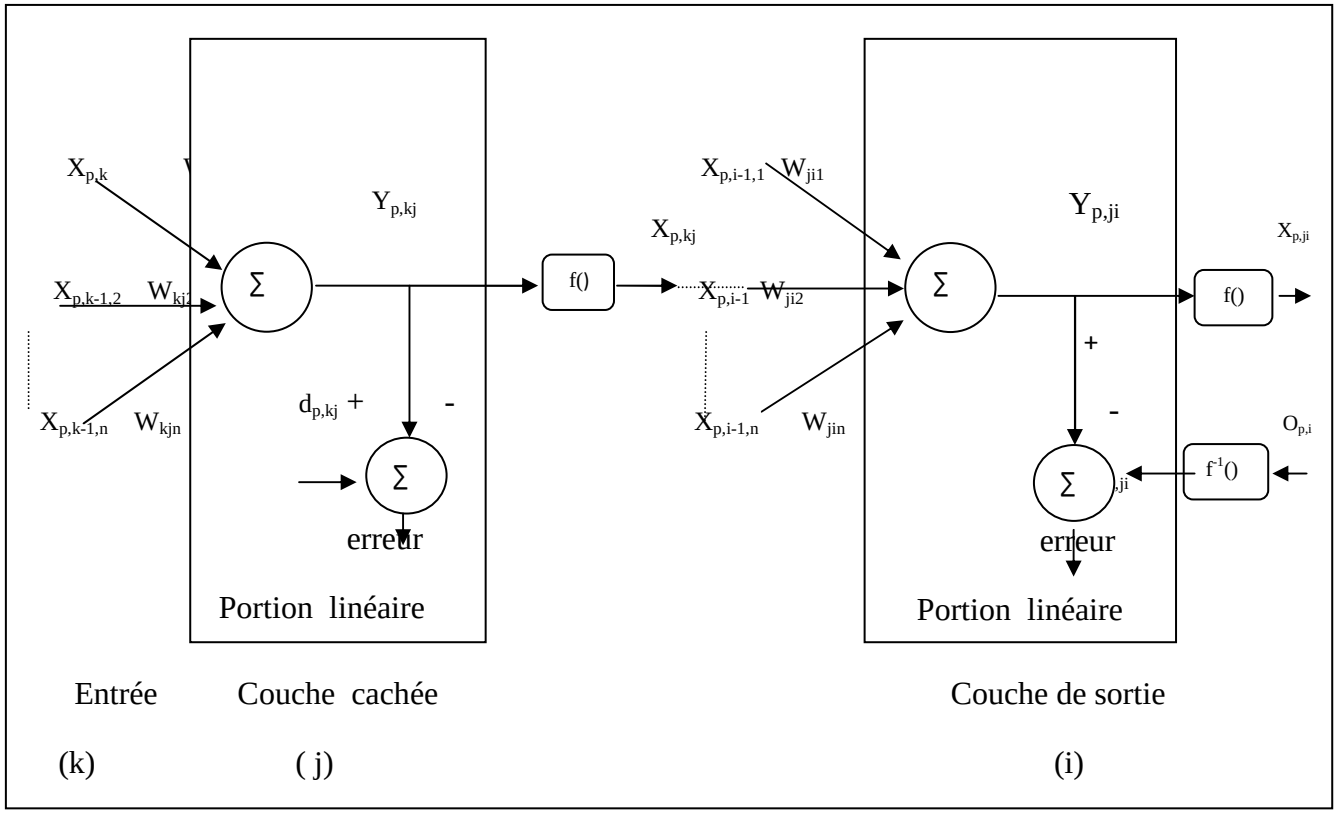


***Figure III.6.1* :** *portion linéaire d'un neurone.*

Si les entrées $X_{p,kj}$ et les sommes pondérées $Y_{p,kj}$ sont connues alors le système devient linéaire, c.à.d c' est un système qui relie le vecteur poids W_{kj} à la somme pondérée des sorties $Y_{p,kj}$ et les neurones d'entrées $X_{p,k-1j}$.(Cas de la Figure III.6.1).

Cependant, au départ de l'apprentissage les entrées et les sorties des neurones cachés ne sont pas connues, elles doivent être estimées. Par ailleurs, ce qui doit être connu à priori sont les entrées des neurones de la première couche, et les sorties des neurones de la dernière couche (la réponse désirée $O_{p,i}$). Par conséquent, les sorties $Y_{p,jk}$ des sommations des neurones de la couche de sortie sont également connues.

On note que la somme des sorties désirées $d_{p,ij}$ pour les neurones de la couches de sortie est toujours connue grâce a la fonction inverse de la réponse désirée

$O_{p\,i}.(\mathbf{d_{p,ij} = f^{-1}(O_{p\,i})})$.

III.6.2 Calcul détaillé

Afin, de minimiser l'erreur on prend la partie linéaire du neurone dans la couche cachée ainsi pour la couche de sortie.

Dans ce qui suit, on prend en considération l 'existence d'un seul neurone. Les indices identifiant les autres neurones seront omis, et seule la partie linéaire du neurone sera considérée dans nos calculs.

Une fois que la somme pondéré Y_p est estimé, on cherche à minimiser l'erreur :

$$E = \frac{1}{2} \sum_{p=1}^{m} (d_p - Y_p)^2 \qquad \textit{(III.6)}$$

- m : étant le nombre d'échantillons de l'apprentissage.
- Y_p : la somme pondérée.
- d_p : la sortie désirée .

La minimisation de cette erreur, se fait en utilisant la méthode des dérivées partielles. On calcule alors la dérivée partielle de E par rapport à chaque poids W_{jk} ; le résultat est obtenu par la résolution des équations suivantes : W_n

$$\frac{\partial E}{\partial W_n} = -\sum_{p=1}^{m} (Y_{dp} - Y_p) \frac{\partial Y_p}{\partial W_n} = 0 \qquad \textit{(III.7)}$$

$$Y_p = \sum_{i=0}^{n} (W_i . X_{pi})$$

Et $$\frac{\partial Y_p}{\partial W_n} = X_{pn}$$

On obtient alors

$$\frac{\partial E}{\partial W_n} = -\sum_{p=0}^{m} (Y_{dp} - \sum_{i=0}^{n} W_i . X_{pi}) . X_{pn} = 0 \qquad \textit{(III.8)}$$

$$\sum Y_{dp} . X_{pn} = \sum_{p=1}^{m} \sum_{i=0}^{n} W_i . X_{pi} . X_{pn}$$

En changeant la sommation par une multiplication de vecteurs :

$$\sum_{p=1}^{m} X_{pn} W^{T} X_{p} = \left[\sum_{p=1}^{m} X_{pn} . X_{p}^{T} . W \right]$$

On définie :

- $$R = \sum_{p=1}^{m} X_{p} . X_{p}^{T} \qquad \textbf{(III.9)}$$

- $$p = \sum_{p=1}^{m} d_{p} X_{p} \qquad \textbf{(III.10)}$$

Ainsi, on écrit : $\mathbf{P} = \mathbf{R}.\mathbf{W}$ ***(III.11)***

- R : matrice de corrélation de la base d'apprentissage.
- P : Le vecteur de corrélation entre l'appris et la sortie désirée correspondante.

Le vecteur poids W est obtenu comme suit :

$$\mathbf{W} = \mathbf{R}^{-1}.\mathbf{P} \qquad \textbf{(III.12)}$$

Durant la phase d'apprentissage, l'équation ***(III.6.7)*** doit être résolue à chaque neurone du réseau pour réajuster les poids.

Nous écrivons :

- $$R(t) = \sum_{k=1}^{t} X(k).X^{T}(k) \qquad \textbf{(III.13)}$$

- $$P(t) = \sum_{k=1}^{t} Y_{d}(k).X(k) \qquad \textbf{(III.14)}$$

Les estimations d'une couche sont basées sur les données issues des estimations de la couche précédente. Lorsque cette dernière subit un mauvais traitement, les estimations de la couche en cours ne seront pas bonnes, et ceci va faire allonger la durée de l'apprentissage. Pour remédier à ce problème, on insère un facteur d'oubli ***b***.

Plus ***b*** est proche de **0** plus le modèle est capable de prévoir des variations brusques, devenant ainsi sensible aux erreurs qui contaminent les données. Si ***b*** est proche de 1 la capacité d'adaptation du modèle est moindre mais il se montre également moins sensible aux données contaminées par des erreurs (appareils défectueux, mesures provenant de sources différentes etc.).

Tel que :

$$R(t) = \sum_{k=1}^{t} b^{(t-k)}.X(k).X^{T}(k)$$

$$P(t) = \sum_{k=1}^{t} b^{(t-k)}.Y_{d}(k).X(k)$$

Le facteur d'oubli va ramener une nouvelle information dont l'importance apparaît quand les estimations de corrélation ont un négligeable effet sur les estimations en cours.

Alors les deux équations deviennent :

$$R(t) = b.R(t-1) + X(t).X^{T}(t) \qquad \textbf{(III.15)}$$

$$P(t) = b.P(t-1) + Y_{d}(t).X(t) \qquad \textbf{(III.16)}$$

Et par conséquent :

$$W(t) = R^{-1}(t).P(t) \qquad \textbf{(III.17)}$$

III.6.3 Filtrage de Kalman

Les équations suivante permettent de calculer les poids par l'utilisation des équations de Kalman , soit le gain de Kalman :

$$K_{k}(t) = \frac{R_{k}^{-1}(t-1).X_{k-1}(t)}{b_{k} + X_{k-1}^{T}(t).R_{k}^{-1}(t-1).X_{k-1}(t)} \qquad \textbf{(III.18)}$$

❖ **L'équation de mise à jour de la matrice inverse** :

$$R^{-1}{}_k(t) = [\, R^{-1}{}_k(t-1) - K_k(t).X^T(k-1)(t).R^{-1}{}_k(t-1)\,].b^{-1}{}_k \qquad \textit{(III.19)}$$

❖ **La mise à jour du vecteur poids de la couche de sortie (i)** est donnée par :

$$W_{ij}(t) = W_{ij}(t-1) + K_i(t).(d_j - Y_{ij}). \qquad \textit{(III.20)}$$

❖ **La mise à jour du vecteur poids dans les couches cachées :**

$$W_{jk}(t) = W_{jk}(t-1) + K_j(t).e_{jk}(t). \qquad \textit{(III.21)}$$

- t : l'itération en cours.
- μ_j : pas d'apprentissage pour la couche j.
- b_j : facteur d'oubli pour la couche j.

III.6.4 Algorithme

1- Initialisation pour toutes les couches :

- Initialiser les entrées $X_{(k-1),0}$ de chaque nœud par une valeur différente de zéro.
- Initialiser les poids du réseau par des valeurs aléatoires.
- Initialiser la matrice inverse R.
- Initialiser la constante sigmoïdale a.

2- Sélectionner les échantillons d'apprentissage en spécifiant les paires (Entrée / Sortie) ou (X_0, O).

3- Propager un échantillon vers la sortie du réseau

- pour chaque couche de k= 1 à i :

Calculer la sortie de la sommation au niveau de chaque neurone tel que :

$$Y_{kj} = \sum_{l=0}^{N} X_{(k-1),l} . W_{kjl} \qquad \textbf{(III.21)}$$

Et calculer la sortie du nœud par :

$$X_{kj} = f(Y_{kj}) = \frac{1-\exp(-aY_{kj})}{1+\exp(-aY_{kj})} \qquad \textbf{(III.22)}$$

Pour chaque neurone k

N = nombre d'entrées.

4- Calculer les équations du filtre de Kalman ; pour chaque couche de k= 1 à i

Calcul du gain de Kalman

$$K_k(t) = \frac{R_k^{-1}(t-1).X_{k-1}(t)}{b_k + X_{k-1}^T(t).R_k^{-1}(t-1).X_{k-1}(t)}$$

et la mise à jour de la matrice inverse :

$$R_k^{-1} = [R_k^{-1} - K_k . X_{k-1}^T . R_k^{-1}]\ b_k^{-1} \qquad \textbf{(III.23)}$$

5- Calculer les erreurs :

- Calculer la dérivée de la fonction non linéaire :

$$f'(Y_{kj}) = \frac{2a \times \exp(-aY_{kj})}{(1+\exp(-aY_{kj}))^2} \qquad \textbf{(III.24)}$$

- Calculer de l'erreur de la couche de sortie (k = i) :

$$E_{ik} = f'(Y_{ik}) . (O_i - X_{ik})$$

- Et pour les couches cachées de k = i-1 à j = 1 :

$$e_{kj} = f'(Y_{kj}).\sum_{l}\left(e_{(k+1),l}.W_{(k+1),l,j}\right) \qquad \textbf{(III.25)}$$

6- Calcul pour chaque neurone de la couche de sortie la sortie désirée en utilisant la fonction inverse par :

$$d_i = (\frac{1}{a}).\ln\left[\frac{1+O_i}{1-O_i}\right] \qquad \textbf{(III.26)}$$

7- Mise à jour des poids :

- pour la couche de sortie :

$$W_{ij} = W_{ij} + K_i.(d_i - Y_{ij}) \qquad \textbf{(III.27)}$$

- Et pour les couches cachées :

$$W_{kj} = W_{kj} + K_k.e_{kj}.\mu_k \quad . \qquad \textbf{(III.28)}$$

8- Test d'arrêt : à ce point on utilise l'erreur quadratique moyenne de la sortie du réseau pour tester la convergence ; ou bien, on fixe un nombre d'itérations. Si le test d'arrêt n'est pas vérifier, on retourne à 2.

III.7 Conclusion

Avec un meilleur choix des paramètres du réseau (nombre de neurones et nombre de couches...), l'algorithme de Kalman présenté ci-dessus est beaucoup plus rapide que celui de la rétro propagation. Nous allons mettre en pratique les applications mises en considération dans ce qui a précédé dans le chapitre suivant. Différents tests seront effectués afin de déterminer les meilleurs paramètres constituant le réseau le plus performant, pour chaque algorithme. Nous comparons alors les deux approches pour essayer de donner une constatations sur les deux dernière.

IV.1 Introduction

Dans ce chapitre, nous présentons quelques résultats de simulation de l'algorithme de rétro propagation du gradient de l'erreur des réseaux de neurones.

Pour illustrer les résultats de segmentation, les résultats seront présentés sous forme de courbe d'évolution de l'erreur quadratique en fonction de nombre d'itérations ou l'on fixe les paramètres du perceptron multicouches neuronal qui sont :

- Le nombre de neurones dans la couche cachée,
- Le pas d'apprentissage : Il sert a faire converger les poids vers une solution, on effectuant des sauts discrets pour les poids ,
- Le momentum : Il sert a réduire le temps d'apprentissage dans le cas où on utilise un grand pas d'apprentissage,
- L'ordre de grandeur des poids synaptiques des neurones,
- Le nombre d'itérations.

Remarque

L'initialisation des poids synaptiques des neurones se fait aléatoirement. Ceci veut dire que pour les mêmes paramètre du réseau, les résultats obtenus seront nettement différents .

IV.2 Les paramètres des classifieurs

Nous utilisons un perceptron à une seule couche cachée. La fonction d'activation est la sigmoïde. La matrice d'entrées (l'image) est de taille 512*512. Le nombre de neurones en sortie est 3 (qui correspond aux 3 classes). Le nombre de neurones dans la couche cachée est déterminé par tâtonnement.

Les performances de classification, sont déterminées par le diagramme de l'erreur quadratique et la matrice de confusion pour la méthode neuronale.

IV.3 Evolution des performances de classification

IV.3.1 Diagramme de l'erreur

C'est une représentation de l'erreur quadratique en fonction du nombre d'itérations, calculée dans la phase d'apprentissage. Le calcul de l'erreur se fait après propagation du vecteur de l'entrée vers la sortie. La sortie obtenue (réelle) est souvent différente de la sortie désirée.

Dans ce cas, cette erreur est rétro propagée, c.à.d. propagée de la sortie vers l'entrée en recalculant les valeurs des poids synaptiques (mise à jour des poids) . Cette procédure est réitérée plusieurs fois (par fois jusqu'à 100000 itérations). A chaque introduction d'un vecteur, l'erreur est calculée et comparée à la valeur précédente (l'itération précédente). Le but est de minimiser cette erreur. Par conséquent, la convergence de l'algorithme est assurée par un nombre très grand d'itérations afin d'éviter un minimum local. Il est évident que dans le cas idéal , nous obtenons une courbe qui décroît rapidement vers une valeur très proche de zéro.

IV.3.2 La matrice de confusion

Elle est de dimension N*N (où N est le nombre de classe de sortie, (égale à 3 dans notre cas). Cette matrice est calculée pour les vecteurs d'apprentissage et pour ceux du test. Son élément m_{ij}, est égal au nombre de vecteurs de la classe i qui sont affectés dans la classe j, comme indiqué en Figure IV.4.2 . idéalement cette matrice ne devait contenir que des « 1 » sur la diagonale et des « 0 » partout ailleurs.

	C1	C2	C3
C1	C1/ C1	C2/C1	C3/C1
C2	C1/C2	C2/C2	C3/C2
C2	C1/C3	C2/C3	C3/C3

Table IV.4 .2 : **Matrice de confusion, N=3**

On définit la probabilité de classification correcte (Pcc) qui est donnée par la moyenne des éléments de la diagonale de la matrice de confusion.

IV.4 Résultats de classification des images pas les K_means

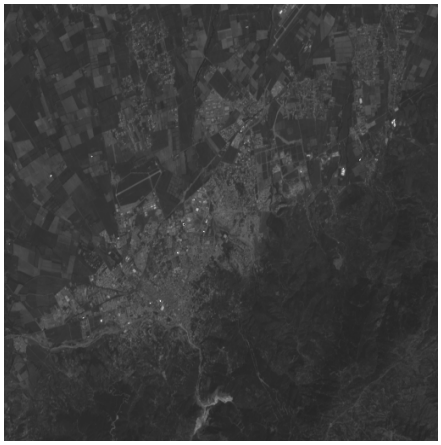
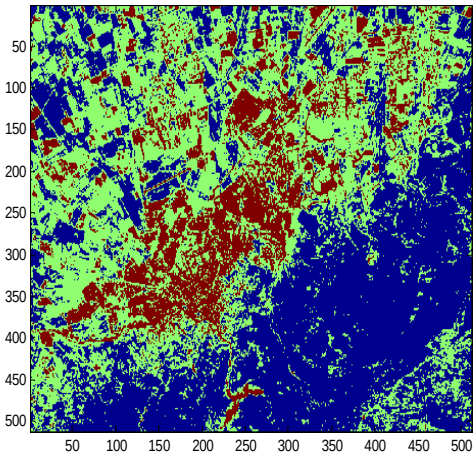
Concernant notre application, nous avons utilisé des images fournies par le satellite SPOT couvrant la région de Blida. Nous nous sommes intéressés à trois classes :

Classe 1 : végétation

Classe 2 : mer

Classe 3 : urbain

Nous nous sommes intéressés à l'apprentissage supervisé. Ceci exige la connaissance à priori de l'appartenance de chaque pixel. Pour ce faire, nous avons appliqué la méthode des k_means pour attribuer la classe d'appartenance aux éléments des trois images. La Figure IV.4 montre le résultat obtenu.

	
L'image originale 1	***L'image segmentée***

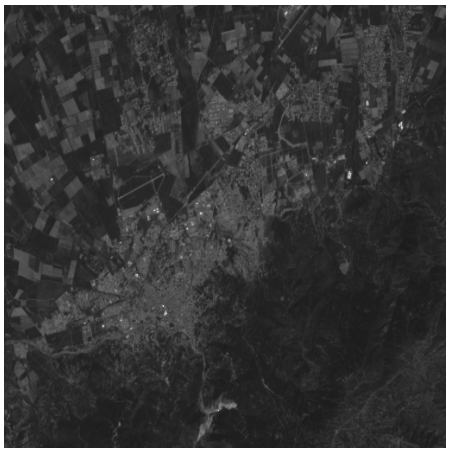
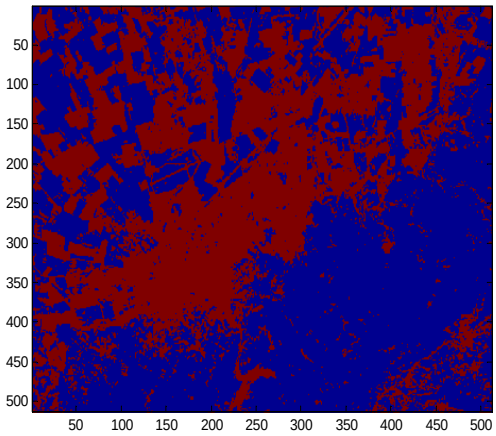
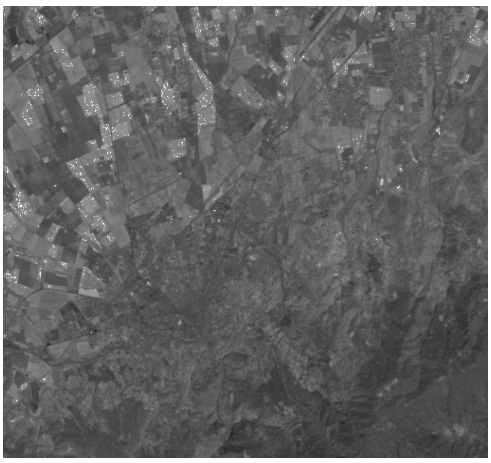
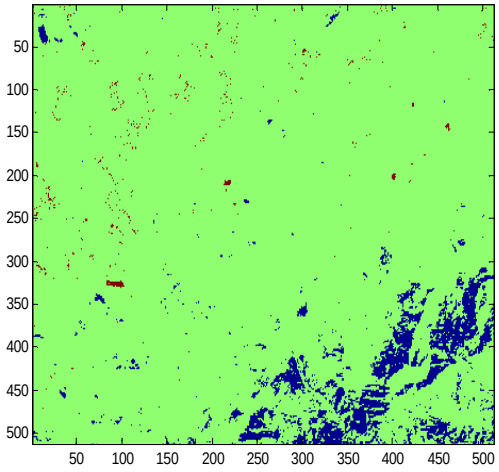
	
L'image originale 2	*L'image segmentée*
	
L'image originale 3	*L'image segmentée*

Figure IV.4 La classification des trois images par les K_means

IV. 5 Présentation de quelques résultats obtenus

Nous présentons dans cette partie les résultats obtenus de l'apprentissage neuronal et ceci pour différentes valeurs des paramètres du réseau. Ainsi pour chaque exécution du programme nous modifions un paramètre et les autres restent inchangés.

IV. 5.1 Influence du nombre d'itérations

Ce test est effectué pour étudier l'influence du nombre d'itérations sur l'erreur commise du réseau. Pour un même réseau, on fixe tous les paramètres et on fait varier seulement le nombre d'itérations. Une itération correspond au passage de toute la base d'apprentissage (les trois images)

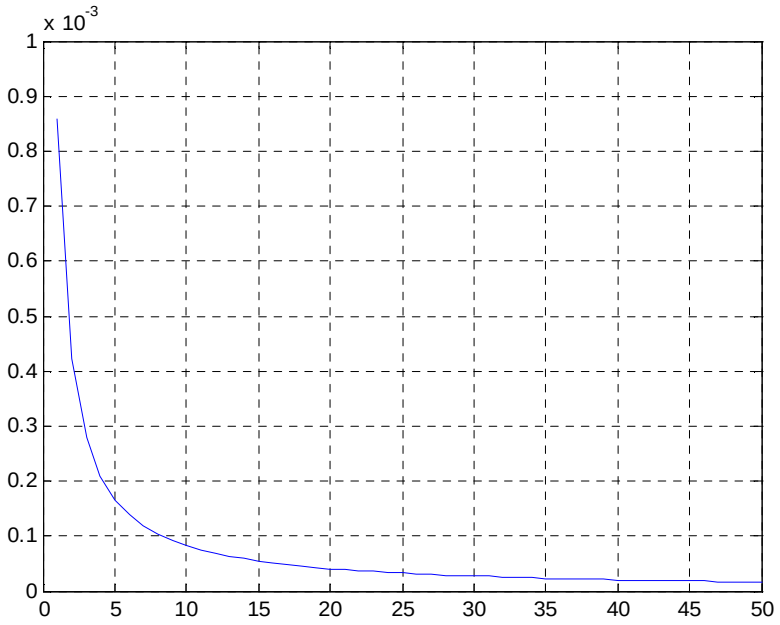


Figure IV .5.1 .1: Représentation de l'erreur quadratique en fonction de nombre d'itérations.
Momentum=0.989, Pas d'apprentissage =0.002, w1=w2=10^-16
Nombre de neurones =1, Nombre d'itération =50, erreur = $1.629*10^{-5}$

1	0	0
1	0	0
1	0	0

Pcc=0.333

Table IV.5.1.2 : Le résultat de la classification avec la matrice

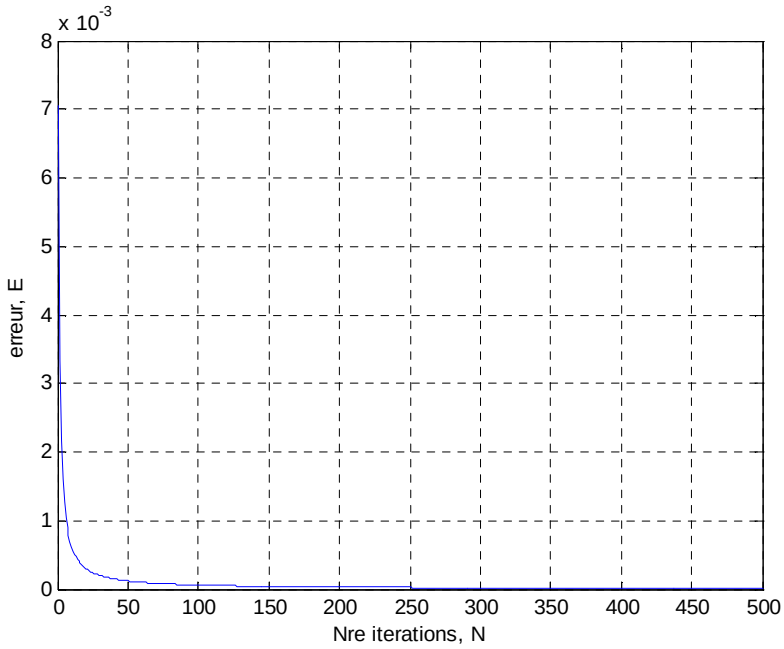


Figure IV .5.1.3: Représentation de l'erreur quadratique en fonction de nombre d'itérations.
Momentum=0.989, Pas d'apprentissage =0.002, w1=w2=10^-16
Nombre de neurones =1, Nombre d'itération =500, erreur = $1.1725*10^{-5}$

1	0	0
1	0	0
1	0	0

Pcc= 0 .333

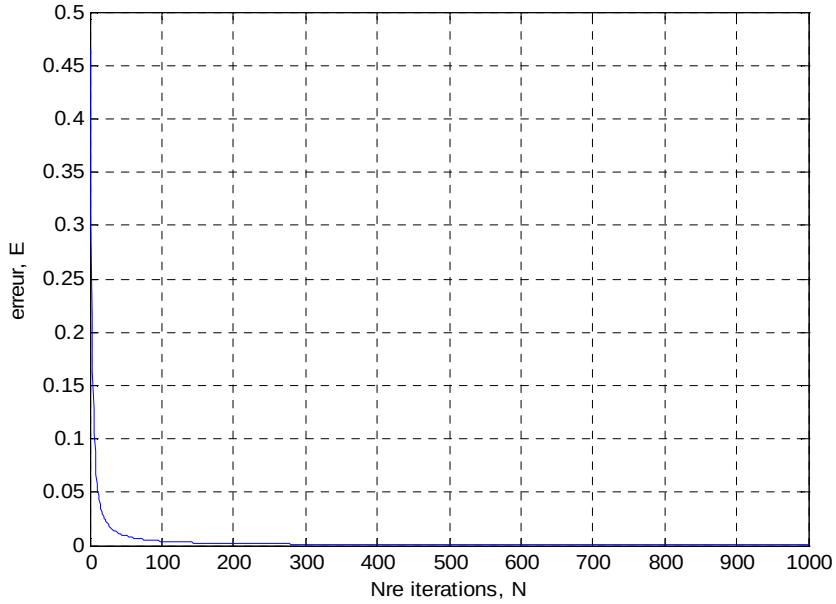


Figure IV .5.1.4: Représentation de l'erreur quadratique en fonction de nombre d'itérations.
Momentum=0.989, Pas d'apprentissage =0.002, w1=w2=10^-16
Nombre de neurones =1, Nombre d'itération =1000, erreur =3.8977 $*10^{-5}$

1	0	0
1	0	0
1	0	0

Pcc = 0.333

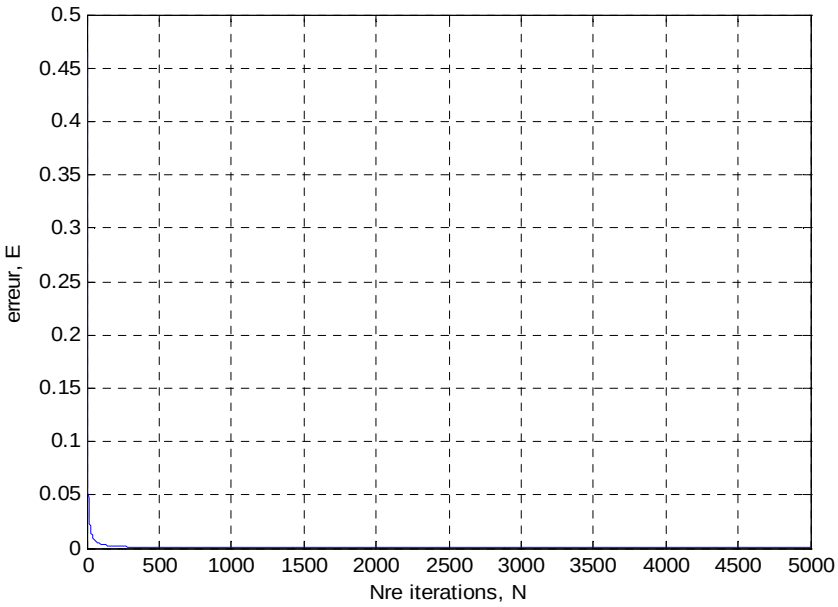


Figure IV .5.1.5: Représentation de l'erreur quadratique en fonction de nombre d'itérations.
Momentum=0.989, Pas d'apprentissage =0.002, w1=w2=10^-16
Nombre de neurones =1, Nombre d'itération =5000, erreur =7,593 *10^{-5}

1	0	0
1	0	0
1	0	0

Pcc = 0.333

D'après les courbes de l'erreur quadratique, nous avons constaté que l'erreur décroit en fonction du nombre d'itérations. On peut dire que l'algorithme converge.

La matrice de confusion donne le résultat du test effectué. Nous constatons que la totalité des pixels sont mal classés, car ils été attribués à la classe 1. Ceci donne une probabilité de classification correcte (Pcc) égale à 0,33.

IV.5.2 Influence de momentum

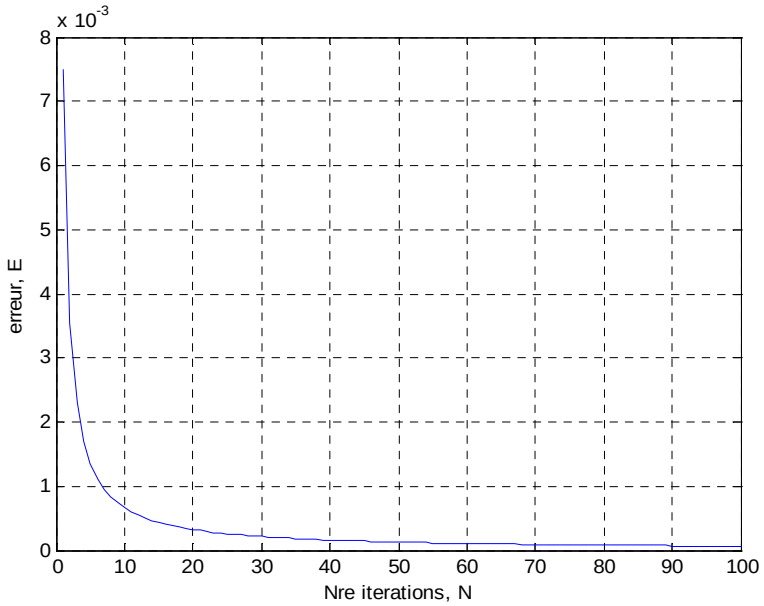


Figure IV .5.2.1: Représentation de l'erreur quadratique en fonction de nombre d'itérations.
Momentum=0.95, Pas d'apprentissage =0.002, w1=w2=10^-16
Nombre de neurones =1, Nombre d'itération =100, erreur = $6.2937*10^{-5}$

1	0	0
1	0	0
1	0	0

Pcc=O.333

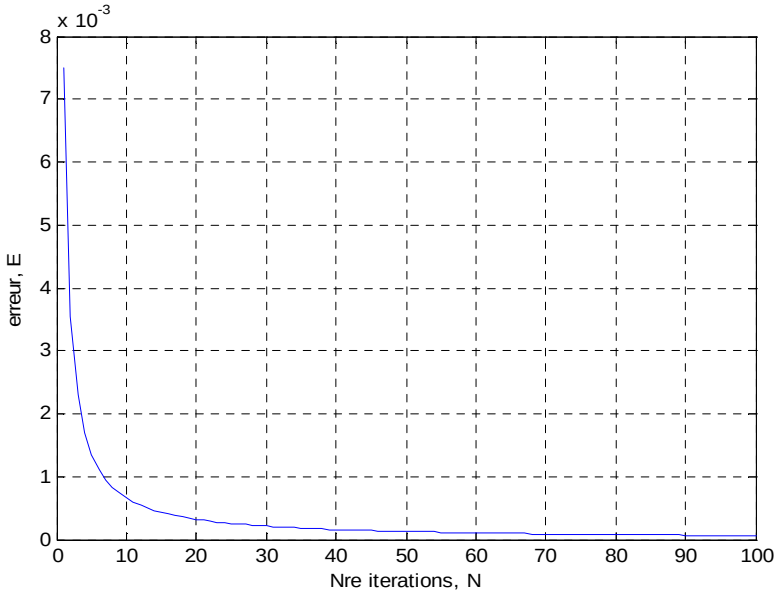


Figure IV .5.2.2: Représentation de l'erreur quadratique en fonction de nombre d'itérations.
Momentum=0.989, Pas d'apprentissage =0.002, w1=w2=10^-16
Nombre de neurones =1, Nombre d'itération =100, erreur =6.9586 *10^{-5}

1	0	0
1	0	0
1	0	0

Pcc=0.333

D'après les Figures IV.5.1.1 ; IV.5.1.3 ; IV.5.1.4 ; IV.5.1.5 ; IV.5.2.1 et IV.5.2.2 , nous constatons que l'erreur quadratique décroit et converge rapidement vers de très petites valeurs.

Ce résultat est prévisible car le momentum permet de lisser la courbe de l'erreur ce qui augmente la rapidité de la convergence de l'algorithme.

D'après les matrices de confusion obtenues, nous constatons que les pixels sont mal classés. Les résultats obtenus sont différents à chaque exécution. Ceci donne une probabilité de classification correcte (Pcc) égale à 0,33.

IV.5.3 Influence du pas du gradient

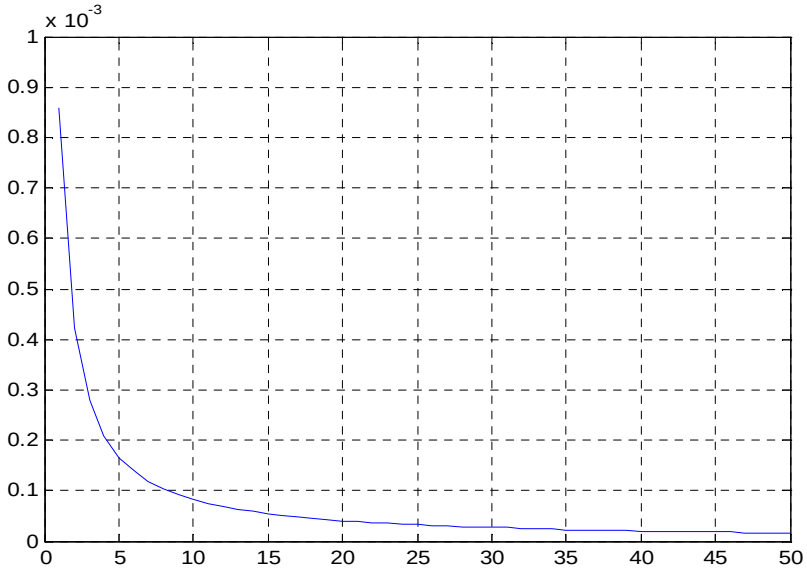


Figure IV .5.3: Représentation de l'erreur quadratique en fonction de nombre d'itérations.
Momentum=0.89, Pas d'apprentissage =0.02, w1=w2=10^-16
Nombre de neurones =1, Nombre d'itération =50, erreur = $1.629*10^{-5}$

1	0	0
1	0	0
1	0	0

Pcc= 0.333

D'après la Figure IV.5 .3 nous constatons que l'erreur quadratique décroit et vers de très petites valeurs semblablement comme pour le cas précédent.

Ce résultat n'est pas prévisible. D'après des travaux antérieurs, quand on modifie la valeur du pas d'apprentissage, on parcourt la courbe « coût » qui est l'erreur quadratique dans les deux sens. L'algorithme doit converger vers un minimum global.

D'après les matrices de confusion obtenues, nous constatons que les pixels sont mal classés également comme pour le cas précédent. La totalité des pixels sont attribués à la classe 1. Ceci donne une probabilité de classification correcte (Pcc) égale à 0,33.

IV.6 Résultats obtenus avec l'algorithme de Kalman

Les images sont très mal segmentées en utilisant l'algorithme de rétro propagation du gradient de l'erreur. Le filtre de Kalman vient corriger les insuffisances de la méthode neuronale. Nous avons obtenu de mauvais résultats avec la méthode neuronale.

Le filtre de Kalman remplace le pas d'apprentissage et le momentum de la méthode neuronale. La méthode est basée sur le calcul de l'inverse de la matrice de corrélation des données à classifier (l'ensemble des pixels des trois images). Nous n'avons pas pu exécuter ce calcul et par conséquent nous n'avons pas de résultats de simulation à présenter avec la méthode de Kalman.

IV.7 Conclusion

D'après les résultats de simulation obtenus, nous remarquons que la courbe de l'erreur quadratique diminue en fonction des différents paramètres du perceptron à une couche cachée.

Les images sont très mal segmentées d'après les valeurs des probabilités de classification correcte obtenue. Ceci peut être interprété par :

- Le mauvais choix des paramètres du réseau neuronal (pas d'apprentissage, momentum, nombre d'itérations,….) ;
- Le type et la qualité des images utilisées ;
- Le choix de la méthode neuronale.

Conclusion

Ce mémoire porte sur la segmentation d'images en utilisant la rétro propagation du gradient de l'erreur et le filtre de Kalman. Le travail demandé était d'appliquer les deux méthodes afin de segmenter des images.

L'algorithme de la rétro propagation neuronale est basé sur l'apprentissage supervisé. Ceci exige la connaissance à priori des classes d'appartenance de chaque pixel. Pour ce faire, nous avons appliqué la méthode des k_means. Le résultat obtenu constitue la sortie désirée du réseau de neurones.

La mise au point de ce modèle neuronal passe par une phase d'apprentissage. Cette étape dépend de l'architecture du réseau et d'une règle d'adaptation des poids des neurones. Le choix d'une procédure d'apprentissage, et de l'architecture d'un réseau, sont de ce fait fortement dépendant de la tâche à apprendre. Lorsque l'apprentissage s'effectue « trop bien », les capacités de généralisation (test) se trouvent détériorées par un phénomène de sur-apprentissage.

Le filtre de Kalman a été proposé afin d'améliorer les résultats obtenus par l'apprentissage neuronal.

L'algorithme de Kalman converge théoriquement beaucoup plus rapidement que celui de la rétro propagation. Du fait que l'algorithme de la rétro propagation dépend de l'initialisation des poids , donc le filtre de Kalman est plus approprié.

Nous avons jugé que les résultats obtenus sont pas du tout satisfaisants. Les images sont mal segmentées. Ceci est dû aux mauvais choix des paramètres du réseau. Eventuellement, le type d'images. Le principal inconvénient de la rétro propagation est que cette méthode est empirique. Le choix des paramètres se fait par tâtonnement. De plus, l'apprentissage du réseau de neurones est très lent.

Perspectives

Comme perspectives, nous proposons :

- d'augmenter le nombre d'itérations, ceci va permettre d'éviter les minimums locaux ;
- prendre une autre fonction d'activation que la sigmoïde (tangente hyperbolique,……) ;
- essayer un autre type d'images ;
- prendre un nombre important de classe, ceci peut éventuellement réduire la confusion.

Bibliographie

[1] HAMMADI A. et SAIDANI A. : **« *Segmentation d'image couleur par les réseaux de neurones par apprentissage compétitif* »**. Mémoire Ingénieur, département d'électronique, Université Mouloud MAMMERI, Tizi Ouzou.2005

[2] BOUBKEUR M., LAMI B. et BAZIZ S. : « ***Segmentation d'image couleur basée sur les espaces colorimétriques ».*** Mémoire Ingénieur, département d'électronique, Université Mouloud MAMMERI, Tizi Ouzou.2008

[**3]** KHADRAOUI M. et MAKHTOUB B. : ***« Application des réseaux de neurones pour la classification des visages ».*** Mémoire Ingénieur, département d'électronique, Université Mouloud MAMMERI, Tizi Ouzou.2007

[4] MESSAOUDI H., OULD DRIS N. et ZEHRAOUI S. : ***« Segmentation d'image par la transformation en ondelettes sous Matlab ».*** Mémoire Ingénieur, département d'électronique, Université Mouloud MAMMERI, Tizi Ouzou.2003

[6] OULD ABEDERRAHMANE S. et MEHALLA Y. : ***«* Application du réseau de neurones multicouches à la classification des caractères isolés ».** Mémoire Ingénieur, département d'automatique, Université Mouloud MAMMRI, Tizi Ouzou, 2007.

[7] KONATE DAOUDA MAMADOU M. : ***« La modélisation des données de précipitations par réseaux de neurones » .*** Mémoire Ingénieur, département d'électronique, Université Mouloud MAMMRI Tizi Ouzou. 2005

[8] OSMANI Hafsa et NEMMOUR Hssiba ; « ***Classification des images satellitaires par les réseaux de neurones*** », mémoire d'ingénieur, FGEI, USTHB, 2001.

[10] TRADIEN Samuel : ***« filtre de Kalman ».*** Ecole nationale des télécommunications, Brique Rose.

[13] Olivier BONNET TORRES : ***« Filtrage de Kalman appliqué à la navigation inertielle ».*** Onera_DCSD , 15 décembre 2003.

Sites internet

[5] **: site k_means**

[9] : http://membres.lycos.fr/haboun/file/chap2.pdf

[11] : www.lodyc.jussien.fr/mblod/Doc/KLmF.pdf.

[12] : fr.wikipidia.org/wiki/filtre_de_Kalman