REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mouloud Mammeri de Tizi-Ouzou
Faculté de Génie Electrique et d'Informatique
Département d'Electronique

Mémoire de fin d'études

En vue de l'obtention du diplôme d'ingénieur d'état en électronique
Option : *Communication*

Thème

Reconnaissance de Locuteurs

Proposé et dirigé par :
Melle AÏT OUAZZOU H.

Présenté par :
GUENOUN DJAMEL

Promotion: 2008

SOMMAIRE

I.1 LA RECONNAISSANCE AUTOMATIQUE DU LOCUTEUR (RAL) [1][4]

La reconnaissance automatique du locuteur (RAL) s'inscrit dans le domaine plus général de la communication homme machine. Il s'agit de reconnaître automatiquement l'identité d'une personne prononçant une ou plusieurs phrases, comme un auditeur humain identifie son interlocuteur au cours d'une conversation. Quelques tâches voisines de la RAL sont l'adaptation au locuteur pour la reconnaissance automatique de la parole et la détection de changement de locuteurs au cours d'un dialogue. Les applications directes de la RAL concernent les problèmes de confidentialité et d'authentification. Nous distinguerons :

1- les applications sur site : serrures vocales pour contrôle d'accès, cabines bancaires en libre service.

2- les applications liées aux télécommunications : ces applications concernent l'identification du locuteur à travers le réseau téléphonique pour accéder à un service de transactions bancaires à distance ou pour interroger des bases de données en accès privé.

3- les applications judiciaires: recherche de suspects, orientations d'enquêtes, preuves lors d'un jugement.

La difficulté de la tâche de reconnaissance n'est pas la même d'une application à l'autre. Dans le cas des applications sur site, l'environnement de prononciation de la phrase ou du mot de passe est plus facilement contrôlé que dans le cas des applications via le réseau téléphonique (distorsions dues au canal, différences entre les combinés téléphoniques, bande passante limitée). Les applications judiciaires présentent quand à elles des difficultés d'un autre ordre (locuteurs non coopératifs, enregistrements de mauvaise qualité).

I.2 STRUCTURE D'UN SYSTÈME DE RAL [1][2][4]

La tâche de reconnaissance automatique du locuteur peut se subdiviser en trois étapes : *paramétrisation, classification et décision*. Un premier module de traitement du signal réalise l'analyse acoustique du signal de parole. A l'issue de cette étape, le

signal est représenté par des vecteurs de coefficients, ce qui permet de réduire l'information en quantité et en redondance. Ces vecteurs sont éventuellement représentés par un modèle mathématique, on parle alors de méthodes paramétriques. Dans la phase de classification, les vecteurs du signal de test (ou leur modèle) sont comparés aux vecteurs des locuteurs de référence (ou à leurs modèles). La phase de décision désigne le locuteur finalement reconnu ou pas. La structure d'un système d'identification du locuteur en ensemble fermé est représentée sur la Figure III.1.

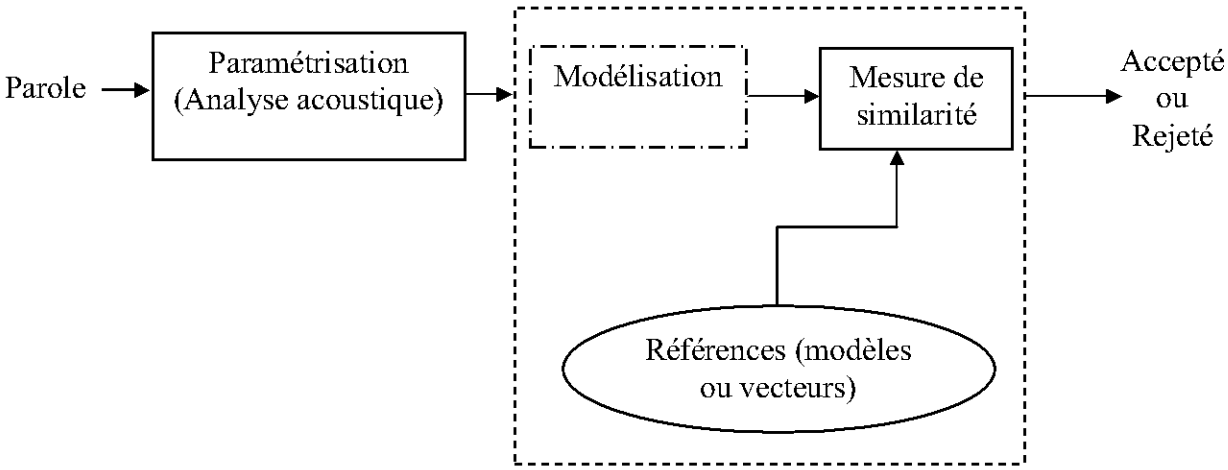

Figure I.1 : Le *schéma modulaire d'un système d'identification du locuteur en ensemble fermé.*

Les différents systèmes de RAL existants se distinguent d'une part suivant les paramètres qu'ils utilisent et d'autre part suivant les différents classifieurs qui prennent la décision finale.

I.3 LA BANQUE DE DONNÉES UTILISÉES

Nous disposons d'une banque d'enregistrement que nous avons constitué. Nous avons enregistré 50 étudiants (hommes et femmes).

La phrase prononcée par les locuteurs utilisée pour l'apprentissage est " *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas*". La phrase prononcée par

les locuteurs utilisée pour le test est " *En pays d'exil, même le printemps manque de charme*".

L'enregistrement a été effectué à l'université de Mouloud Mammeri, Faculté de Génie Electrique et d'Informatique. Les voix des étudiants ont été enregistrées à l'amphi "A", en utilisant un microphone " Sony " et un logiciel d'acquisition " GoldWave ".

La fréquence d'échantillonnage est 8000 Hz. La durée moyenne des enregistrements effectués est 9 secondes.

Les Figures suivantes présentent quelques exemples d'enregistrement de trois étudiantes (Imane, Linda et Ouiza) et trois étudiants (Ghani, Mourad et Toufik).

La phrase prononcée est " *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas*".

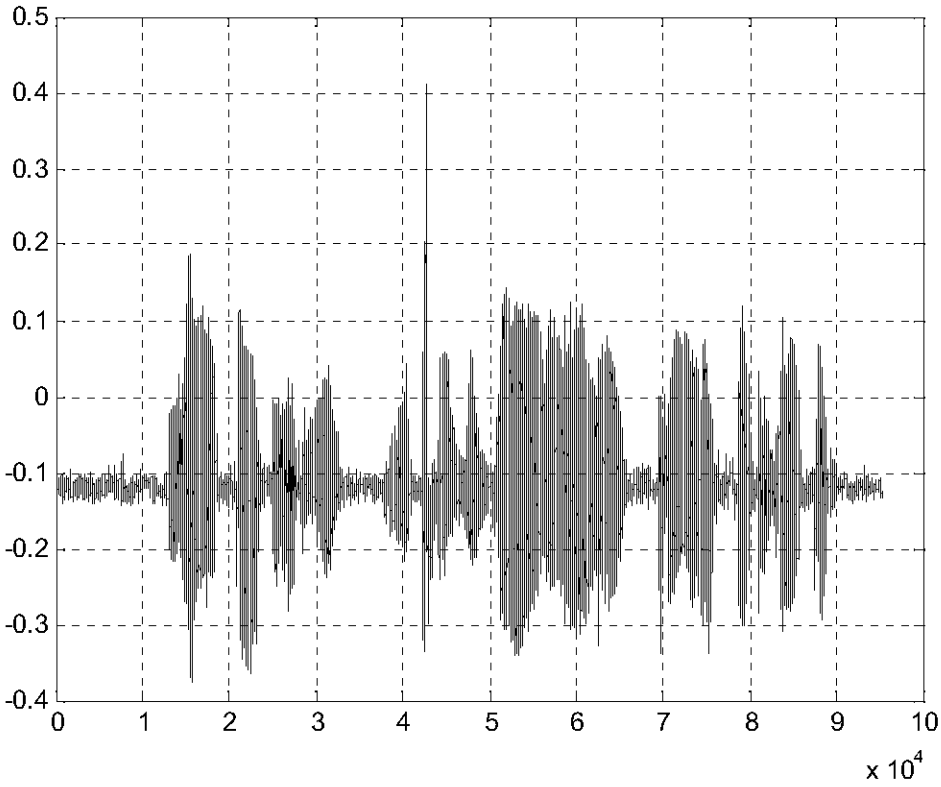

Figure I.2 : Représentation temporelle de la *voix de* ***Imane*** pour la phrase : " *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

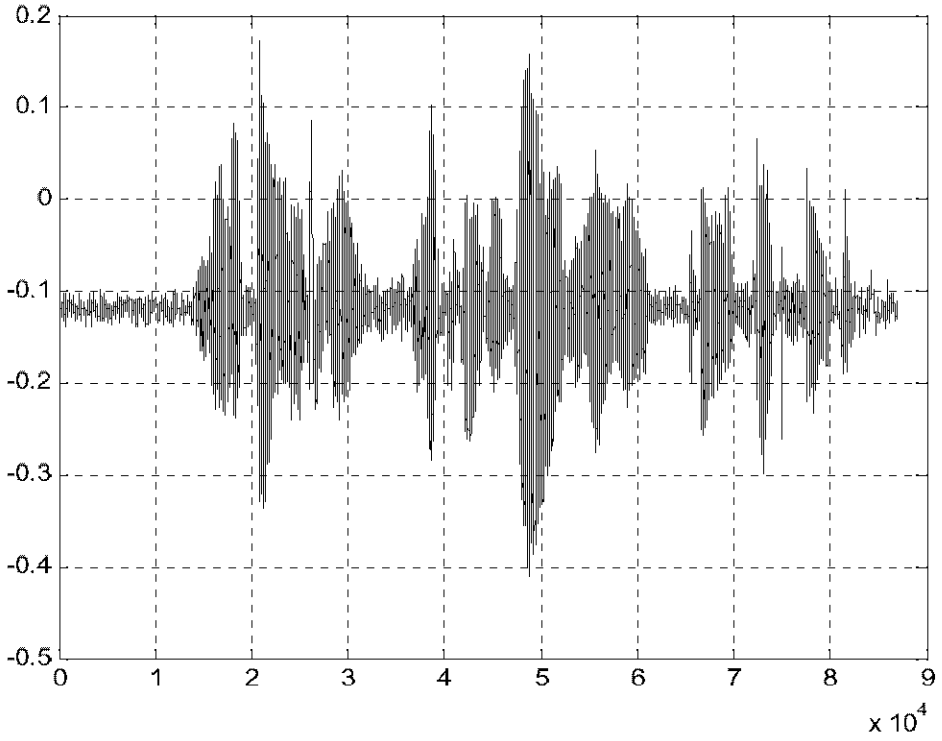

Figure I.3 : Représentation temporelle de la *voix de* ***Linda*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

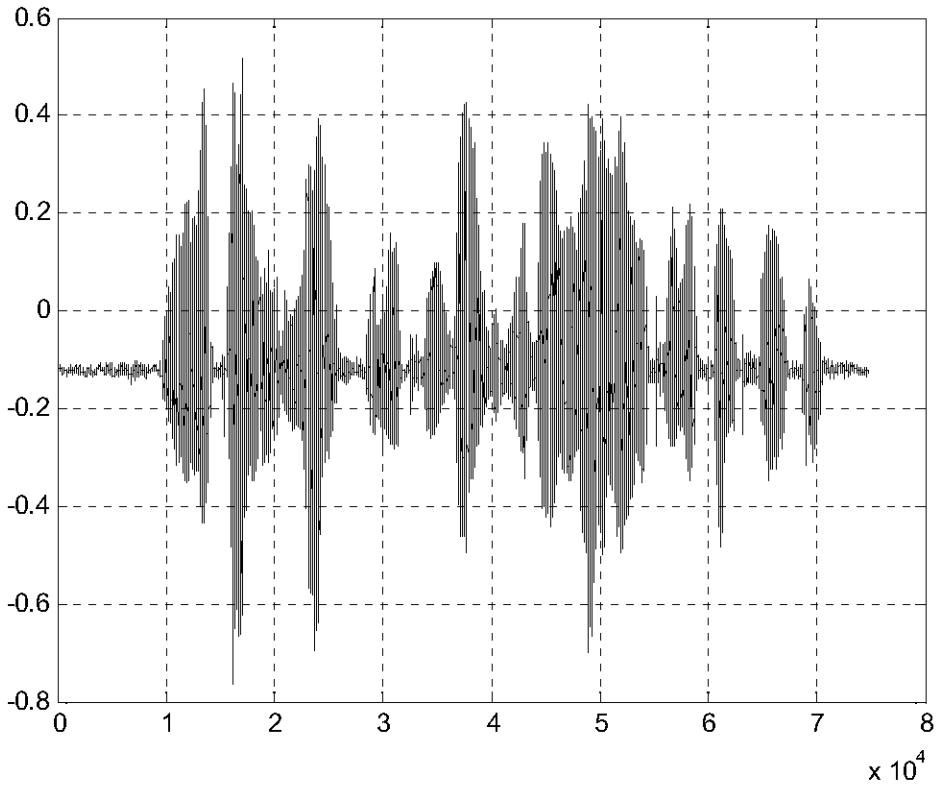

Figure I.4 : Représentation temporelle de la *voix de* ***Ouiza*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

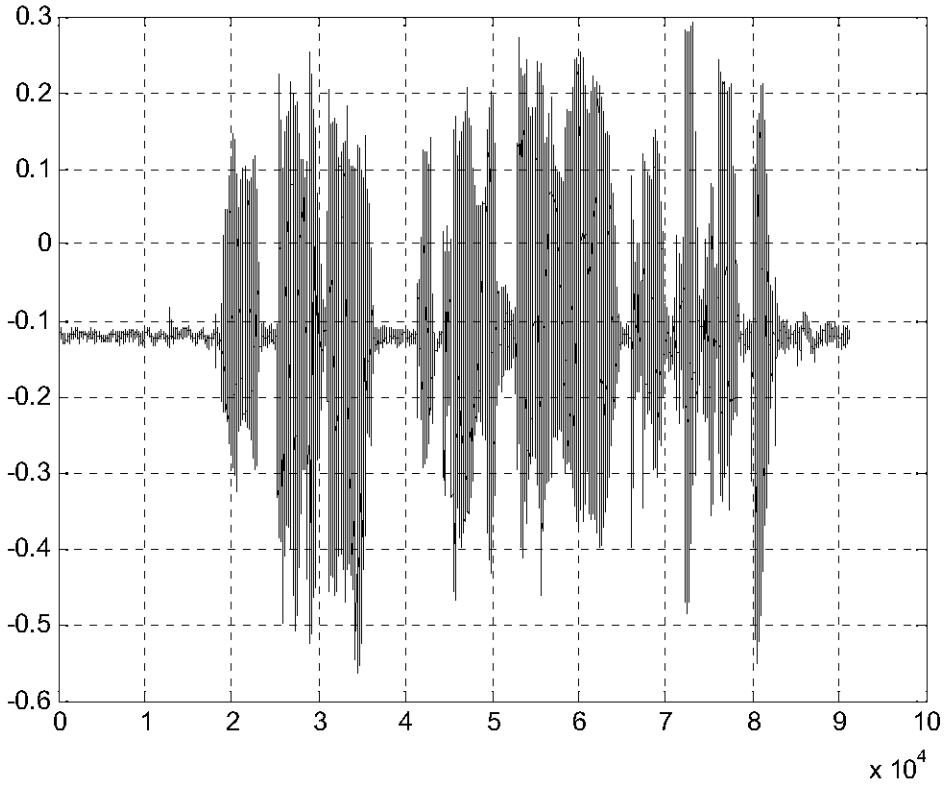

Figure I.5 : Représentation temporelle de la *voix de* ***Ghani*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

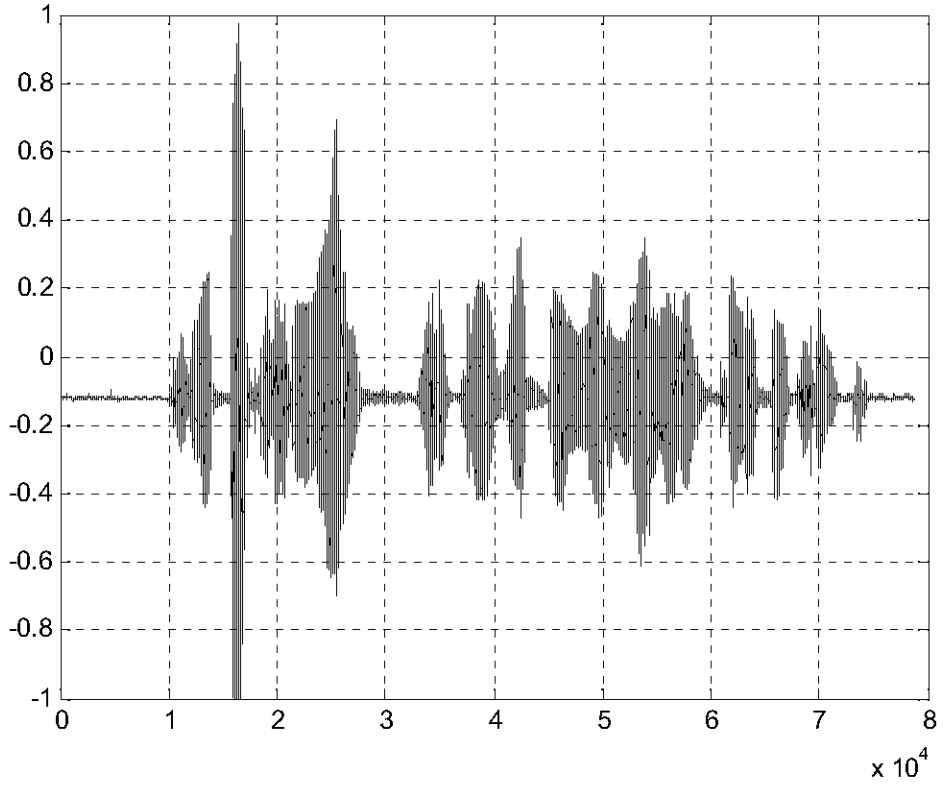

Figure I.6 : Représentation temporelle de la *voix de* ***Mourad*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

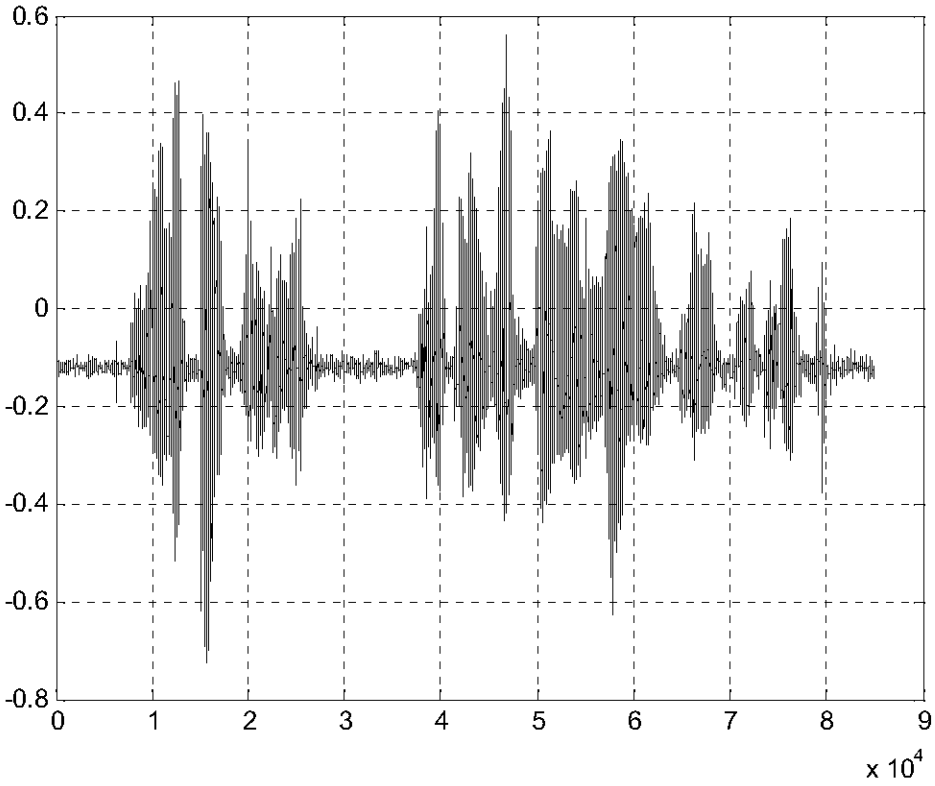

Figure I.7 : Représentation temporelle de la *voix de* ***Toufik*** pour la phrase : " *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

I.4 TRAITEMENT DES DONNÉES

Le prétraitement des données est une étape importante car elle permet de sélectionner, dans l'espace de représentation, l'information nécessaire à l'application. Cette sélection passe par l'élimination du bruit dû aux conditions d'acquisition, la normalisation des données, et la suppression de la redondance.

Les données brutes comportent du bruit. Les classifieurs exigent un nombre de caractéristiques relativement faible pour répondre aux impératifs du traitement en temps réel. Une opération de prétraitement sur les données brutes est donc nécessaire pour préparer la phase de classification.

1.4.1 Application de la transformée en ondelettes

Nous avons utilisé la décomposition en ondelettes pour filtrer les données et réduire la dimension des vecteurs obtenus.

D'après la bibliographie, on utilise souvent l'ondelette Daubechies 1 pour filtrer et spécialement pour débruiter les données.

Concernant notre application, nous avons effectué une décomposition à cinq niveaux. Les Figures ci-dessous donnent quelques exemples.

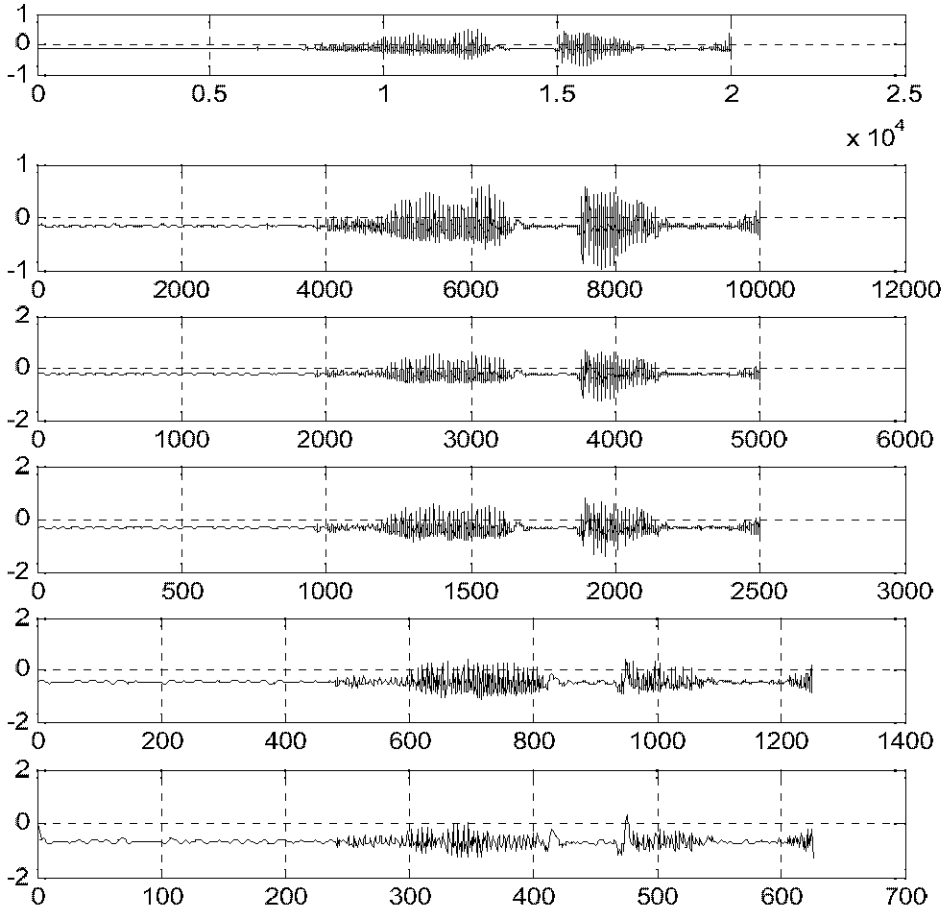

Figure I.8 : La *décomposition à 5 niveaux de la voix de* ***Toufik*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

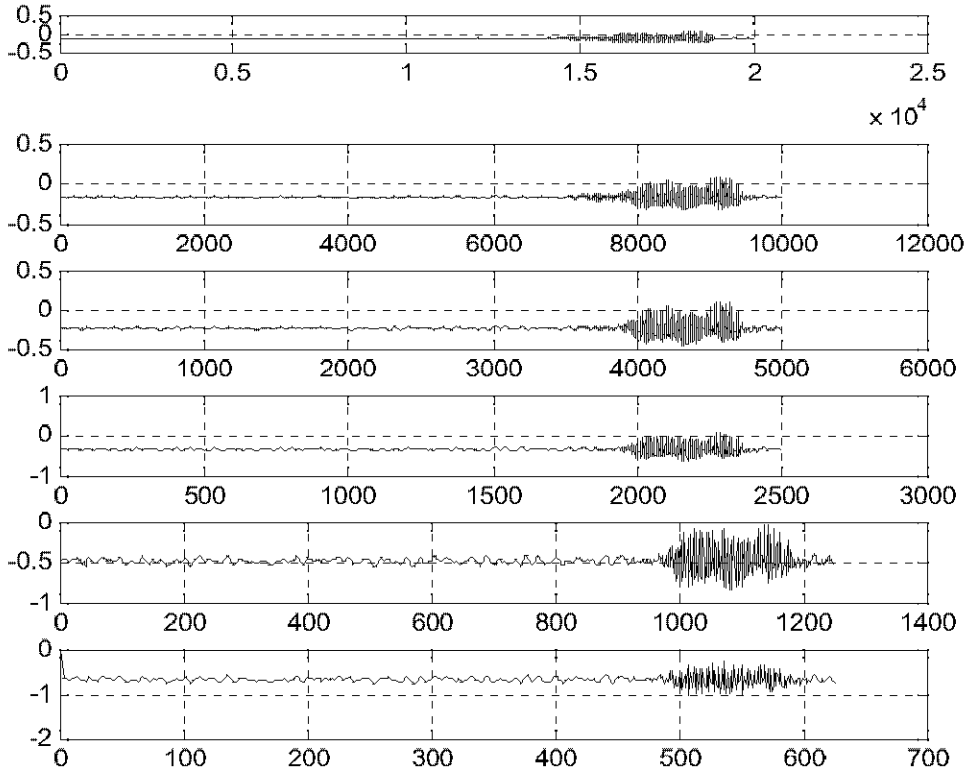

Figure I.9 : La *décomposition à 5 niveaux de la voix de* ***Linda*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

1.4.2 L'opération de la préaccentuation

En pratique, on constate que le spectre des sons voisés chute d'environ 6 dB par octave, ce qui a pour effet de mal représenter les hautes fréquences comparativement aux basses fréquences. Pour remédier à cette situation, on applique un filtre de préaccentuation ou pré emphase selon l'équation $S' = S_n - k\ S_{n-1}$ Où S_n est le signal échantillonné, l'équation précédente est appliquée à tous les N échantillon (S_n avec n = 1,.., N) que compte chaque fenêtre. k est le coefficient de préaccentuation tel que : $0 \leq k \leq 1$. Généralement k est choisit entre 0,9 et 1.
Concernant notre application, k = 0,97.

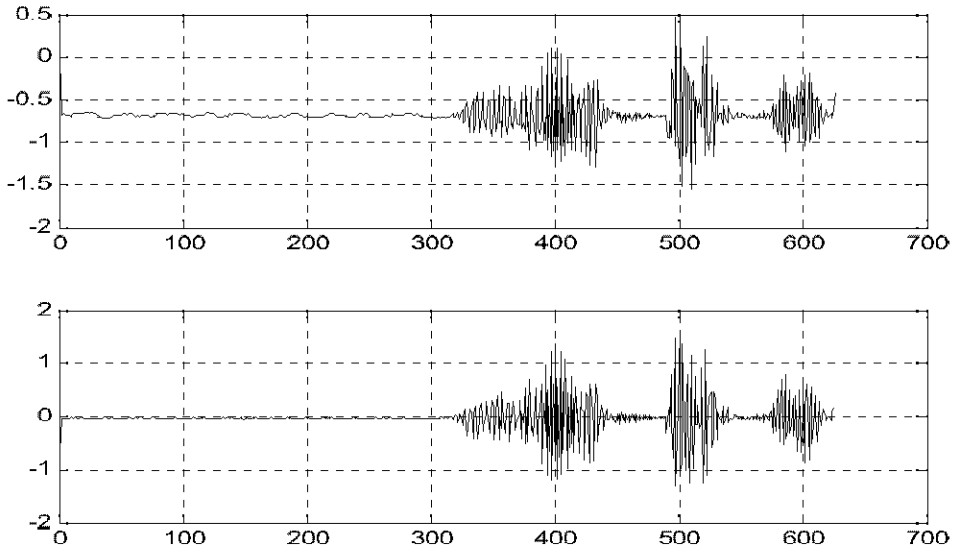

Figure I.10 : *L'effet de la préaccentuation de la voix de* ***Mourad*** pour la phrase : *" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".*

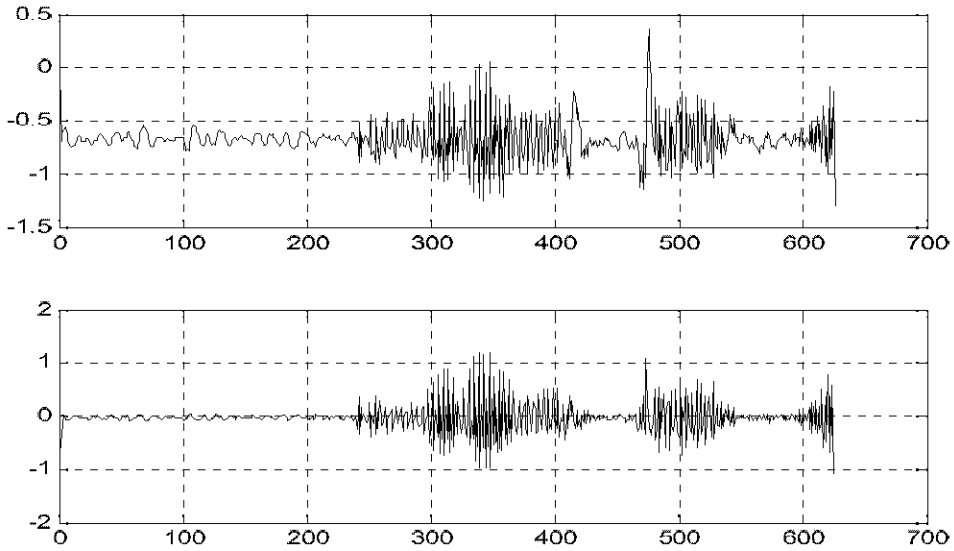

Figure I.11 : *L'effet de la préaccentuation de la voix de* ***Toufik*** pour la phrase : *" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".*

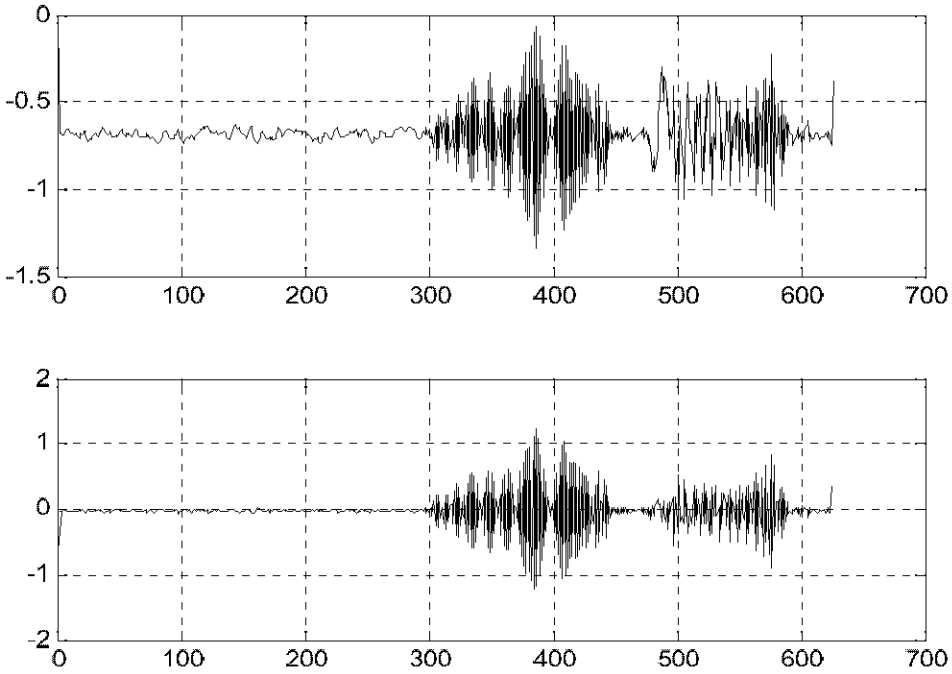

Figure I.12 : *L'effet de la préaccentuation de la voix de* ***Ouiza*** pour la phrase : *" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".*

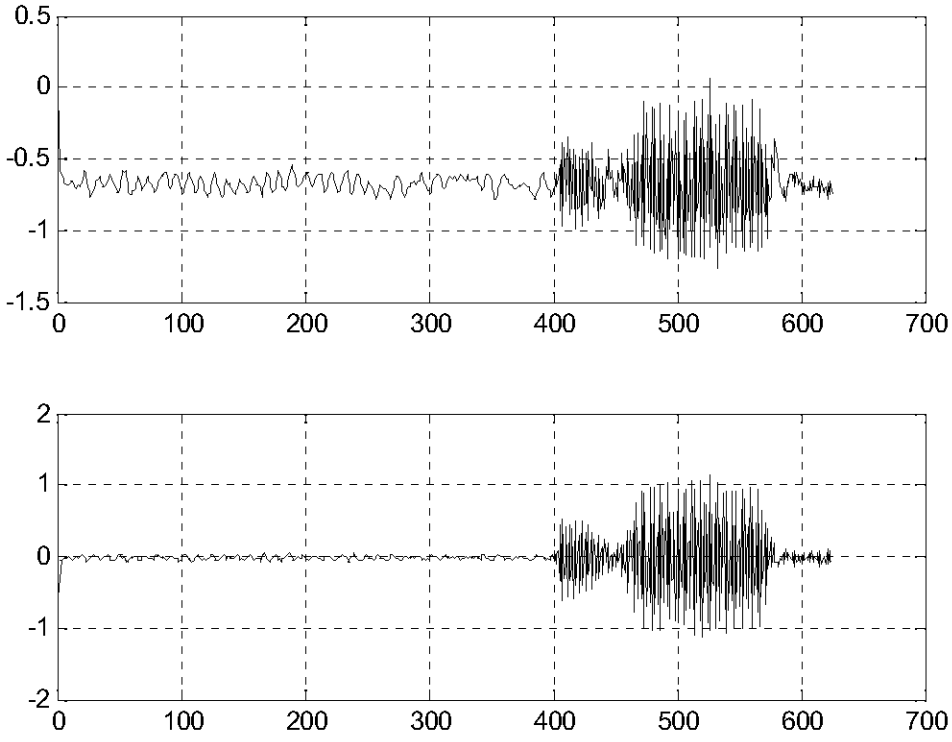

Figure I.13 : *L'effet de la préaccentuation de la voix de* ***Imane*** pour la phrase : *" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".*

1.4.3 Application du fenêtrage (Pondération)

L'opération du fenêtrage est nécessaire car la décomposition du vecteur en plusieurs trames est inévitable. Par conséquent, le vecteur à décomposer est multiplié par une porte rectangulaire ce qui fait apparaître des lobes secondaires dans le spectre du signal à décomposer.

Pour pallier à ce problème, d'autres fenêtres sont utilisées autres que la porte rectangulaire telles la fenêtre Hamming, Hanning, Blackman, Barteltt,…….. Ces fenêtres transforment les (*n*) éléments symétriques en vecteur (*W*), *n* est entier.

La fenêtre Hamming

$$W[k+1] = 0,54 - 0,46 \cos\left(2\pi \frac{k}{n-1}\right) \quad ; \qquad k = 0, 1\ldots\ldots\ldots, n-1.$$

La fenêtre IIanning

$$W[k+1] = 0,5\left[1 - \cos\left(2\pi \frac{k}{n-1}\right)\right] \quad ; \qquad k = 0, 1\ldots\ldots\ldots, n-1.$$

La fenêtre Blackman

$$W[k+1] = 0,42 - 0,5 \cos\left(2\pi \frac{k}{n-1}\right) + 0,08 \cos\left(4\pi \frac{k}{n-1}\right) \quad ; \quad k = 0\ldots\ldots\ldots n-1$$

Nous donnons des exemples de fenêtres de pondération. Voir Figure I.14. Concernant notre application, nous avons transformé le vecteur en une matrice en utilisant une fenêtre Hamming avec un chevauchement entre les trames de 50%.

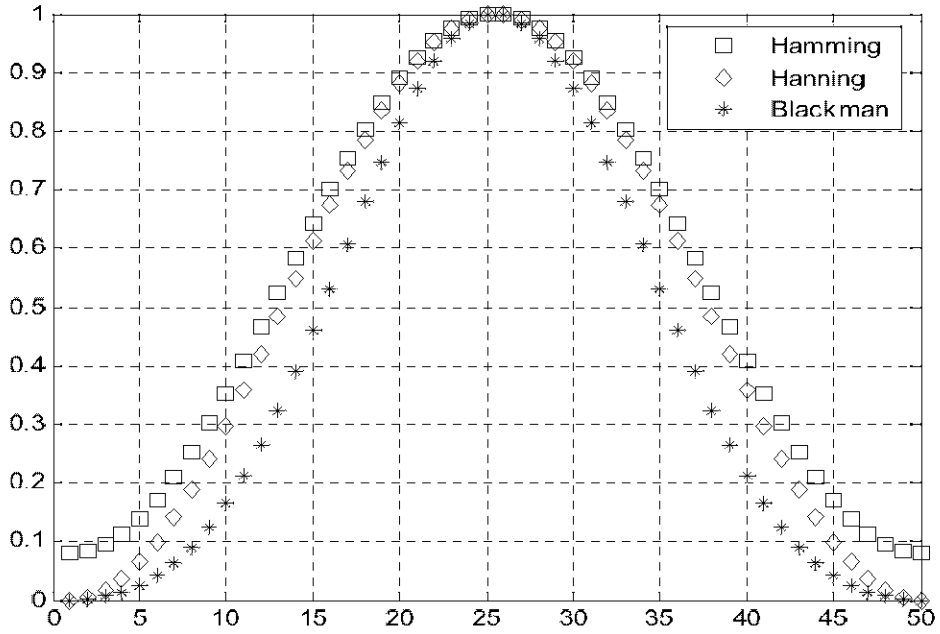

Figure I.14 : *Exemple de trois fenetres de pondération. Hamming, Hanning et Blackman.*

Les Figures suivantes donnent des exemples de pondération de quelques signaux temporels.

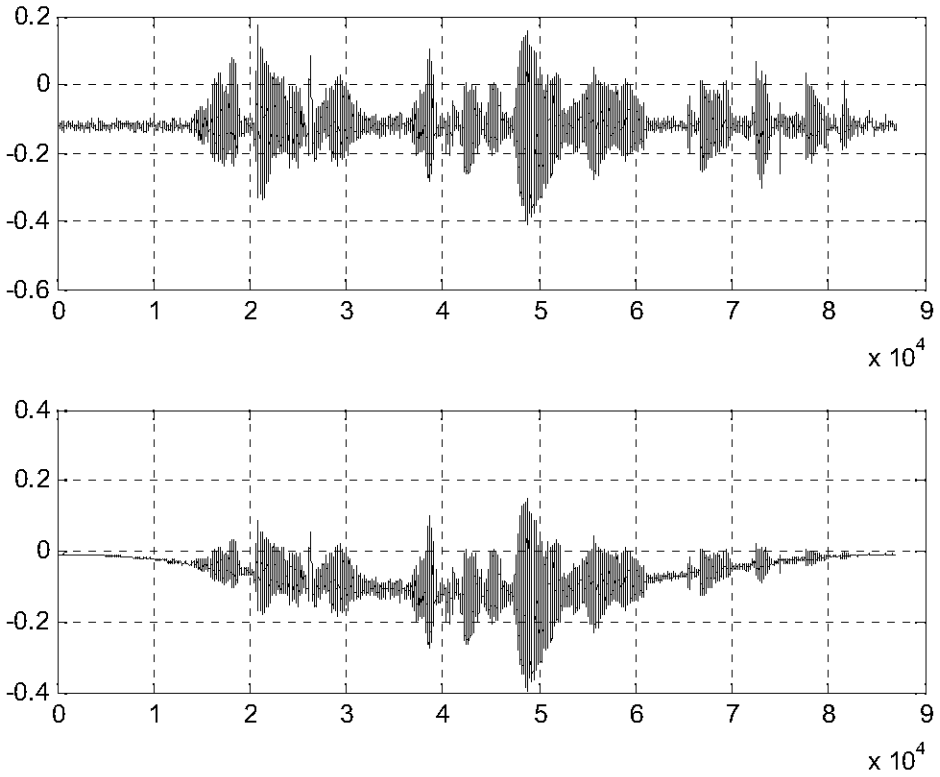

Figure I.15 : *Pondération Hamming de la voix de* ***Linda*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

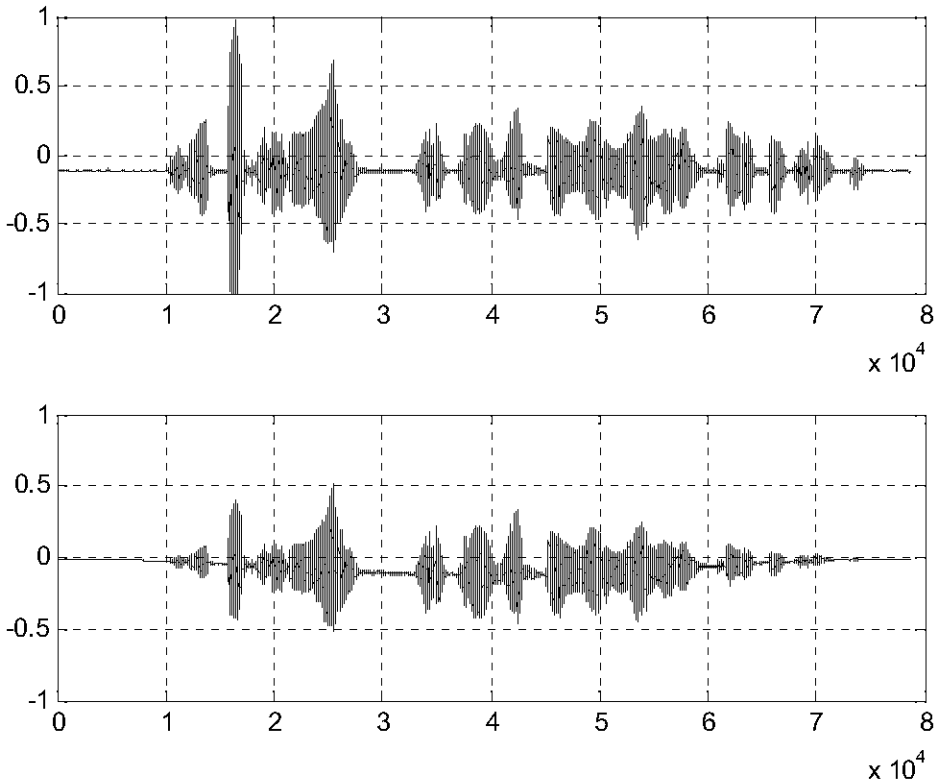

Figure I.16 : *Pondération Hamming de la voix de* ***Mourad*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

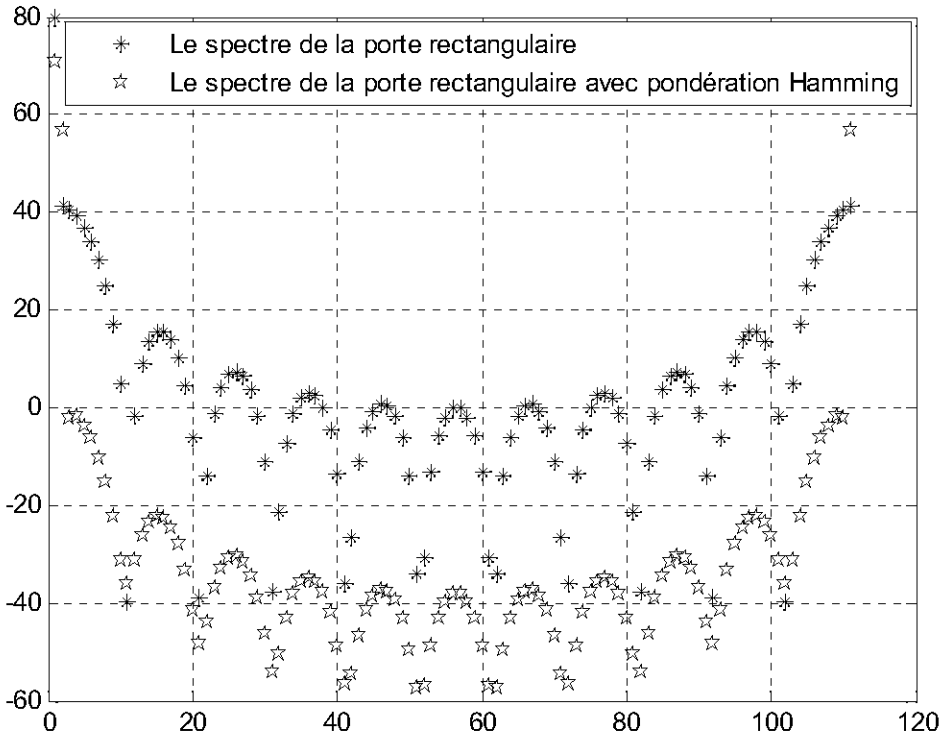

Figure I.17 : *Réduction des niveaux des lobes secondaire en pondérant la porte rectangulaire par Hamming.*

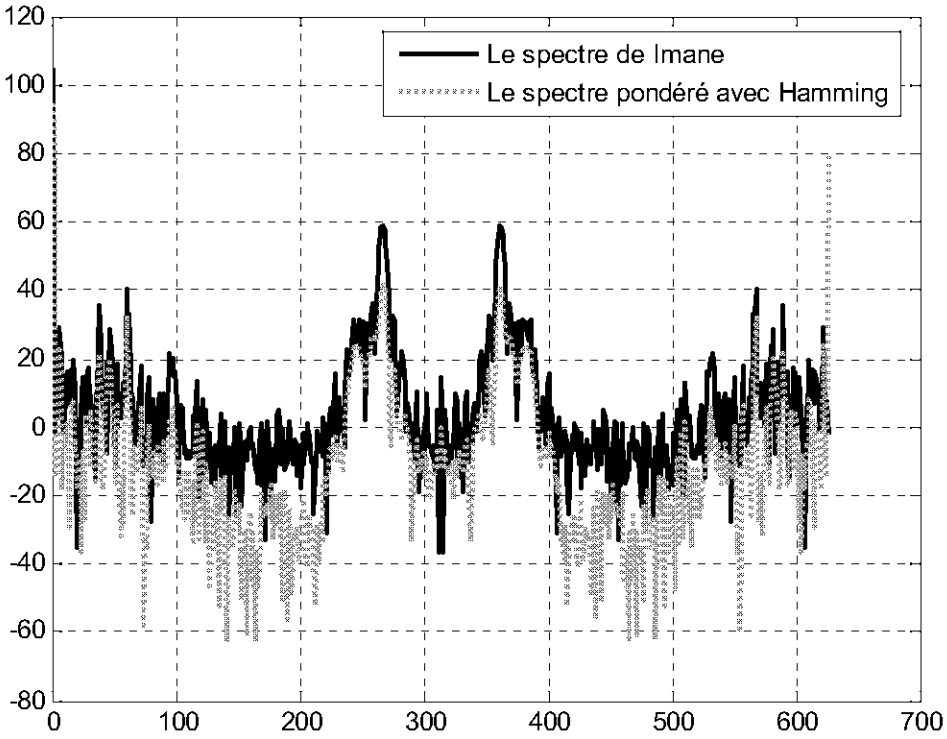

Figure I.18 : *Réduction des niveaux des lobes secondaires du spectre de Imane avec Hamming.*

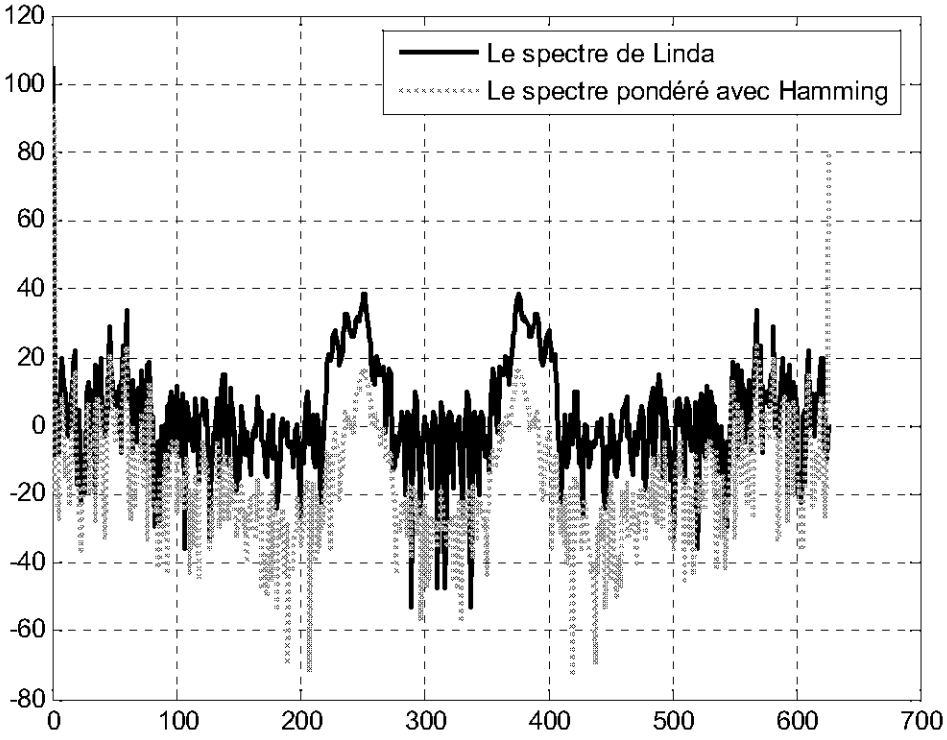

Figure I.19 : *Réduction des niveaux des lobes secondaires du spectre de Linda avec Hamming.*

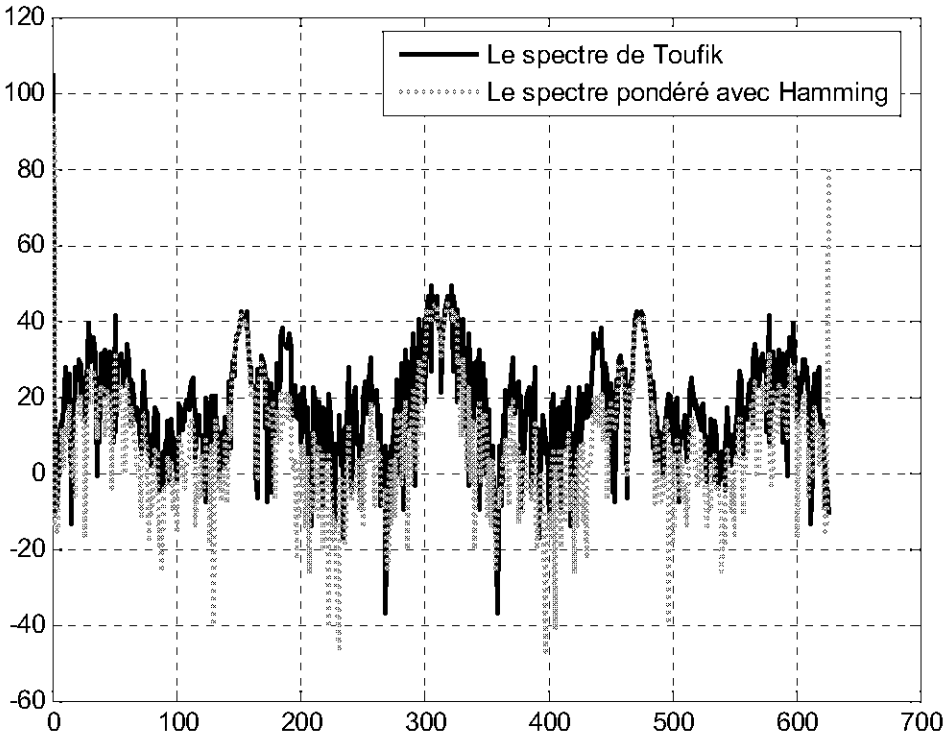

Figure I.20 : *Réduction des niveaux des lobes secondaires du spectre de Toufik avec Hamming.*

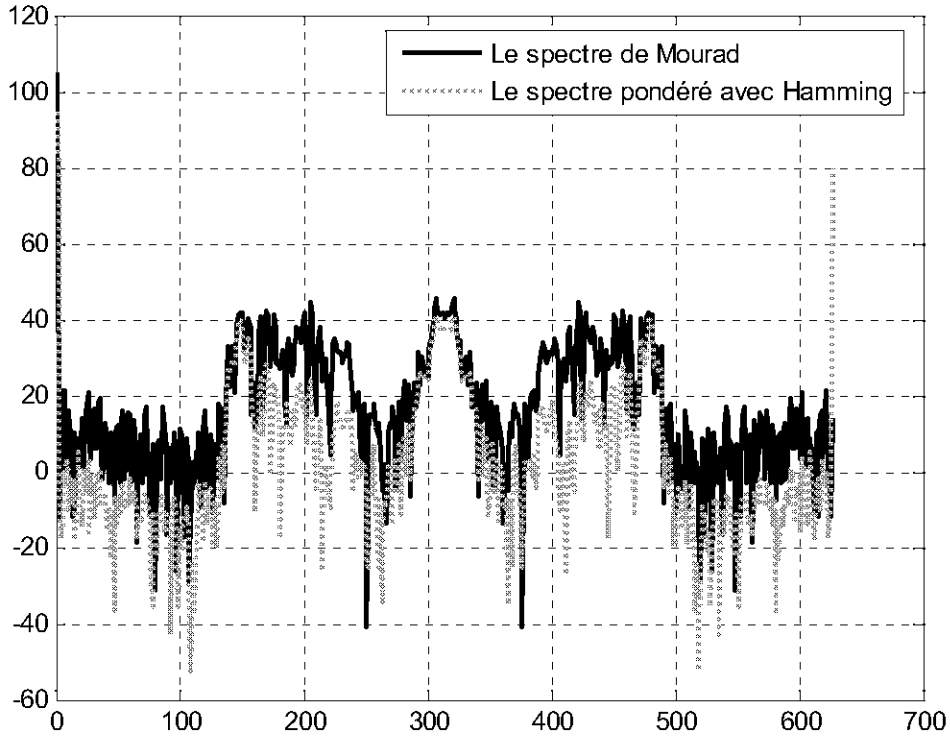

Figure I.21 : *Réduction des niveaux des lobes secondaires du spectre de Mourad avec Hamming.*

Les Figures I.15 et I.16 montrent l'effet de la pondération sur les signaux dans le domaine temporel. Le signal résultant prend pratiquement la forme de la fenêtre de pondération.

La Figure I.17 montre que la fenêtre de pondération Hamming permet de réduire le niveau des lobes secondaires de 40 dB pour le fenêtre rectangulaire. L'analyse des Figures I.18, I.19, I.20 et I.21 montre l'efficacité de l'opération de fenêtrage. De cette façon, les fréquences noyées auparavant dans les lobes secondaires vont apparaître dans le spectre du signal pondéré.

Remarque :

Nous tenons à rappeler que nous avons appliqué la pondération Hamming sur les vecteurs issus de la décomposition en ondelettes.

1.4.4 Le spectogramme

Nous présentons dans ce qui suit les spectogrammes de quelques voix enregistrées. Le spectogramme est une représentation tridimensionnelle. Nous

avons le temps en abscisses, la fréquence en ordonnées et l'amplitude de la transformée de Fourier en niveaux de gris.

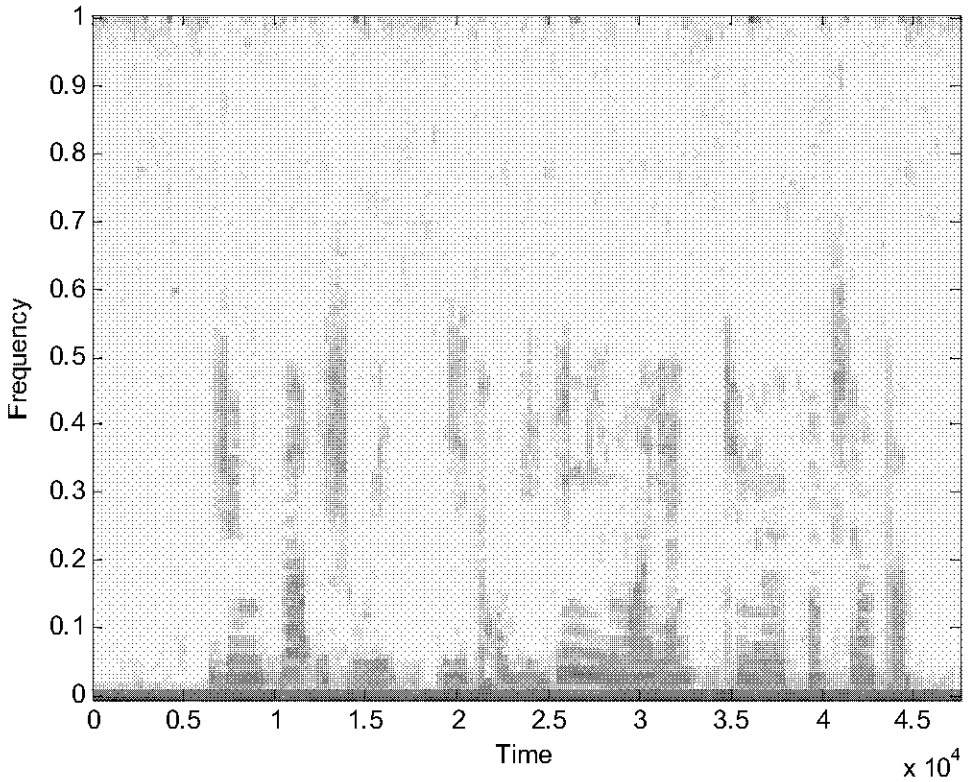

Figure I.22 : *Le spectogramme de* ***Imane*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

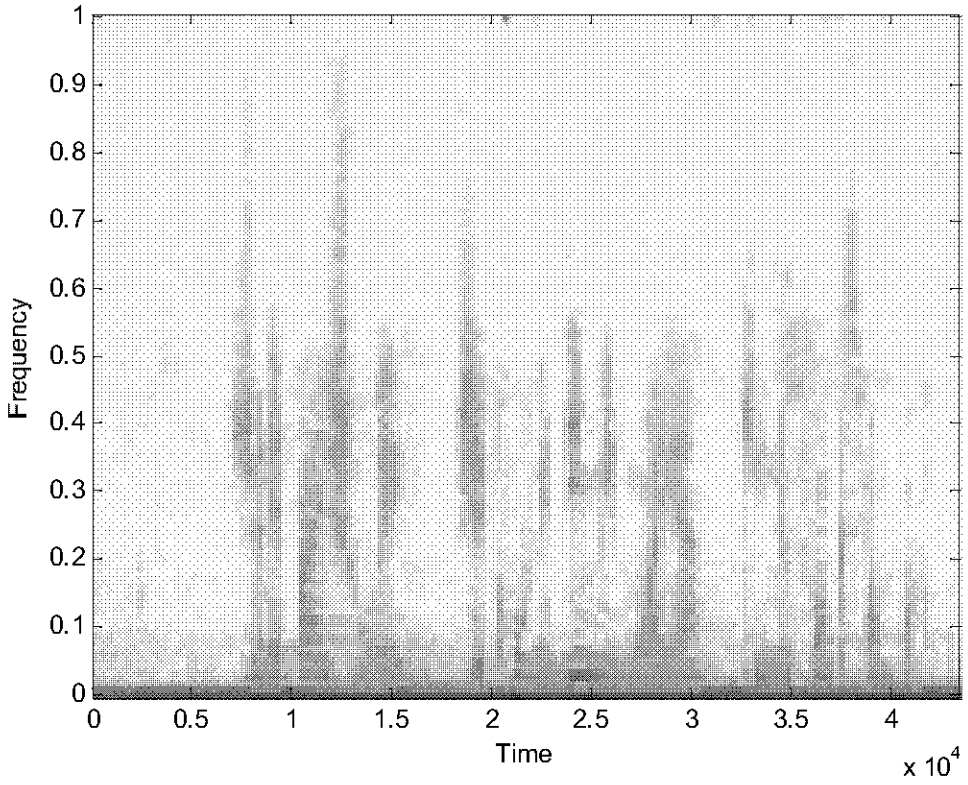

Figure I.23 : *Le spectogramme de* ***Linda*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

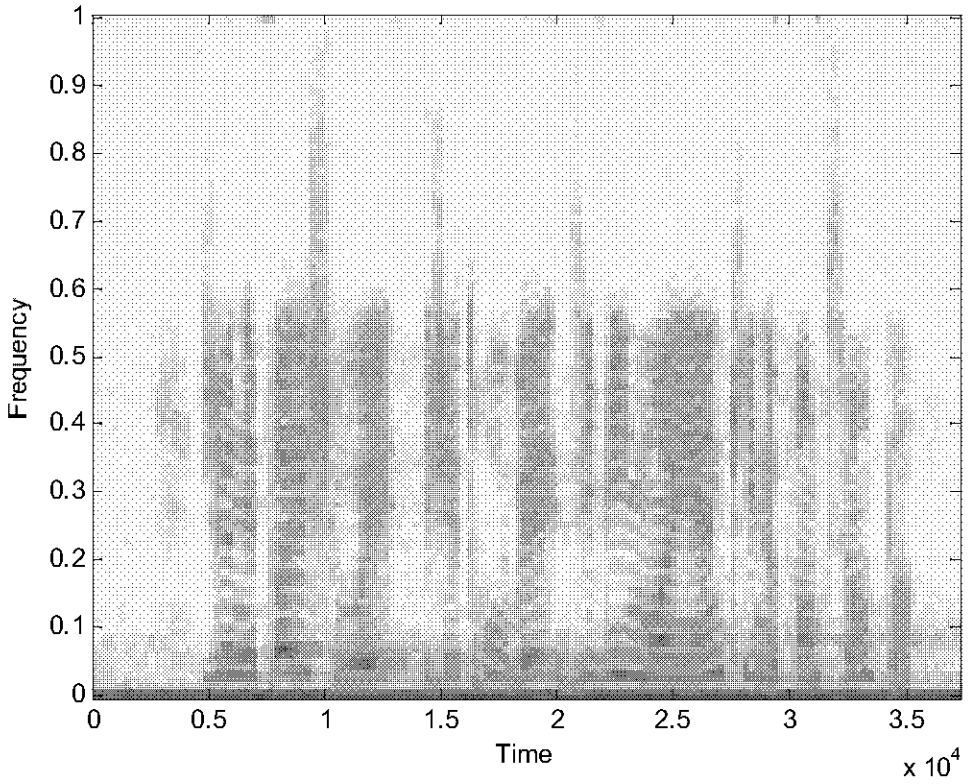

Figure I.24 : *Le spectogramme de* ***Ouiza*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

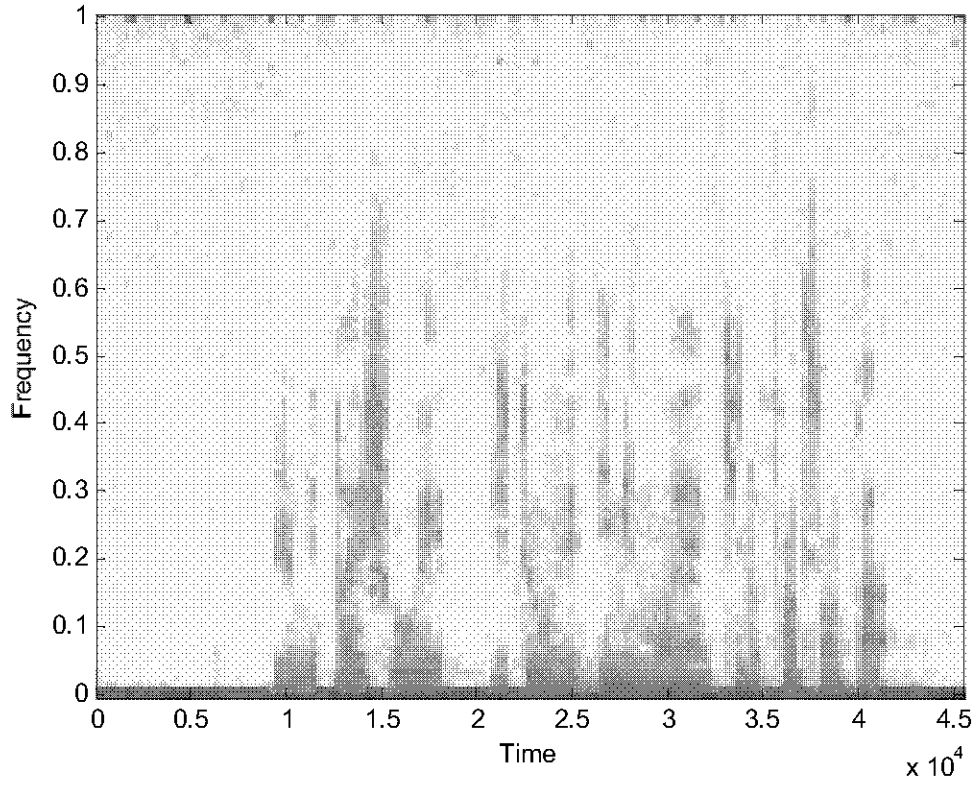

Figure I.25 : *Le spectogramme de* ***Ghani*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

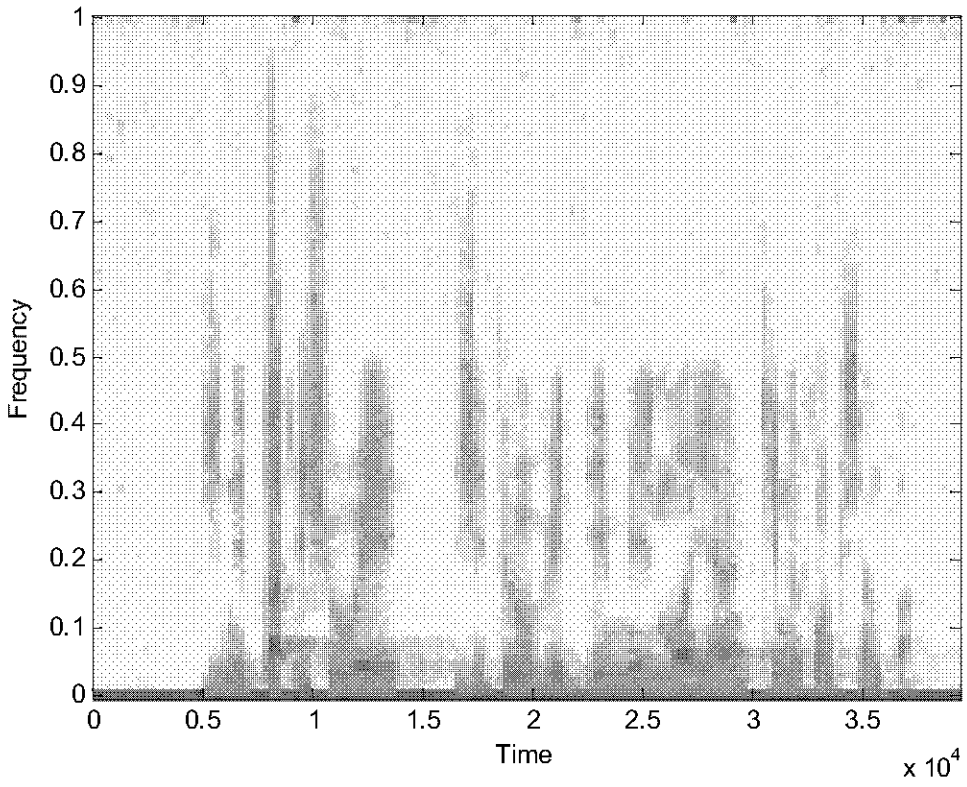

Figure I.26 : *Le spectogramme de* ***Mourad*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

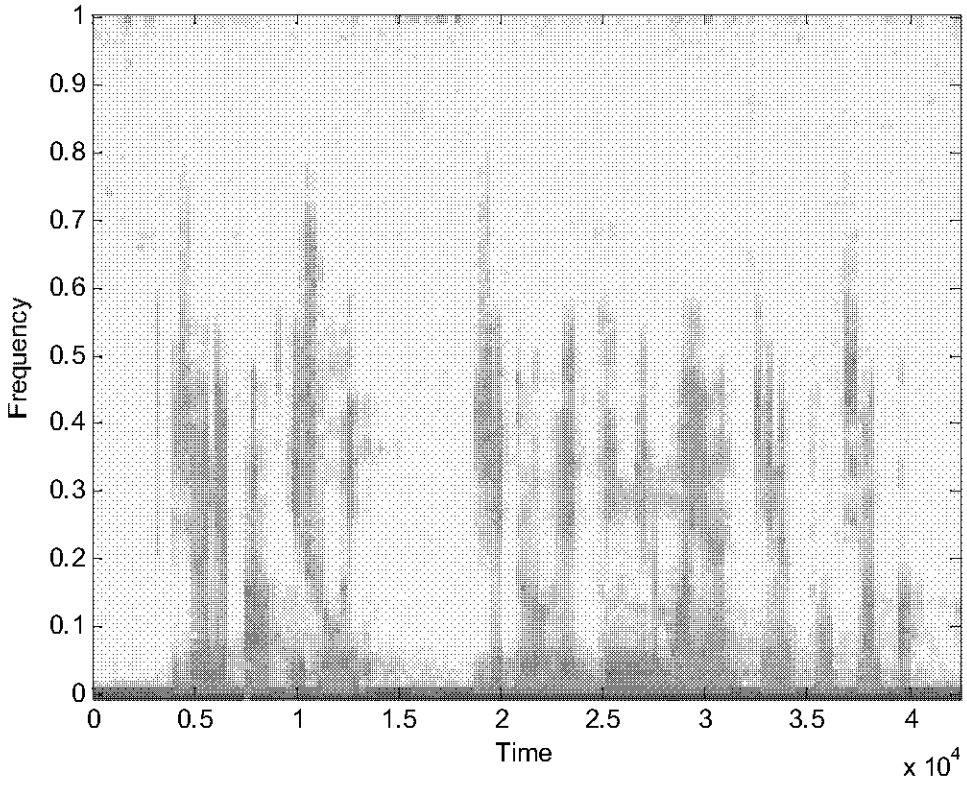

Figure I.27 : *Le spectogramme de* ***Toufik*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

Les spectogrammes précédents sont à bande étroite. Ils sont obtenus avec des fenêtres de pondération Hamming de longue durée (30 ms).

I. 4.5 Coefficient MFCC (Mel Frequency Cepstral Coefficients)

L'extraction de coefficients MFCC est basée sur l'analyse par banc de filtre qui consiste à filtrer le signal par un ensemble de filtres passe bande en générales des filtres triangulaires. Pour simuler le fonctionnement du système auditif humain qui ne présente pas une résolution fréquentielle linéaire à travers le spectre audio, les fréquences centrales (des filtres) sont réparties uniformément sur une échelle perceptive. Plus la fréquence centrale d'un filtre est élevée, plus sa bande passante est large. Les échelles perceptives les plus utilisées sont l'échelle de Mel ou l'échelle de Bark:

$$B_{ark} = 13 Arctg\left(\frac{0.76\ F_{HZ}}{100}\right) + 3.5 Arctg\left(\frac{F_{HZ}}{7500}\right)^2 \quad \text{(I.1)}$$

$$Mel(f) = X \cdot \log_{10}(1 + \frac{f}{Y}) \quad \text{(I.2)}$$

Plusieurs valeurs sont utilisées pour X et Y, mais les valeurs les plus couramment utilisées sont x = 2595 et y = 700. On peut donc réécrire l'équation I.2 :

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700}) \quad \text{(I.3)}$$

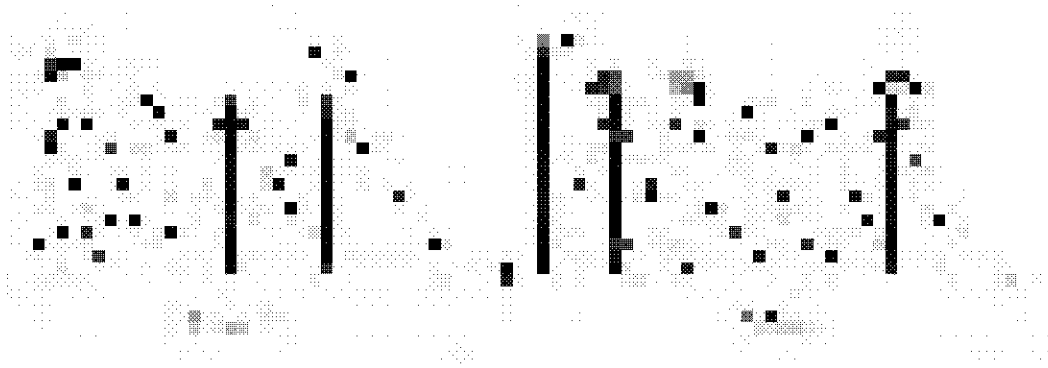

Figure I.28 : *Répartition des filtres triangulaires sur les échelles fréquentielle et Mel.*

L'échelle de Mel est linéaire dans un espace logarithmique et logarithmique dans une échelle linéaire. En effet, dans une échelle logarithmique, les fréquences sont régulièrement espacées d'un facteur de $\frac{1}{700}$ ce qui n'est plus le cas lorsqu'on passe dans une échelle linéaire comme l'indique l'équation (I.3).

Pour des raisons pratique, on considère que cette échelle est linéaire jusqu'à certaine valeur et logarithmique au-dessus de cette valeur (par exemple 1000 Hz comme l'indique Calliope (1989) **[5].**

Le bloc diagramme d'un système d'extraction de MFCC par l'analyse en banc de filtres est montré en Figure I.29. La description qui suit en résume les principales étapes. Pour chaque échantillon de signal *S(n)*, la transformée de Fourier discrète du spectre est obtenue via une transformée de Fourier rapide (FFT).

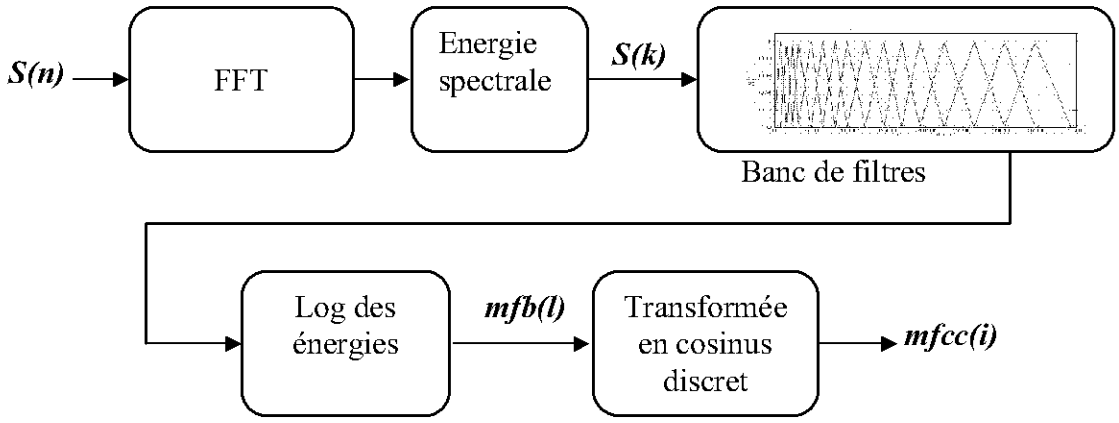

Figure I.29 : *Calcul des coefficients MFCC avec une échelle Mel.*

Une fois la FFT obtenue, le carré de l'amplitude du spectre est déduit. Les portions du spectre contenant les hautes fréquences et le résultat *S(k)*, passent à travers un banc de filtres. Les fréquences centrales de ces filtres triangulaires sont uniformément espacées de 100 Hz selon l'échelle de Mel. Leurs largeurs de bande sont telles que les fréquences de coupure supérieures et inférieures de chaque filtre coïncident avec les fréquences centrales des filtres adjacents, pour des largeurs de

bandes égales sur l'échelle de Mel mais décroissantes sur une échelle linéaire de fréquence. Le nombre Banc de filtres de filtres *NF*, et choisi de façon a couvrir toute la largeur de bande [O, $\frac{f_s}{2}$] (Hz) du signal (ou f_s est la fréquence d'échantillonnage).

Désignons par f_l un filtre, $l \in$ [1, NF], de Fréquence centrale fc_l.

Les fréquences de coupure supérieure et inférieure de f_l, sont respectivement fc_{l-1} et fc_{l+1} avec $fc_0 = 0$ et $fc_l < \frac{f_s}{2} \ \forall \ l$.

Si *S(k)* est la transformée de Fourier discrète du signal de parole en N points, la transformée discrète de Fourier pour chaque filtre triangulaire f_l est donnée par :

$$F_l(k) = \begin{cases} \dfrac{(\frac{k}{N})fs - fc_{l-1}}{fc_l - fc_{l-1}} & L_l \le k \le C_l \\ \dfrac{fc_{l+1} - (\frac{k}{N})fs}{fc_{l+1} - fc_l} & C_l \le k \le U_l \end{cases} \qquad \text{(I.4)}$$

où k$\in$ [0 , $\frac{N}{2}$] ; $C_l = \frac{fc_l}{fs} N$; $U_l = \frac{fc_{l+1}}{fs} N$; $L_l = \frac{fc_{l-1}}{fs_l} N$.

A la sortie de chaque filtre f_l , on retient le logarithme de l'énergie du signal *mfb(l)* défini comme :

$$mfb(l) = \log\left(\frac{1}{A_l} \sum_{A_l}^{U_l} F_l(k) \ S(k) \right) \qquad \text{(I.5)}$$

Où A_l est un facteur de normalisation tenant compte des largeurs de bande variables des différents filtres. A_l est défini comme $A_l = \sum_{k=L_l}^{U_l} F_l(k)$. L'équation (I.5) montre aussi que l'énergie du signal à la sortie du filtre est obtenue par corrélation entre le signal de parole et le filtre dans l'espace de Fourier. On constate également, que la sortie de chaque filtre apporte une contribution élémentaire au spectre, les différentes sorties

peuvent être combinées afin de constituer un vecteur de paramètres utilisables dans un système de reconnaissance.

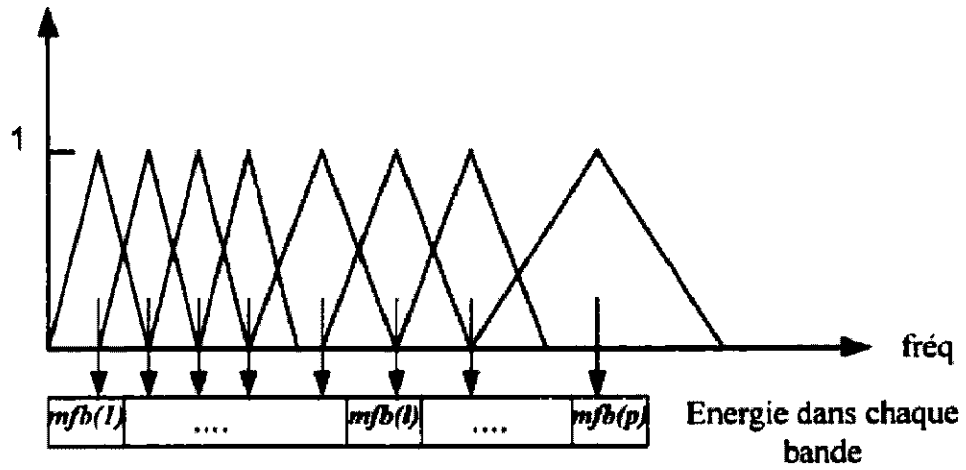

Figure I.30 : Le *banc de filtre en échelle de Mel.*

Finalement, les coefficients cepstraux de l'échelle de Mel, *mfcc(i),* sont déduits de l'équation (II.22) par :

$$mfcc(i) = \sqrt{\frac{2}{NF}} \sum_{l=1}^{NF} mbf(i) \cos\left(i\left(l - \frac{1}{2} \right) \frac{\pi}{NF} \right) \qquad \text{avec : } i = 1,\ldots,\ NF\text{-}1. \quad \text{(I.6)}$$

L'équation (I.6) montre que les *mfcc* sont obtenus en prenant le logarithme de la transformée de Fourier discrète de l'équation (I.5). Soulignons que le coefficient *mfcc(0)* représente l'énergie moyenne dans le signal de parole, *mfcc(1)* permet de faire la détection des sons voisés et/ou non-voisés. En plus d'avoir un grand pouvoir discriminant, les *mfcc* présentent un certain nombre d'avantages supplémentaires. Par exemple, si l'on fait passer un signal audio à travers un canal de transmission on sait bien que, dans le domaine des fréquences, cette action revient à multiplier le spectre du signal avec la fonction de transfert du canal. Or le domaine cepstral, du fait qu'il est logarithmique, ramène cette multiplication à une addition. On peut donc réussir à réduire l'effet du canal de transmission sur le signal de parole.

Les Figures ci-dessous donnent quelques résultats obtenus de l'extraction des cœfficients MFCC.

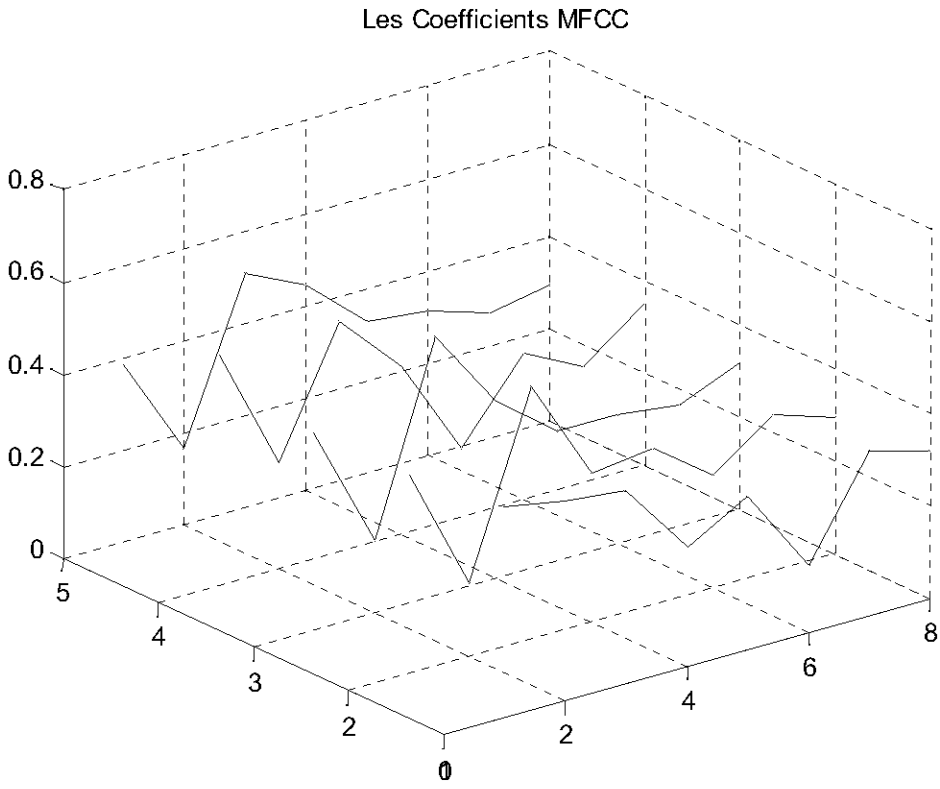

Figure I.31 : *Les 8 coefficients MFCC de la voix de* ***Imane*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

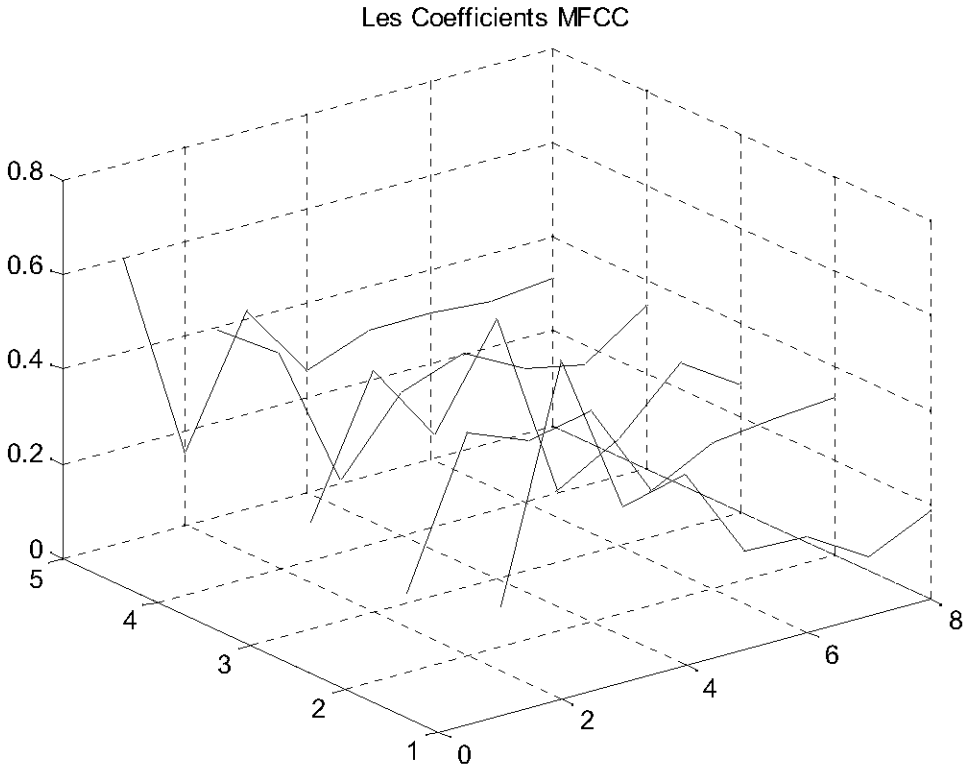

Figure I.32 : *Les 8 coefficients MFCC de la voix de* ***Linda*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

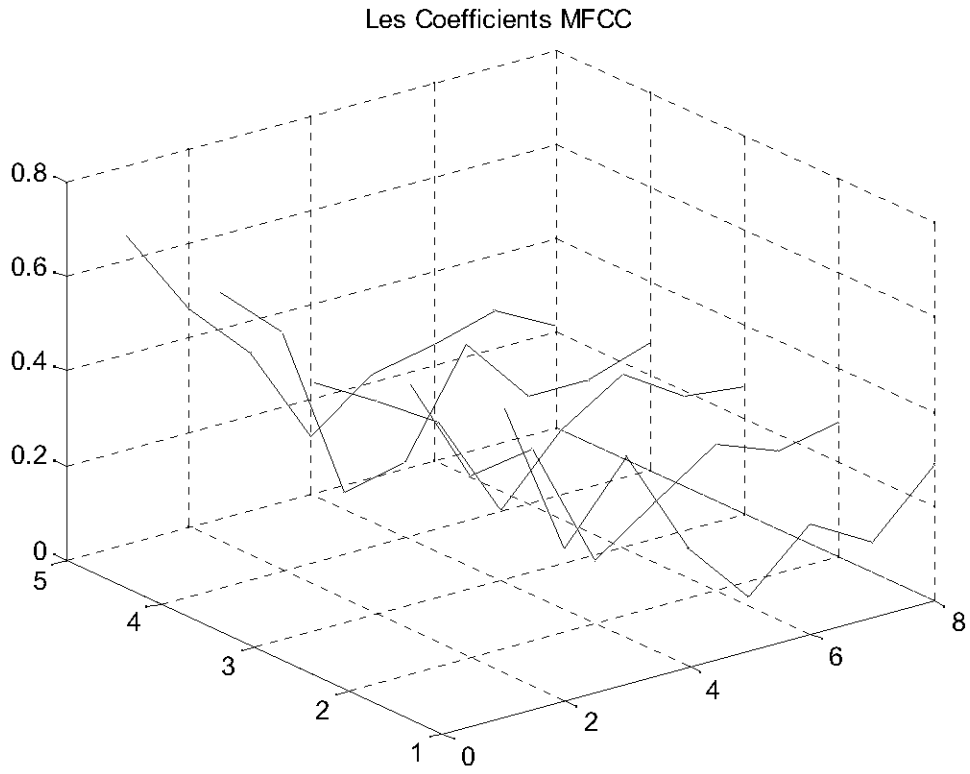

Figure I.33 : *Les 8 coefficients MFCC de la voix de* ***Ouiza*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

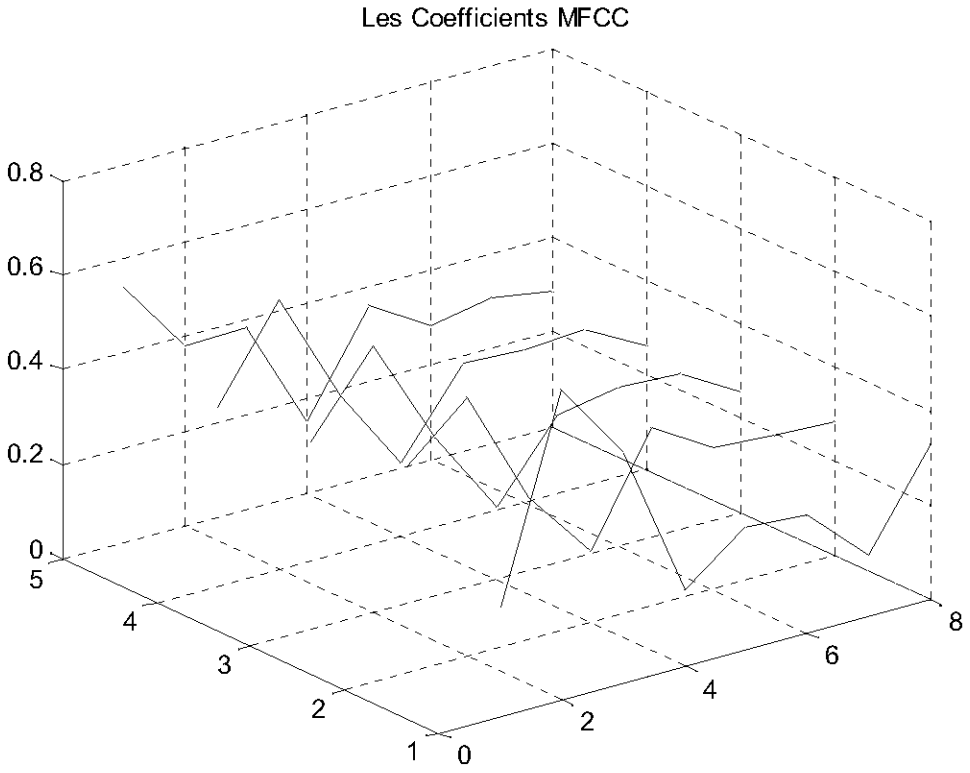

Figure I.34 : *Les 8 coefficients MFCC de la voix de* ***Ghani*** pour la phrase :
" *Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas* ".

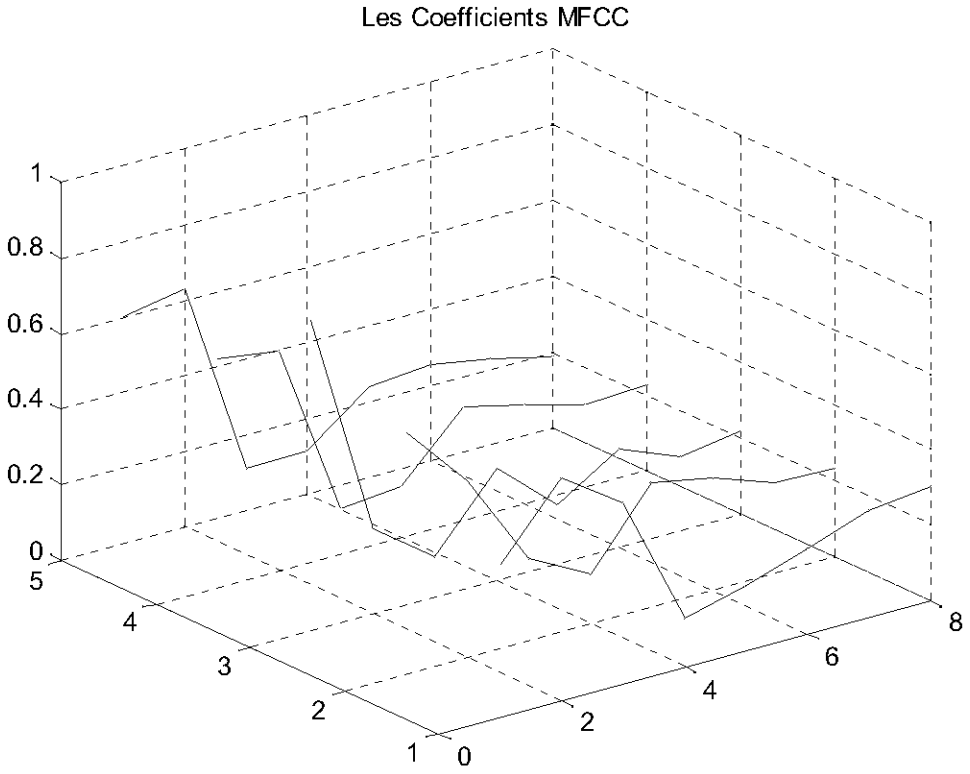

Figure I.35 : *Les 8 coefficients MFCC de la voix de* ***Mourad*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

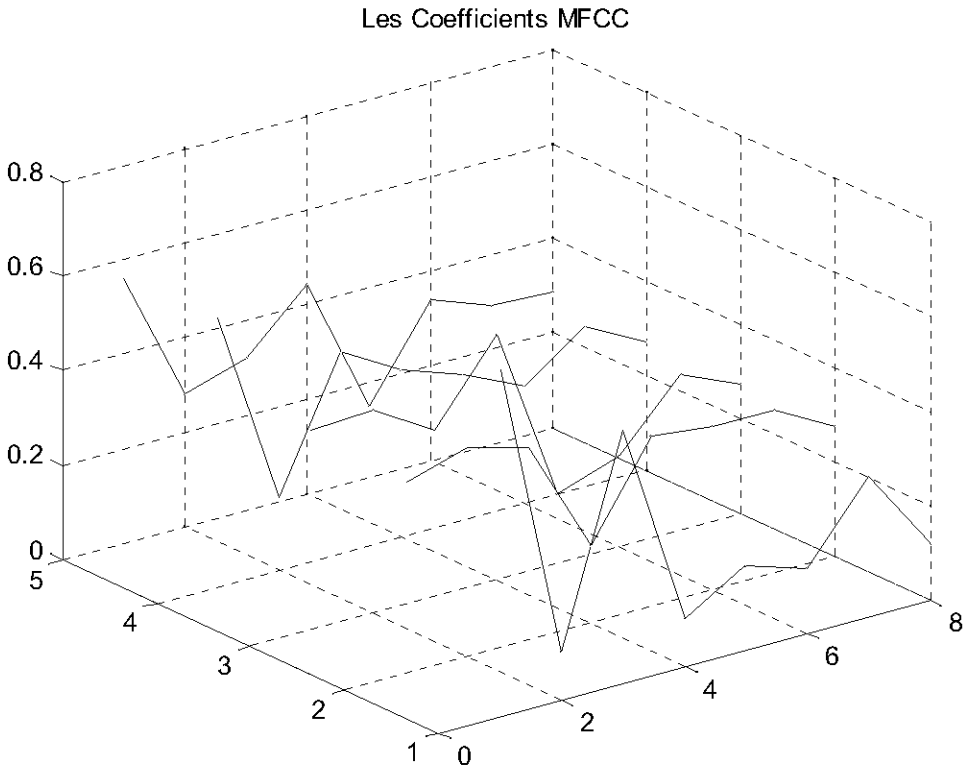

Figure I.36 : *Les 8 coefficients MFCC de la voix de* ***Toufik*** pour la phrase :
" Je n'ai pas échoué, j'ai trouvé 10000 moyens qui ne fonctionnent pas ".

Concernant notre application, nous avons obtenu 8 cœfficients MFCC pour chaque trame (vecteur). Chaque cœfficient est calculé en utilisant 16 filtres.

Finalement, à partir de l'enregistrement (un vecteur de 20001 éléments), nous obtenons une matrice de 60 vecteurs. Chaque vecteur est constitué des huit coefficients MFCC.

Le schéma ci-dessous résume les opérations de traitement appliquées à chaque enregistrement.

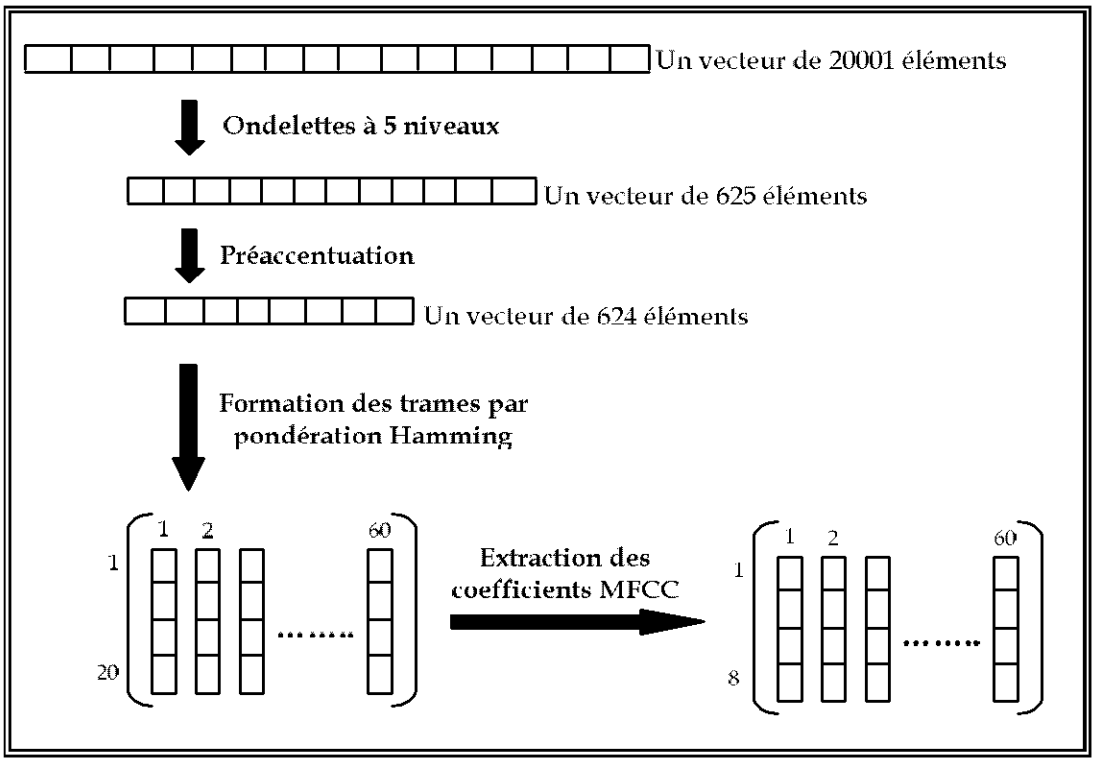

Figure I.37 : *Les différentes étapes de traitement des voix enregistrées.*

Dans ce chapitre, nous avons détaillé les différentes opérations de prétraitements appliquées aux données. Elles se résument comme suit :

- Pondération Hamming;
- Compression et filtrage par ondelettes;
- Extraction des coefficients MFCC.

II.1 INTRODUCTION

Nous avons présenté dans le chapitre précèdent les prétraitements effectués sur notre banque de données. Le but de l'étape précédente est d'extraire l'information essentielle caractérisant un locuteur, et de la présenter sous la forme la plus concentrée possible au classifieur.

Les techniques de classification possibles, utilisées pour la reconnaissance de locuteurs sont nombreuses. Deux méthodes sont présentées dans ce chapitre :

- Les modèles cachés de Markov;
- La Rétropropagation du gradient de l'erreur des réseaux de neurones.

Les vecteurs de données sont divisés en deux ensembles : données d'apprentissage et données de test. Les vecteurs d'apprentissage constituent 75% de l'ensemble de données, les 25% restant forment l'ensemble test. Le choix des vecteurs de test s'effectue de façon aléatoire.

II.2 MÉTHODE DES HMM

Pour mieux comprendre les HMM, il est important de rappeler le contexte de leur utilisation **[8]**.

Les processus, dans le monde réel, produisent généralement des observations de sortie appelées signaux. Ces signaux peuvent être discrets ou continus (échantillons de parole, mesures de température, etc.). La source du signal peut être stationnaire (ses propriétés ne varient pas avec le temps) ou non stationnaire. Le signal peut être pur (provenir strictement d'une source unique) ou corrompu soit par des signaux provenant d'autres sources (bruit) soit par des distorsions de transmission ou carrément des deux. Un problème d'intérêt fondamental est de caractériser ces signaux du monde réel par des modèles de signaux. Il y a plusieurs raisons qui concourent à cela. Tout d'abord, un modèle de signal peut fournir la base d'une description théorique d'un système de traitement de signal qui peut être utilisé pour traiter le signal de manière à produire la sortie voulue. Par exemple, si nous voulons extraire un signal de parole corrompu par du bruit et une distorsion de transmission, nous pouvons nous servir du modèle théorique de signal pour

concevoir un système qui va éliminer le bruit et la distorsion et ce de façon optimale. La deuxième raison qui nous emmène à considérer des modèles de signaux est qu'ils nous permettent de potentiellement comprendre les sources de signal (processus de production des signaux dans le monde réel) sans avoir besoin de disposer de ces sources. C'est une propriété très importante lorsque le coût d'obtention de la source réelle est très élevé. Dans ce cas, avec un bon modèle de signal, nous pouvons en étudier la source grâce à des simulations. Cette propriété est très importante car elle indique que la connaissance du signal infère sur la connaissance de la source. On peut alors distinguer des sources différentes par la seule connaissance des signaux qu'elles émettent. Il y a plusieurs choix de modèles possibles afin de caractériser les propriétés d'un signal. On peut dire qu'il existe deux types de modèles de signaux : la classe des modèles déterministes et la classe des modèles statistiques. Les modèles déterministes exploitent quelques propriétés spécifiques du signal ; par exemple, on peut considérer que le signal est une onde sinusoïdale ou une somme d'exponentielles. Dans ce cas, il est facile de déterminer le modèle du signal : il suffit, pour cela, de déterminer les valeurs des paramètres du modèle du signal (amplitude, fréquence, phase de l'onde sinusoïdale, amplitude et taux des exponentielles, etc.). La deuxième classe des modèles de signaux est la classe des modèles statistiques dans laquelle l'on essaie de caractériser les propriétés statistiques du signal. Parmi ces modèles, on retrouve les modèles de Markov. Dans les modèles de signaux statistiques, l'hypothèse de base est que le signal peut être parfaitement caractérise comme un processus aléatoire paramétrique et que les paramètres du processus stochastique peuvent être estimés de manière précise et bien définie à partir du signal.

Nous allons d'abord revoir la théorie des chaînes de Markov et ensuite étendre les idées à la classe des HMM en utilisant quelques exemples classiques. Nous allons ensuite porter notre attention sur les trois problèmes fondamentaux de la conception des HMM. Ces problèmes sont: 1'évaluation des probabilités d'une séquence d'observation étant donné un modèle spécifique de HMM, la détermination de la meilleure séquence des états du modèle; l'ajustement des paramètres du modèle de telle sorte qu'ils tiennent comptent du signal observé.

II.2.1 Présentation mathématiques des processus stochastiques

Dans sa définition la plus simple, un processus stochastique est une suite finie de variables aléatoires dont chacune à un nombre fini de résultats possibles avec des probabilités données. Les variables aléatoires $\{X(t),\ t \in T_s\}$ constituant le processus stochastique, sont donc définies dans un espace probabilisé et indicées par le paramètre t. Le paramètre t appartient à un ensemble d'indices T_s, cet ensemble est lui-même un sous-ensemble de $(-\infty,+\infty)$). En général, T_s, représente un ensemble de temps. $X(t)$ est appelé l'observation au temps t, c'est une grandeur physique. Les valeurs prises par la variable aléatoire $X(t) = S_i$, $i \in [1,2, ..., T]$ sont appelées les états du processus. L'ensemble des états constitue l'espace des états du processus. Une évolution du système est donc une suite de transitions d'états, a partir d'un état de départ $X(1)=S_1$:

$$S_1 \to S_2 \to S_3 \to \ldots \to S_T$$

La loi d'évolution du système consiste à avoir les chaînes de transitions S_1 , S_2, $S_3,\ldots, S_m$ $\forall m \le T$ avec la probabilité $P(S_1 , S_2,\ldots, S_T)$.

Définissons deux événements E et F . Par définition, la probabilité conditionnelle de réaliser F sachant que E a été réalisé est donnée par $P(F|E) = \dfrac{P(E,F)}{P(E)}$, où $P(E, F)$ est la probabilité de réaliser à la fois F et E. Si maintenant on pose que $E = \{S_1 , S_2,\ldots, S_T\}$ et $F = (S_T)$ on a :

$$\begin{aligned} P(S_1 , S_2,\ldots, S_T) &= P(S_1 , S_2,\ldots, S_{T-1}) \times P(S_T \mid S_1 , S_2,\ldots, S_{T-1}) \\ &= P(S_1 , S_2,\ldots, S_{T-2}) \times P(S_{T-1} \mid S_1 , S_2,\ldots, S_{T-2}) \times P(S_T \mid S_1 , S_2,\ldots, S_{T-1}) \\ &= P(S_1) \times P(S_2 \mid S_1) \times P (S_3 \mid S_2, S_1) \times \ldots \times P(S_T \mid S_1 , S_2,\ldots, S_{T-1}) \end{aligned}$$

Ainsi pour calculer $P(S_1 , S_2,\ldots, S_T)$, on évolue de proche en proche jusqu'à $P(S_1$).En d'autres termes, il suffit de se donner la probabilité initiale $P(S_1)$ et les probabilités des états conditionnés par les évolutions antérieures. La situation générale est celle où la loi de probabilité des états à un certain instant dépend de la totalité de l'histoire du système. On dira que celui-ci garde la mémoire de son passé.

II.2.2 Les propriétés de Markov

Un processus stochastique vérifie la propriété de Markov si :

$$\forall\, t,\ P(q_t = s_i \mid q_{t-1} = s_j\ ,\ q_{t-2} = s_k\ , \ldots) = P(q_t = s_i \mid q_{t-1} = s_j\ ) \qquad \text{(II.1)}$$

Ainsi formulée la propriété de Markov stipule que la probabilité d'un état donné ne dépend que l'état qui le précède immédiatement et d'aucun autre état, ce qui nous permet d'écrire :

$$P(q_1, \ldots\ , q_T\ ) = P(q_1) \times P(q_2 \mid q_1) \times P(q_3 \mid q_2) \times \ldots \times P(q_T \mid q_{T-1}) \qquad \text{(II.2)}$$

L'évolution du système est entièrement déterminée par la probabilité initiale et par les probabilités de transition successives. Un processus ainsi défini est appelé processus de Markov du premier ordre. Un processus de Markov est dit stationnaire si :

$$\forall\, t,k,\quad P(q_t = s_i \mid q_{t-1} = s_j\ ) = P(q_{t+k} = s_i \mid q_{t+k-1} = s_j\ ) \qquad \text{(II.3)}$$

L'équation (II.3) signifie qu'un processus markovien stationnaire demeure invariant pour tout changement de l'origine des temps : ce qui va être le cas dans notre application. On définit dans ce cas une matrice de probabilités de transition $A = \{a_{ij}\}$ telle que :

$$a_{ij} = P\ (q_t = s_i \mid q_{t-1} = s_j\ ) \qquad 1 \le i \le N \quad , \quad 1 \le j \le N \qquad \text{(II.4)}$$

avec :

$$\forall\, t,j \qquad a_{ij} \ge 0 \qquad \text{et} \qquad \forall i \sum_{j=1}^{N} a_{ij} = 1 \qquad \text{(II.5)}$$

Toute matrice carrée dont les éléments vérifient l'équation (II.5) est appelée matrice stochastique. La matrice des probabilités de transition est donc une matrice stochastique.

II.2.3 Exemple d'application

Nous allons maintenant présenter un exemple. Dans une station météo, on veut modéliser l'évolution de la météo à partir d'observations effectuées quotidiennement. Pour cela on utilise une chaîne de Markov à trois états. Chaque état représente un aspect de la météo :

État 1 → pluie = S_1.

État 2 → nuage = S_2.

État 3 → soleil = S_3.

On considère que le temps d'un jour t est caractérisé par un seul de ces trois états, et que la matrice des probabilités de transition A est :

$$A = a_{ij} = \begin{bmatrix} 0.8 & 0.15 & 0.05 \\ 0.7 & 0.2 & 0.1 \\ 0.5 & 0.3 & 0.2 \end{bmatrix}$$

Supposons que le temps au jour un (t = 1) soit "soleil" (état 3). Nous pouvons nous poser la question : quelle est la probabilité que le temps des sept jours à venir soit " soleil - soleil - pluie - pluie - soleil - nuage - soleil " connaissant le modèle ? De manière plus formelle nous venons de définir une séquence d'observations $O=\{S_3 , S_3 , S_1 , S_1 , S_3 , S_2 , S_3 \}$ correspondant aux jours $t=1,2,3, ..., 7$. Nous voulons alors déterminer la probabilité de cette séquence connaissant le modèle. Pour cela nous utilisons la propriété de Markov (équations II.1 et II.2), elle nous permet de calculer cette valeur de la manière suivante :

$$\begin{aligned} P(o \mid Modèle) &= P[S_3 , S_3 , S_3 , S_1 , S_1 , S_3 , S_2 , S_3 \mid Modèle] \\ &= P[S_3] \times P[S_3 \mid S_3] \times P[S_3 \mid S_3] \times P[S_1 \mid S_3] \times P[S_1 \mid S_1] \times P[S_3 \mid S_1] \\ &\quad \times P[S_2 \mid S_3] \times P[S_3 \mid S_2] \\ &= \pi_3 \times a_{33} \times a_{33} \times a_{31} \times a_{11} \times a_{13} \times a_{32} \times a_{23} \\ &= 1 \times 0.2 \times 0.2 \times 0.5 \times 0.8 \times 0.05 \times 0.3 \times 0.1 \\ &= 0.24 \times 10^{-4} \end{aligned}$$

où nous avons utilise π_i pour spécifier les probabilités de l'état initial :

$$\pi_i = P[q_1 = S_i] \qquad 1 \le i \le N \qquad \text{(II.6)}$$

De même s'il fait "soleil" aujourd'hui, quelle va être la distribution du temps dans trois jours ? Pour répondre à cette question, nous disons que l'état de départ est caractérisé par le vecteur $P^{(0)} = (0, 0, 1)$ car nous partons de S_3. Ainsi, la distribution du temps sera :

$$P^{(1)} = P^{(0)} A = (0, 0, 1) \begin{bmatrix} 0.8 & 0.15 & 0.05 \\ 0.7 & 0.2 & 0.1 \\ 0.5 & 0.3 & 0.2 \end{bmatrix} = (0.5, 0.3, 0.2)$$

$$P^{(2)} = P^{(1)} A = (0.71, 0.195, 0.095)$$

$$P^{(3)} = P^{(2)} A = (0.752, 0.174, 0.074)$$

On conclut donc que dans trois jours on aura 75,2% de chance de pluie, 17,4% de chance qu'il soit nuageux et enfin 7,4% de chance que le temps soit ensoleillé.

II.2.4 Concept des HMM

Le modèle de Markov défini précédemment n'est pas d'un grand intérêt en reconnaissance de la parole d'une part et par extension en reconnaissance du locuteur. Il nous faut donc étendre cette notion au concept de HMM. Afin de mieux saisir ce concept, nous allons introduire deux exemples **[8]**. Puis, nous établirons une définition plus formelle.

Dans le premier exemple, qui est une variante du jeu de pile ou face, on va supposer que vous êtes assis dans un local, ce dernier est divisé en deux par un mur de béton : vous ne pouvez pas voir ce qui se passe de l'autre côté du mur. Dans l'autre portion du local, il y a une autre personne qui joue à pile ou face avec une pièce de monnaie (ou plusieurs pièces). L'autre personne ne vous dit pas ce qu'elle fait exactement. Elle ne vous dit pas comment elle procède (une ou plusieurs pièces) mais vous communique seulement le résultat d'un lancer. Donc une séquence cachée

d'expériences (pile ou face) est faite, produisant comme résultat une séquence d'observations constituée d'une série de pile ou face. Une séquence typique est:

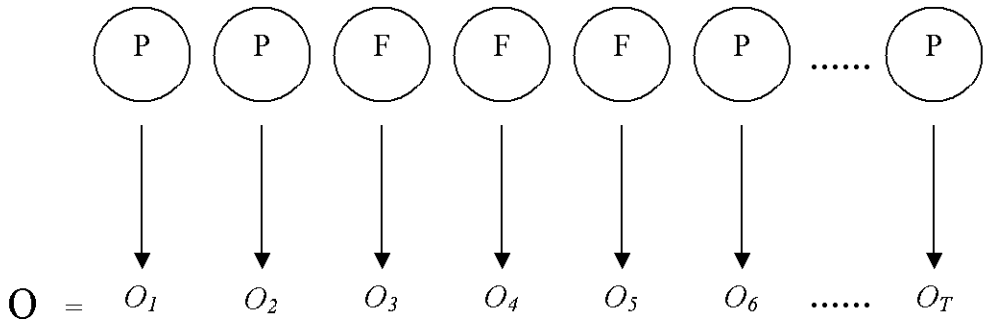

Figure II.1 : *Séquence possible avec P pour pile et F pour face.*

Étant donné le scénario ci-dessus, le problème intéressant qu'il souligne est celui de la construction d'un HMM qui permette de générer la séquence observée de pile et de face (Figure II.1). Pour ce faire, il faudrait d'abord définir les états du modèle. En d'autres mots, il s'agit de décider à quoi correspondent les états du modèle et ensuite le nombre d'états présents dans le modèle. Un choix possible consiste à dire qu'une pièce et une seule a été successivement lancée afin d'obtenir la séquence observée. Dans ce cas, nous pouvons utiliser un modèle à deux états ou chaque état correspond à un côté de la pièce (Figure II.2). Une deuxième forme de HMM pour expliquer la séquence observée est donnée par la figure (II.3). Dans ce cas, il y a deux états et chaque état correspond à une pièce. Chaque pièce est caractérisée par une distribution de probabilité de pile et face et les transitions entre les états sont caractérisées par une matrice de transition d'état. Le mécanisme physique permettant les transitions entre états peut lui-même être représenté par un processus de type pile ou face, ou de tout autre événement probabiliste.

Une troisième forme de HMM pour expliquer la séquence de pièces observée est donnée à la figure (II.4). Ce modèle correspond à l'usage de trois pièces et le choix d'une des pièces est aléatoire. Nous avons ici deux processus stochastiques : le tirage aléatoire d'un des trois modèles et le tirage d'un côté de la pièce.

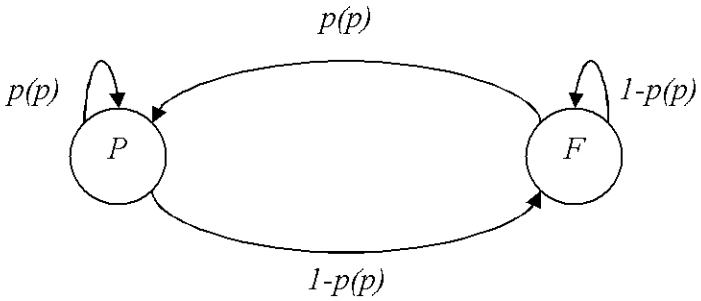

Figure II.2 : *Chaîne de Markov à un état. p(p): est la probabilité de "pile ". 1-p(p): est la probabilité de "face ".*

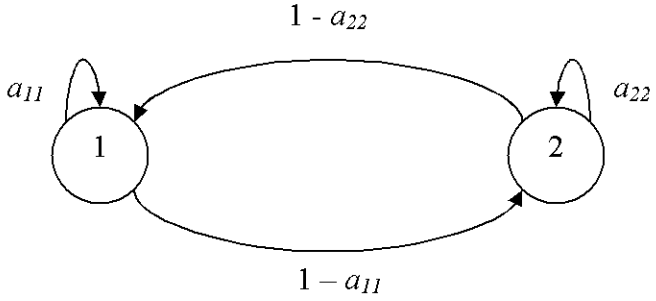

Figure II.3 : *Chaîne de Markov à deux états mais avec deux pièces différentes. a_{ii} : est la probabilité de demeurer à l'état i.*

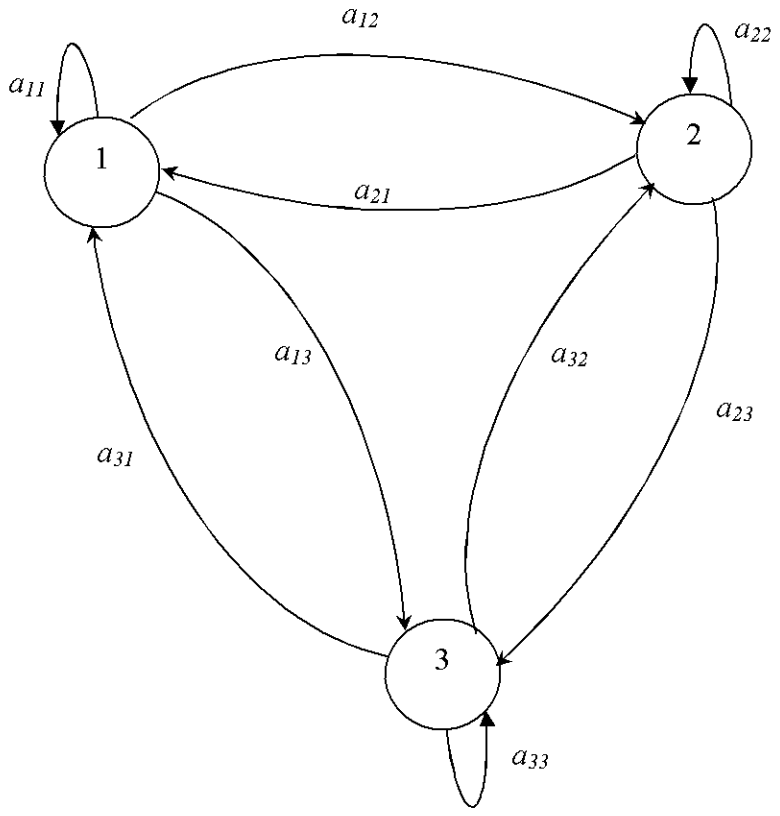

Figure II.4 : *Chaîne de Markov à trois états.*

Étant donnés les trois modèles précédents qui rendent tous compte de la même séquence d'observations, l'une des questions que l'on pourrait se poser serait de savoir lequel des trois génère le mieux la séquence d'observations. La réponse à cette question dépend de l'utilisation, raisonnablement le modèle de la figure (II.4) peut être utilisé afin de modéliser un éventail plus large de variantes du jeu de pile ou face mais pour des raisons pratiques le modèle de la figure (II.2) est largement suffisant ici.

Nous allons étendre le concept des HMM avec la dernière illustration suivante : le cas des traditionnelles boules et urnes (figure II.5). Supposons qu'il y a N grandes urnes (urne 1, urne 2,..., urne N) en verre dans une salle, à l'intérieur de chaque urne, il y a un grand nombre de boules colorées (boules rouges, blanches, etc.). Il y a au total M couleurs différentes. Le processus physique permettant d'obtenir des suites de boules ou observations est le suivant. Une personne est dans la salle, elle choisit aléatoirement (sur la base d'un processus aléatoire) une urne initiale. De cette urne, une balle est choisie aléatoirement et sa couleur est enregistrée comme une observation. La balle est ensuite remise dans l'urne dont elle a été retirée. Une nouvelle urne est sélectionnée en accord avec un processus de sélection aléatoire associé avec l'urne courante, et le processus de sélection de la balle se répète. Et ainsi de suite. On génère ainsi une séquence finie de boules de différentes couleurs, cette séquence constitue la sortie observable du HMM. On a ici deux processus stochastiques le premier est le choix d'une urne et le second est te tirage d'une boule. Le plus simple HMM capable de décrire le processus précédent est celui dans lequel chaque état correspond à une urne et pour lequel la probabilité d'une couleur est définie pour chaque état. Le choix d'une urne est dicté par la matrice de transition d'états du HMM.

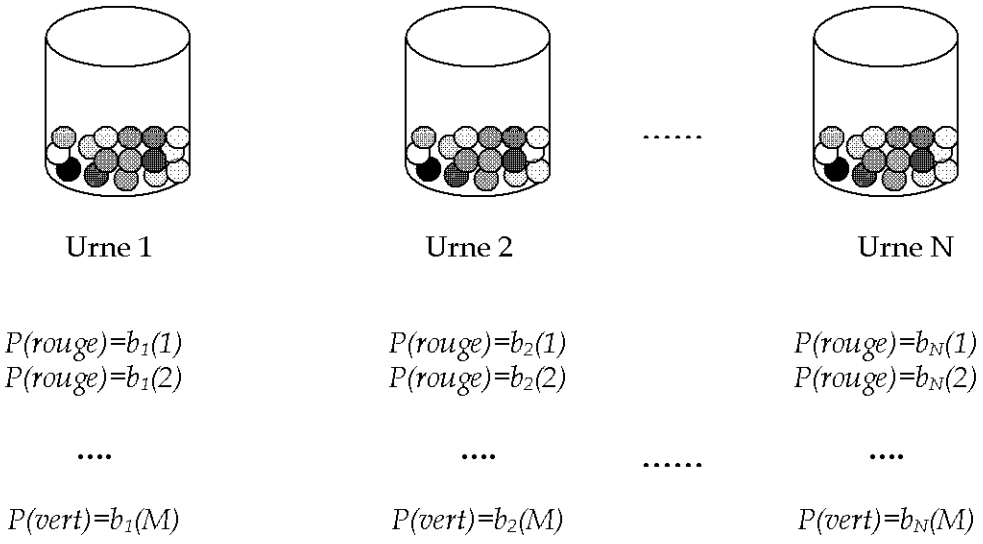

Figure II.5 : *Tirage aléatoire d'une urne et d'une boule.*

II.2.5 Définition formelle d'un HMM

Un modèle de Markov caché est un modèle doublement stochastique dont une composante est une chaîne de Markov non observable, mais qui peut être observée à travers un autre ensemble de processus stochastiques qui produit une suite de symboles observables. La chaîne cachée correspond a la suite d'états q_1 , q_2 , q_3 ,…, q_T , notée $Q\ (1 : T\ )$ où les q_i , prennent leur valeur au cours du temps parmi l'ensemble des N états { S_1 , S_2,…, S_N } du modèle. La suite observable correspond à la suite des observations O_1 , O_2 , O_3 ,…., O_T notée $O\ (\ 1: T\ )$, où les O_i sont aussi fonction du temps et se réalisent parmi un ensemble de M symboles observables { V_1 , V_2 ,… V_M}.

De tels modèles permettent de représenter des problèmes où les états ne sont pas directement observables, donc cachés. Cependant ces états peuvent être estimés à travers des processus stochastiques, ceux qui produisent les séquences

d'observations. Nous pouvons, avec de tels modèles, représenter des processus plus complexes qu'avec les simples processus de Markov (exemple de la météo présenté plus haut). Pour cela il nous faut bien sûr des connaissances supplémentaires comme M le nombre d'observations et B la matrice des probabilités d'observation.

Un modèle de Markov caché est entièrement défini par un ensemble de cinq paramètres (N, M, A, B, π) :

- N: le nombre d'états du modèle. Bien que les états soient cachés, pour beaucoup d'applications ils ont une signification physique comme un côté d'une pièce de monnaie ou une urne pour se référer aux exemples employés plus haut. L'ensemble des états est généralement noté S, $S = \{ S_1, S_2, \ldots, S_N \}$. De plus l'état à l'instant t est noté q_t, $q_t \in S$.

- M: le nombre de symboles observables pour chaque état. Ces symboles correspondent aux sorties physiques du système à modéliser (pile ou face pour le lancer d'une pièce ou la couleur d'une boule tirée). L'ensemble des symboles d'observations est noté V, $V = \{ v_1, v_2, \ldots v_M \}$.

- A : la matrice des probabilités de transition, $A = \{a_{ij}\}$, où a, représente la probabilité que le modèle évolue de l'état i vers l'état j : $a_{ij} = P[q_{t+1} = S_j \mid q_t = S_i]$ avec $1 \le i, j \le N$ *et* $\forall t$

- B : la distribution des probabilités d'observation des symboles, B = { $(b_j(k))$ }, où $b_j(k)$ représente la probabilité que l'on observe le symbole v_k alors que le modèle se trouve dans l'état j : $b_j(k) = P[O_i = v_k \mid q_i = S_j]$ avec $1 \le j \le N$ *et* $1 \le k \le M$

- π : la distribution des probabilités initiales, $\pi = \{ \pi_i \}$, où π_i représente la probabilité que l'état de départ du modèle soit l'état i : $\pi_i = P[q_1 = S_i]$ avec $1 \le i \le N$.

Nous pouvons conclure que la spécification complète d'un modèle de Markov caché requiert la définition de deux paramètres N et M, la définition des vecteurs d'observations et les distributions des probabilités A, B et π . La notation d'un modèle entièrement défini est généralement la suivante : $\lambda = (A , B , \pi)$. Une fois tous les paramètres spécifiés, nous pouvons utiliser le modèle de Markov caché comme un générateur de séquences d'observations : $O = O_1 O_2 O_3 \dots O_T$

La procédure permettant de générer une telle séquence est la suivante :

- 1. Choisir un état initial à l'instant $t = 1$, $q_1 = S_i$ selon la distribution de l'état initial π_i .
- 2. Choisir O_t selon la distribution de probabilité de l'observation dans l'état S_i , c'est-à dire $b_j(O_t)$.
- 3. Si $t < T$ alors passer à un état $q_{t+1} = S_j$ avec la distribution de probabilité de transition d'état pour l'état S_i, c'est-à-dire a_{ij} .
- 4. Incrémenter t. Si $t \leq T$ retourner à l'étape 2, sinon arrêter.

Comme nous venons de le voir dans cette procédure, l'utilisation d'un modèle de Markov caché passe par la définition de plusieurs distributions de probabilité. Pour qu'un modèle reflète correctement le phénomène que nous voulons modéliser, il faut ajuster de manière précise ces différentes distributions.

II.2.6 Les problèmes pratiques à résoudre et leurs solutions

En fait, trois problèmes fondamentaux sont à résoudre pour pouvoir utiliser efficacement un modèle de Markov sur des applications réelles. Dans un premier temps nous énoncerons ces problèmes. Par la suite nous décrirons les algorithmes utilisés afin de résoudre ces problèmes.

- Problème 1 : évaluation du modèle. Étant donnée une suite d'observations $O = O_1 O_2 \dots O_T$ et un modèle $\lambda = (A , B , \pi)$, comment peut-on calculer efficacement la probabilité de la suite d'observations connaissant le modèle $P(O | \lambda)$?

- Problème 2 : estimation de la séquence d'états cachés. Étant donnée une séquence d'observations $O = O_1 O_2 \dots O_T$ et un modèle $\lambda = (A , B , \pi)$,comment peut-on choisir une séquence d'états $Q = q_1, q_2, q_3, \dots, q_T$ selon un critère convenable (la séquence d'états qui «explique»le mieux la séquence d'observations) ?

- **Problème 3** : apprentissage. Comment peut-on ajuster les paramètres du modèle λ = (A , B , π) pour maximiser P(O | λ) ?

Nous allons maintenant présenter des méthodes pour trouver des solutions aux trois problèmes que nous venons d'énoncer.

II.2.6.1 L'évaluation du modèle

Le problème consiste a évaluer le modèle afin de choisir parmi plusieurs celui qui génère le mieux la suite d'observations. IL existe plusieurs techniques pour résoudre ce problème : la méthode d'évaluation directe, la procédure de recherche avant-arrière et l'algorithme de Viterbi.

a) L'évaluation directe

La probabilité P(O |λ) d'une suite d'observations O, sachant qu'un modèle λ est donné, est la somme sur tous les chemins d'états Q possibles, des probabilités conjointes de O et de Q par rapport à ce modèle :

$$P(O|\lambda) = \sum_{Q} P(O|Q,\lambda) = \sum_{Q} P(O|Q,\lambda)\ P(Q|\lambda) \qquad \text{(II.7)}$$

où $Q = q_1q_2q_3...q_T$, $q_T = S_i\ 1 \le i \le N$ et T est le nombre d'observations de la séquence. La probabilité de la séquence d'observations O pour la suite d'états Q s'écrit :

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_t,\lambda) \qquad \text{(II.8)}$$

En considérant l'indépendance des observations, nous pouvons écrire cette probabilité de la manière suivante :

$$P(O|Q,\lambda) = b_{q_1}(O_1)b_{q2}(O_2)b_{q3}(O_3)...b_{qT}(O_T) \qquad \text{(II.9)}$$

La probabilité de la séquence d'état Q connaissant le modèle, peut être écrite de la manière suivante :

$$P(Q|\lambda) = \pi_{q1}a_{q1q2}a_{q2q3}...a_{qT-1qT} \qquad \text{(II.10)}$$

Nous obtenons donc la probabilité de la séquence d'observations connaissant le modèle :

$$P(Q|\lambda) = \sum_{Q} \pi_{q1}\ b_{q1}(O_1)a_{q1q2}\ b_{q2}(O_2)a_{q2q3}\ b_{q3}(O_3)a_{q1q2}.....b_{qT-1}(O_{T-1})a_{qT-1qT}\ b_{qT}(O_T) \quad \text{(II.11)}$$

Le calcul de cette probabilité nécessite $2TN^T$. En effet, a chaque instant t = 1,2,3,…,T il y a N étais possibles qui peuvent être atteints (c'est-à-dire N^T séquences d'états possibles) et pour chaque séquence d'états il y a $2T$ calculs à effectuer. Même pour de petites valeurs de N et T le nombre d'opérations devient vite très important (N=5 et T=100 conduisent à 10^{72} opérations). Cette méthode n'est pas vraiment envisageable. Nous allons donc trouver une autre technique de calcul : la procédure de recherche avant-arrière.

b) La procédure de recherche avant-arrière (forward-backward)

Dans cette approche, on considère que l'observation peut se faire en deux étapes : d'abord l'émission de la suite d'observations $O_1\ O_2\ O_3 \ldots\ O_t$ et la réalisation de l'état S_i au temps t, puis l'émission de la suite d'observations $O_{t+1}\ O_{t+2} \ldots\ O_T$ en partant de l'état S_i au temps t. Dans ce cas la probabilité de la séquence d'observations connaissant le modèle peut s'écrire :

$$P(O|\lambda) = \sum_{i} \alpha_t(i)\beta_t(i) \quad \text{(II.12)}$$

où $a_t(i)$ est la probabilité d'émettre la suite d'observations $O_1\ O_2\ O_3 \ldots\ O_t$ et d'aboutir à S_i à l'instant t connaissant le modèle, et $\beta_t(i)$ est la probabilité d'émettre la suite d'observations $O_{t+1}\ O_{t+2} \ldots\ O_T$ en partant de l'état S_i à l'instant t, connaissant le modèle.

$$\alpha_t(t) = P(O_1O_2O_3..O_t, q_t = S_i|\lambda) \qquad 1 \le i \le N \ , \ 1 \le t \le T \quad \text{(II.13)}$$

$$\beta_t(t) = P(O_{t+1}O_{t+2}O_{t+3}..O_{Tt}, q_t = S_i|\lambda) \qquad 1 \le i \le N \ , \ 1 \le t \le T \quad \text{(II.14)}$$

L'algorithme de recherche avant (forward) se présente de la manière suivante :

- **1. Initialisation : t = 1**

$$\alpha_t(t) = \pi_i b_1(O_1) \qquad i = 1,2,3,\ldots,N \quad \text{(II.15)}$$

Cette étape initialise la probabilité avant (forward). C'est la probabilité conjointe de l'état S_i , i =1, 2,3,…, N, et de l'observation initiale.

- **2. Induction**

$$\alpha_t(j) = \left[\sum_{j=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(O_t) \qquad j=1,2,3,..,N \;\; ; \;\; t=2,3,4,\ldots,T \qquad \text{(II.16)}$$

Cette étape montre comment l'état S_i , peut être visité au temps $t+1$ à partir de N états possibles S_i , $1 \le i \le N$, au temps t.

- **3. Terminaison**

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \qquad \text{(II.17)}$$

Pour calculer la probabilité de la séquence d'observations par cette méthode, il faut effectuer N^2T opérations. En prenant les mêmes valeurs pour N et T que précédemment, nous obtenons un total de 3000 opérations, ce qui est bien moins que les 10^{72} de la méthode directe.

Nous présentons maintenant l'algorithme de recherche arrière (backward) :

- **1. Initialisation, t = T**

$$\beta_t(i) = 1 \qquad i = 1,2,3,\ldots,N \qquad \text{(II.18)}$$

Cette étape définit arbitrairement $\beta_t(i) = 1$, pour tous les états i.

- **2. Induction**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad i = 1,2,3,\ldots,N \; ; \;\; t = T\text{-}1, T\text{-}2,\ldots,1 \qquad \text{(II.19)}$$

Cette étape montre que pour être dans l'état S_i au temps t, et pour tenir compte de la suite d'observations de $t+1$ à T, nous devons considérer tous les états possibles S_j à l'instant $t+1$ et tenir compte des transitions de l'état S_i à l'état S_j (les termes a_{ij}) ainsi que des observations émises à l'état j (les termes $b_j,(O_{t+1})$), puis considérer la suite d'observations partielle restante à partir de l'état j (les termes $\beta_{t+1}(j)$).

Le calcul de la probabilité $P(O \mid \lambda)$ par cette méthode nécessite $N^2 T$ opérations. Les deux variables $a_t(j)$ et $\beta_t(j)$ peuvent être utilisées pour calculer $P(O \mid \lambda)$ à chaque instant t, avec $1 \le t \le T$ de la manière suivante :

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i) \qquad \text{(II.20)}$$

$$P(O|\lambda) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(i) \qquad \text{(II.21)}$$

Cette formule sera utilisée pour résoudre le problème 3, celui de l'apprentissage.

c) L'algorithme de Viterbi

Cette approche est basée sur une technique de programmation dynamique. L'algorithme en lui-même sera développé plus loin, lors de la résolution du second problème, celui de l'estimation de la suite d'états cachés. L'algorithme nous permettant d'utiliser cette technique pour résoudre le problème de l'évaluation du modèle est le suivant :

- 1-Initialisation, $t = 1$
- 2- Si q_1 est connu a priori ($q_1 = S_i$) , alors $\delta_1(i) = O$
- 3- Sinon $\delta_1(i) = \pi_i b_t(O_1)$; $i = 1,2,3,...,N$ (II.22)
- 4- Induction

$$\delta_t(j) = \underset{1 \le i \le N}{Max}\left[\delta_{t-1}(i)a_{ij}\right]b_j(O_t) \quad \text{(II.23)}$$

- 5- Terminaison

$$P(O|\lambda) = \sum_{t=1}^{N} \delta_T(i) \quad \text{(II.24)}$$

Ceci met fin aux solutions du problème 1. Nous passons maintenant au problème 2.

II.2.6.2 L'estimation de la suite d'états cachés

Étant donnée une suite d'observations $O = O_1\ O_2\ O_3\ \ldots\ O_T$ et un modèle $\lambda = (A,B,\pi)$, comment peut-on choisir une suite d'états $Q = q_1,q_2,q_3,\ldots q_T$ selon un critère convenable. Contrairement au problème 1 (l'évaluation du modèle), pour lequel une solution exacte peut être trouvée, ici il nous faut déterminer la séquence d'états optimale associée à la séquence d'observations. La difficulté principale réside dans la définition du critère d'optimalité. Nous allons présenter ci-dessous deux approches.

a) Estimation de l'état le plus probable

Le critère d'optimalité utilisé par cette technique est le choix de l'état $q_t = S_i$ le plus probable individuellement. Ceci revient à choisir à l'instant t l'état qui maximise la probabilité d'être dans l'état S_i connaissant la séquence d'observations O et le modèle λ :

$$\gamma_t(i) = P(q_t = S_i \mid O,\lambda) \quad \text{(II.25)}$$

Cette équation peut être réécrite en fonction des variables α et β, précédemment définies :

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)} \qquad \text{(II.26)}$$

Dans cette écriture, la variable $\alpha_t(i)$ représente la probabilité de la séquence d'observations partielle $O_1\ O_2\ \ldots\ O_t$ et la réalisation de l'état S_i à l' instant t, alors que $\beta_t(i)$ représente le reste de la séquence d'observations $O_{t+1}\ O_{t+2}\ \ldots\ O_T$ sachant que nous sommes dans l'état S_i à l'instant t. Le facteur de normalisation $P(O|\lambda) = \sum_{i=1}^{N}\alpha_t(i)\beta_t(i)$ nous assure que $\gamma_t(i)$ est réellement une probabilité, c'est-à-dire :

$$\sum_{i=1}^{N}\gamma_t(i) = 1 \qquad \text{(II.27)}$$

L'utilisation de la variable $\gamma_t(i)$ nous permet de calculer l'état le plus probable à chaque instant t pris individuellement :

$$q_t = \arg \max_{1\le i\le N}\left[\gamma_t(i)\right] \qquad \text{(II.28)}$$

L'utilisation de cette technique nous permet de maximiser le nombre espéré d'états individuellement, en sélectionnant l'état le plus probable à chaque instant t. Cependant il peut exister un problème avec la séquence d'états résultante. En effet, pour un modèle donne, s'il existe des probabilités de transition nulles (a_{ij} = 0), la séquence d'états résultante peut être non valide. Ceci vient du fait que cette séquence est déterminée sans tenir compte de la probabilité d'occurrence des séquences d'états. En d'autres mots, la séquence d'états optimale que nous allons obtenir peut ne pas être réalisable car la probabilité de transition d'un état k à un état $k+1$ de cette séquence peut être nulle.

Une solution possible à ce problème est de modifier le critère d'optimalité, Nous pouvons par exemple déterminer la séquence d'états qui maximise la probabilité de paire (q_t ,q_{t+1}) ou de triplet (q_t , q_{t+1} , q_{t+2}) d'états. En fait le critère le plus utilisé est celui qui permet de trouver la séquence d'états optimale et unique. Cela revient à maximiser la probabilité $P(Q|\ O,\lambda)$. ce qui est équivalent à maximiser la probabilité $P(Q,\ O|\ \lambda)$. Une technique formelle permettant de trouver l'unique et meilleure

séquence d'états existe. Elle est basée sur la programmation dynamique, c'est l'algorithme de Viterbi, dont le développement va suivre.

b) L'algorithme de Viterbi

Pour trouver l'unique et meilleure séquence d'états Q=q_1 q_2 q_3 ...q_T correspondant à la séquence d'observations O_1 O_2 ... O_T, nous avons besoin de définir la quantité suivante :

$$\delta_t(i) = \max_{q1,q2,q3....,qt-1} P\left[q_1 q_2 q_3 ... q_t = i, O_1 O_2 .. O|\lambda\right] \qquad \text{(II.29)}$$

$\delta_t(i)$ est le meilleur score. C'est-à-dire la probabilité la plus élevée, correspondant à une trajectoire unique pour arriver à l'état S_i au temps t, et qui prend en compte les t premières observations. Cette quantité peut s'écrire de manière récursive :

$$\delta_{t+1}(j) = \left(\max_t \left[\delta_t(i) a_{ij}\right] \right) b_j(O_{t+1}) \qquad \text{(II.30)}$$

Afin de retrouver la séquence d'états, nous devons garder une trace de l'argument qui maximise cette quantité pour chaque instant t et chaque état j. Nous effectuons ceci a l'aide d'une matrice, généralement appelée $\psi_t(j)$. Nous allons présenter maintenant la procédure qui permet de trouver la meilleure séquence d'états :

- 1-Initialisation

$$\delta_1(i) = \pi_i \; b_i(O_1) \qquad 1 \le i \le N \qquad \text{(II.31)}$$

$$\psi_1(i) = 0 \qquad \text{(II.32)}$$

- 2- Induction

$$\delta_t(j) = \left(\max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right] \right) b_j(O_t) \text{ pour } 2 \le t \le T \text{ et } 1 \le j \le N \qquad \text{(II.33)}$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right] \qquad \text{pour } 2 \le t \le T \text{ et } 1 \le j \le N \qquad \text{(II.34)}$$

- 3-Terminaison

$$P^* = \max_{1 \le i \le N} \left[\delta_T(i)\right] \qquad \text{(II.35)}$$

$$q_T^* = \arg \max_{1 \le i \le N} \left[\delta_T(i)\right] \qquad \text{(II.36)}$$

- 4-Obtention de la séquence d'états optimale (backtracking)

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad \text{pour } t = T-1, T-2, \ldots, 1 \qquad \text{(II.37)}$$

Nous constatons que l'algorithme de Viterbi est assez proche de la procédure de recherche avant, mis à part la procédure de recherche arrière. La différence majeure se situe au niveau de l'induction. Dans cet algorithme une maximisation sur les états précédents est mise en œuvre, au lieu d'une somme dans le cas de la procédure de recherche avant.

II.2.6.3 L'apprentissage

Le troisième et le plus difficile des problèmes à résoudre est celui de l'apprentissage Comment pouvons nous ajuster les paramètres du modèle λ (A,B, π) pour maximiser la probabilité de la séquence d'observations $P(O \mid \lambda)$?

En fait, pour toute séquence d'observations finie, il n'existe pas de méthode analytique permettant de trouver la valeur optimale des paramètres du modèle, maximisant la probabilité $P(O \mid \lambda)$. Cependant nous pouvons trouver des valeurs de ces paramètres correspondant à un maximum local de la solution, ceci en utilisant des procédures itératives telles que celle de Baum-Welch ou une méthode équivalente appelée EM (pour Expectation-Modification ou Maximisation). Ces méthodes permettent de déterminer un nouveau modèle par rapport à un modèle initial, qui remplace alors le modèle précédent et ce jusqu'a un critère d'arrêt défini au préalable. Le principe de ces méthodes de ré-estimation est le suivant :

- 1- Initialiser les paramètres du modèle.
- 2- Pour chaque observation sur le modèle considéré
- 3- Calculer les variables de la méthode considérée.
- 4- Comptabiliser les différentes transitions.
- 5- Comptabiliser les symboles émis par chaque état.
- 6- Ré-estimer le modèle.
- 7- Retourner en 2 jusqu'à ce que le critère de fin soit vérifié.

Le critère de fin peut être de différents types. Nous pouvons par exemple fixer au préalable le nombre d'itérations ou mettre fin aux itérations lorsqu'il n'y a plus de variation notable des paramètres du modèle. Nous allons maintenant présenter l'algorithme de Baum-Welch pour l'estimation des paramètres du modèle.

L'algorithme de Baum-Welch

Le fonctionnement de cet algorithme est basé sur la ré-estimation des paramètres du modèle en partant d'un modèle initial. Afin de décrire cette procédure, nous allons définir la probabilité $\xi_t\ (i,\ j)$.Cette quantité exprime la probabilité d'être à l'état S_i, à l'instant t et à l'état S_j à l'instant $t+1$, connaissant le modèle et la séquence d'observations :

$$\xi_t\ (i,\ j) = P\ (q_t = S_i\ , q_{t+1} = S_j\ |\ O,\ \lambda\ ) \qquad \text{(II.38)}$$

Nous pouvons écrire cette valeur d'une autre manière à l'aide des variables α et β (définies précédemment) :

$$\xi_t(i,\ j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \qquad \text{(II.39)}$$

où le numérateur est la probabilité $P\ (q_t = S_i\ , q_{t+1} = S_j\ , O\ |\ \lambda\ )$ et le numérateur la probabilité de la séquence d'observations connaissant le modèle. Le rapport de ces valeurs nous permet d'obtenir la quantité désirée.

Nous pouvons réécrire la variable $\gamma_t(i)$, à savoir la probabilité d'être dans un état S_t à l'instant t, connaissant la séquence d'observations et le modèle, en fonction de la quantité $\xi_t\ (i,\ j)$, de la manière suivante :

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,\ j) \qquad \text{(II.40)}$$

Si maintenant nous effectuons la somme des variables $\gamma_t(i)$ suivant t, nous obtenons une quantité qui peut être interprétée comme le nombre de fois où l'état S_i est visité ou encore le nombre attendu de transitions atteintes depuis S_i (si nous excluons l'instant final t = T de la somme). De manière identique, la somme des termes $\xi_t\ (i,\ j)$ suivant t (toujours en excluant l'instant final) peut être interprétée comme le nombre attendu de transitions de l'état S_i vers l'état S_j. Nous pouvons donc écrire :

$$\sum_{t=1}^{T-1}\gamma_t(i) = \text{nombre attendu de transitions atteintes depuis } S_i \qquad \text{(II.41)}$$

En utilisant ces formules et le concept de comptage d'occurrence des événements, nous pouvons donner une méthode de ré-estimation des paramètres du modèle A, B *et* π :

$$\overline{\pi_t} = \textit{nombre attendu d'occurrence de l'état } S_i \textit{ à l'instant } t = 1 \qquad \text{(II.42)}$$

$$\overline{a_{ij}} = \frac{\textit{nombre attendu de transitions de l'état } S_i \textit{ vers l'état } S_j}{\textit{nombre attendu de transitions atteintes depuis } S_i} \qquad \text{(II.43)}$$

$$\overline{b_j}(k) = \frac{\textit{nombre attendu d'occurences de l'observation } v_k \textit{ depuis l'etat } j}{\textit{nombre attendu d'occurences de l'etat } j} \qquad \text{(II.44)}$$

Nous pouvons écrire ces valeurs en fonction des variable $\gamma_t(i)$ $\xi_t(i,j)$:

$$\overline{\pi_t} = \gamma_1(i) \;;\quad \overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad ; \quad \overline{b_j}(k) = \frac{\sum_{t=1 \text{ pour } Ot=Vk}^{T} \gamma_t(j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \text{(II.45)}$$

À partir de ces valeurs, nous obtenons un nouveau modèle $\overline{\lambda} = (\overline{A}\ \overline{B}\ \overline{\pi}\ )$ Baum et ses collègues ont montré dans Baum et al. (1970), que si le modèle initial λ définit un point critique de la fonction de probabilité $P(O|\ \lambda)$ alors dans ce cas $\overline{\lambda} = \lambda$ (ceci est un cas rare) et que sinon le modèle $\overline{\lambda}$ est plus probable que le modèle initial, c'est-à-dire que :

$p\ (\ O|\ \overline{\lambda}\ ) > p\ (\ O\ |\lambda\ )$ et dans ce cas nous avons un nouveau modèle à partir duquel la séquence d'observations a plus de chance d'être produite.

En effectuant cette procédure de manière itérative et en fixant un critère de fin, nous pouvons améliorer la probabilité que la séquence O soit observée par le modèle. Le résultat final de cette procédure itérative est appelé : estimation par le maximum de vraisemblance des paramètres du modèle de Markov caché. Une autre écriture de la ré-estimation des paramètres a également été proposée par Baum. Soit la fonction auxiliaire de Baum :

$$Q(\lambda, \overline{\lambda}) = \sum_{Q} P(Q \mid O, \lambda)\ \log\left[P(O, Q \mid \overline{\lambda})\right] \qquad \text{(II.46)}$$

Les fonctions de ré-estimation des paramètres du modèle présentées plus haut peuvent être directement dérivées de cette fonction auxiliaire. Baum et ses collègues ont prouvé que la maximisation de cette quantité par rapport à $\bar{\lambda}$ permet d'augmenter la probabilité de la séquence d'observations :

$$\max_{\bar{\lambda}} \left[Q(\lambda, \bar{\lambda}) \right] \Rightarrow P(O \mid \bar{\lambda}) \geq P(O \mid \lambda) \qquad \text{(II.47)}$$

Sous cette forme la ré-estimation des paramètres peut être interprétée comme une implémentation de l'algorithme statistique EM (Expectation Modification ou Maximisation). La phase E est alors le calcul de la fonction auxiliaire et la phase M est la maximisation.

II.2.7 Les modèles discrets et modèles continus

Dans les sections précédentes, nous avons supposé que le modèle de Markov caché émettait à chaque transition l'un des M symboles d'un alphabet fini. Un tel modèle est qualifié de discret. Ceci implique que l'on doit appliquer une quantification vectorielle sur le signal d'entrée entraînant ainsi une perte naturelle de précision. Pour minimiser cette distorsion de quantification vectorielle, nous pouvons augmenter le nombre de symboles de cet alphabet. Cependant, ceci entraîne un accroissement de la taille de la matrice B. La taille de la base d'apprentissage doit également être augmentée afin d'obtenir une bonne estimation des paramètres, ce qui n'est pas toujours possible. Si nous supposons que les vecteurs d'observations émis par un état S_i sont obtenus à partir d'une distribution de probabilité continue quelconque, nous pouvons remplacer la ligne correspondant à cet état dans la matrice B, par les paramètres de cette distribution. Ceci est analogue à l'approximation d'un histogramme de données quelconque par une distribution normale, où l'on remplace les probabilités individuelles dans l'histogramme par la moyenne et la variance estimées de la distribution normale. Nous parlons alors de modèles continus. Il reste cependant un problème à résoudre : l'estimation des paramètres de la distribution continue. La solution est donnée par un résultat mathématique qui montre que toute fonction de probabilité continue multi variable arbitraire peut être approximée, pour toute précision désirée, par un mélange (somme pondérée) de fonctions de densité de probabilité continue gaussienne et multi variables. Par conséquent, nous pouvons modéliser la probabilité $b_j(O_t)$

d'observer le vecteur O_t à partir d'un état S_i par un mélange de gaussiennes de la forme :

$$b_j(O_t) = \sum_{j=1}^{X} c_{jm} N(O_t, \mu_{jm}, \Sigma_{jm}) \qquad j = 1,2,3,\ldots,N \tag{II.48}$$

où :

N représente une densité normale multi variable.

C_{jm} est le poids de la $m^{ième}$ composante du mélange pour l'état j avec :

$$0 \le C_{jm} \le 1 \quad \text{et} \quad \sum_{m=1}^{X} C_{jm} = 1 \tag{II.49}$$

X est le nombre de composantes du mélange.

μ_{jm} et Σ_{jm} sont respectivement le vecteur moyen et la matrice de covariance de la $m^{ième}$ composante du mélange pour l'état j.

Un modèle de Markov caché utilisant des densités de probabilité continues comme décrit ci-dessus est qualifié de continu. En résumé, pour chaque état du modèle, nous avons un mélange de gaussiennes qui nous permet d'obtenir la probabilité de l'observation traitée. Donc il va nous falloir estimer, pour chaque état, les paramètres des X gaussiennes, ainsi que les coefficients du mélange C_{jm}. Ceci augmente considérablement le nombre total de paramètres à estimer.

Les formules de ré-estimation des coefficients de la densité de mélange C_{jm}, μ_{jm} et Σ_{jm} sont de la forme :

$$\overline{C}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)}{\sum_{t=1}^{T} \sum_{k=1}^{X} \gamma_t(j,m)} \tag{II.50}$$

$$\overline{\mu}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)\ O_t}{\sum_{t=1}^{T} \gamma_t(j,m)} \tag{II.51}$$

$$\overline{\Sigma}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(j,m)(O_t - \mu_{jm})(O_t - \mu_{jm})'}{\sum_{t=1}^{T} \gamma_t(j,m)} \tag{II.52}$$

où prime correspond au vecteur transposé, et $\gamma_t(j,m)$ est la probabilité d'être a l'état S_i à l'instant t, et que la $m^{ième}$ composante du mélange compte pour l'observation O_t :

$$\gamma_t(j,m) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}\right] \cdot \left[\frac{C_{jm}N(O_t,\mu_{jm},\Sigma_{jm})}{\sum_{n=1}^{X}C_{jm}N(O_t,\mu_{jm},\Sigma_{jm})}\right] \qquad \text{(II.53)}$$

Ce terme généralise le terme $\gamma_t(j)$, utilisé dans le cas de densité discrète ou de mélange simple. La formule permettant la ré-estimation des coefficients C_{jm} est en fait le rapport entre le nombre espéré de fois où le système est à l'état S_j en utilisant la $m^{ième}$ composante du mélange, sur le nombre espéré de fois où le système est à l'état S_j. La formule de ré-estimation du vecteur des moyennes μ_{jm} est proche de celle des coefficients C_{jm}, la seule différence est la pondération du numérateur par l'observation. Ceci permet d'obtenir la valeur espérée de la contribution du vecteur d'observations pour la $m^{ième}$ composante du mélange.

II.3 MÉTHODE DE LA RÉTROPROPAGATION NEURONALE

Les réseaux de neurones formels sont des structures (la plupart du temps simulées par des algorithmes exécutés sur des ordinateurs d'usage général, parfois sur des machines ou même des circuits spécialisés) qui prennent leur inspiration (souvent de façon assez lointaine) dans le fonctionnement élémentaire des systèmes nerveux. Ils sont utilisés essentiellement à résoudre des problèmes de classification, de reconnaissance de formes, d'association, d'extraction de caractéristiques, d'identification. Ils deviennent des compléments aux méthodes classiques, et sont même susceptibles de se substituer à celles-ci avec un taux de succès supérieur. Un schéma classique d'un neurone formel est illustré à la Figure II.6.

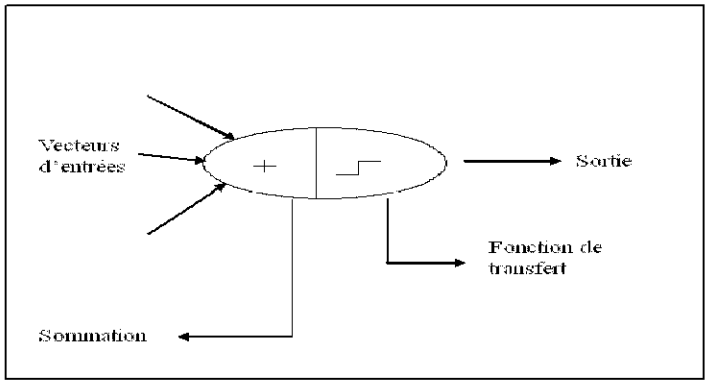

Figure II.6 : *Schéma classique d'un neurone.*

II.3.1 Le neurone formel

Un neurone formel fait une somme pondérée des potentielles d'action qui lui parvient (chacun de ces potentiel est une valeur numérique qui présente l'état du neurone qui lui émis), puis s'active suivant la valeur de cette sommation pondérée. Si celle-ci dépasse un certain seuil, le neurone est activé et transmet une réponse (sous forme de potentiel d'activation) dont la valeur est celle de son activation, si le neurone n'est pas activé, il ne transmet rien.

Le modèle du neurone formel peut être constitué comme illustré à la Figure II.7.

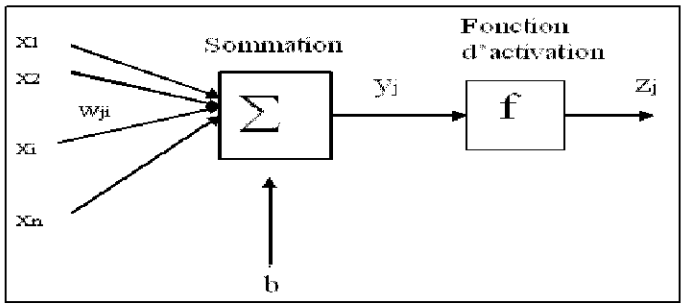

Figure II.7 : *Model d'un neurone formel.*

Où :

Xi : représentent les vecteurs d'entrée du neurone (j)

Wji : les poids de ces vecteurs

b : le seuil du neurone (b=0 , dans notre application)

$$y\,j = \sum_{i=1}^{n} Wji \cdot Xi + b$$

zj : la fonction du neurone (sigmoide-logarithmique dans notre application)

$$zj = 1/1 + \exp(-x)$$

II.3.2 La fonction d'activation

La fonction d'activation du neurone définit son état interne en fonction de son entrée totale. Le comportement d'un neurone dépend essentiellement du choix de sa fonction d'activation.

La fonction sigmoïde

$$a = 1/(1+\exp(-n))$$

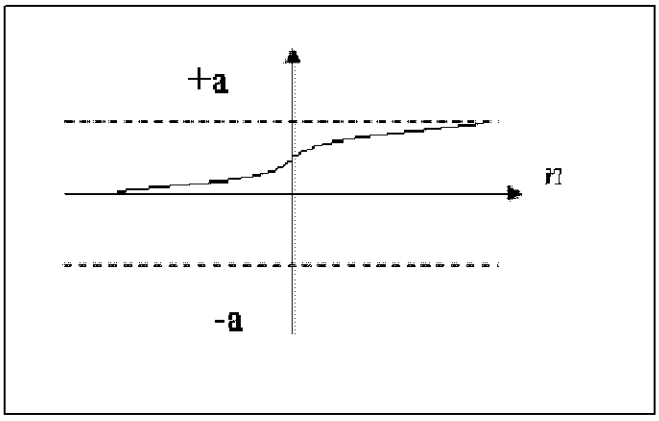

Figure II.8 : *fonction de transfert logarithmique sigmoïde.*

II.3.3 Réseaux de neurones

Concernant notre application, nous nous intéressons uniquement aux réseaux de neurones de type perceptrons multicouches (multilayers perceptron) « MLP », appropriés à l'apprentissage supervisé. La Figure II.9 représente un schéma synoptique d'un réseau de neurones.

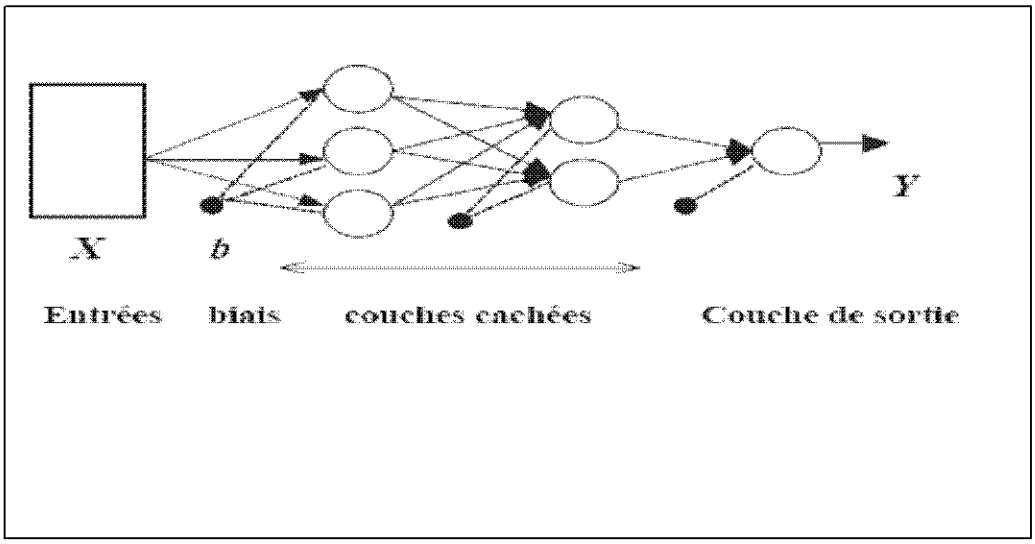

Figure II.9 : *Schéma d'un réseau de neurone.*

- La couche d'entrée :

Elle reçoit les données que l'on veut utiliser. Sa taille est donc directement déterminée par le nombre de variables d'entrées.

- La couche cachée :

La seconde couche est la couche cachée, dans le sens qu'elle n'a pas de contact direct avec l'extérieur. Les fonctions d'activations sont en général non linéaires.

Le choix de sa taille (nombre de neurones) n'est pas implicite et doit être ajusté.

- La couche de sortie :

La dernière couche est appelée couche de sortie, elle donne le résultat obtenu après introduction des données d'entrées dans la première couche cachée.

La taille de cette couche est directement déterminée par le nombre de formes (classes) désiré.

II.3.4 Apprentissage

L'apprentissage est une phase du développement du réseau de neurones durant laquelle on calcule les poids des neurones de telle manière que les sorties du réseau soient aussi proches que possibles des sorties désirées. L'apprentissage fait appel à des exemples de comportement du processus de classification.

L'apprentissage est dit **supervisé** lorsque les exemples sont constitués de couples de valeurs du type : (valeur d'entrée, valeur de sortie désirée). Tout le problème de l'apprentissage supervisé consiste a faire rapprocher la sortie réelle de la sortie désirée, donc déterminer le vecteur des poids des neurones capables de prédire le même vecteur de sortie à partir du même vecteur d'entrée.

L'apprentissage est qualifié de **non supervisé** lorsque seules les valeurs d'entrée sont disponibles. Dans ce cas, les exemples présentés à l'entrée provoquent une auto adaptation du réseau afin de produire des valeurs de sortie qui soient proche en réponse pour des valeurs d'entrées similaires.

Durant l'apprentissage, les poids sont ajustés itérativement pour minimiser la fonction de performance mesurant l'écart entre les sorties effectives S et les sorties désirées D ; l'erreur la plus couramment employée est l'erreur quadratique

$$Ep=1/2\sum_{K=1}^{M}(Dk - Zk)^2 \qquad \text{erreur de l'exemple P.} \qquad (II.54)$$

(Z k , D k) sont successivement la sortie réelle et la sortie désirée de la couche k).

Dans ce chapitre, nous allons décrire l'algorithme d'apprentissage, il utilise le gradient de la fonction de performance pour déterminer comment ajuster les poids pour minimiser l'erreur.

Le problème pour un réseau à une ou plusieurs couches cachées est d'évaluer les valeurs optimales des poids des connexions. La solution proposée est de rétropropager l'erreur proportionnellement aux poids.

II.3.5 Algorithme de Rétropropagation

Le principe de l'algorithme de Rétropropagation consiste à ajuster les poids dans le sens inverse du gradient de la fonction de performance comme suit :

$$\Delta W_{j,i} = -\partial E / \partial W_{j,i} \qquad \text{(II.55)}$$

Pour avoir un bon fonctionnement de l'algorithme de Rétropropagation, il a eu d'introduire deux paramètres en plus, qui sont :

- **le pas d'apprentissage (a) :**

  Il sert à faire converger les poids vers une solution, en effectuant des sauts discret pour les poids, il prend des valeurs comprises dans l'intervalle [0.1 , 1].

- **le momentum (δ) :** (cœfficient d'inertie)

  Il sert à réduire le temps d'apprentissage dans le cas où on utilise un grand pas d'apprentissage, il prend des valeurs comprises dans l'intervalle [0.6 , 0.9].

II.3.6 Etapes de l'algorithme de Rétropropagation de l'erreur

Les étapes principales de cet algorithme sont classées comme suit :

1- Initialisation des poids synaptiques W ji , W kj avec des valeurs aléatoires faibles.

2- Application des valeurs d'entrée et des vecteurs de sortie désirée.

3- Calcul des activations des cellules cachées : $Y_j = \sum_{i=1}^{N} Wji \cdot Xi$

4- Calcul des sortie des cellules cachées : Z j = fj (Y j)

5- Calcul des activation des cellules de sorties par : $Y_k = \sum_{j=1}^{L} Wkj \cdot Zj$

6- Calcul des sorties des cellules de sortie par : Z k =fk (Y k)

7- Calcul de l'erreur quadratique : $E=1/2 \sum_{K=1}^{M} (D_k - Z_k)^2$

8- Adaptation des poids de la couche de sortie (mise à jour)

$\partial E / \partial Wkj = -(Dk-Zk)\ \partial(Dk - Zk) / \partial Wkj$

$= (D_k - Z_k)(\partial Z_k / \partial W_{kj})$

$= -\,err(k)(\partial Z_k / \partial (Y_j)\ (\partial Y_j / \partial W_{kj})$

$= -\,err(k)\,F'_k(Y_k)\ (\partial Y_k / \partial W_{kj})$

$= err(k)(F'_k(Y_j))\,Y_j$

$\Delta W_{kj} = -\,n\ \partial E / \partial W_{kj}$

$\Delta W_{kj} = n\,[\,err(k)\ (F'_k(Y_k))\ (-a)\ Y_k\,]$

$W_{kj}(t+1) = \Delta W_{kj} + W_{kj}(t) + (m \,.\, \partial W_{kj})$

Où :

$W_{kj}(t)$: est l'ancienne valeur des poids de la couche de sortie.

$W_{kj}(t+1)$: est la nouvelle valeur des poids après l'adaptation (la mise à jour).

9- Adaptation des poids de la couche cachée :

$$\partial E / \partial W_{ji} = 1/2 \sum_{K=1}^{M} (D_k - Z_k)^2 \ / \ \partial W_{ji}$$

$$= -\sum_{K=11}^{M} (\partial Z_k / \partial W_{ji})\ (D_k - Z_k)$$

$$= -\sum_{K=11}^{M} err(k)\ (\partial Z_k / \partial W_{ji})$$

$$= -\sum_{K=11}^{M} err(k)\ (\partial Z_k / \partial Y_k)(\partial Y_k / \partial Z_j)(\partial Z_j / \partial Y_j)(\partial Y_j / \partial W_{ji})$$

$$= -\sum_{K=11}^{M} err(k)\ (F'_k(Y_k))\ W_{kj}\ (F'_j(Y_k))\ X_i$$

$$= -\,X_i\ F'_j(Y_j) \sum_{K=11}^{M} err(k)\,(F'_k(Y_k))\,W_{kj}$$

$$\Delta W_{ji} = n\ [F'_j(Y_j)\ X_i \sum_{K=11}^{M} err(k)\,(F'_k(Y_k)\ (-a)\ W_{kj}$$

$W_{ji}(t+1) = \Delta W_{ji} + W_{ji}(t) + (m \,.\, \partial W_{kj})$

Où :

$W_{ji}(t)$: est l'ancienne valeur des poids de la couche caché.

$W_{ji}(t+1)$: est la nouvelle valeur des poids après l'adaptation.

Voici le réseau entraîné :

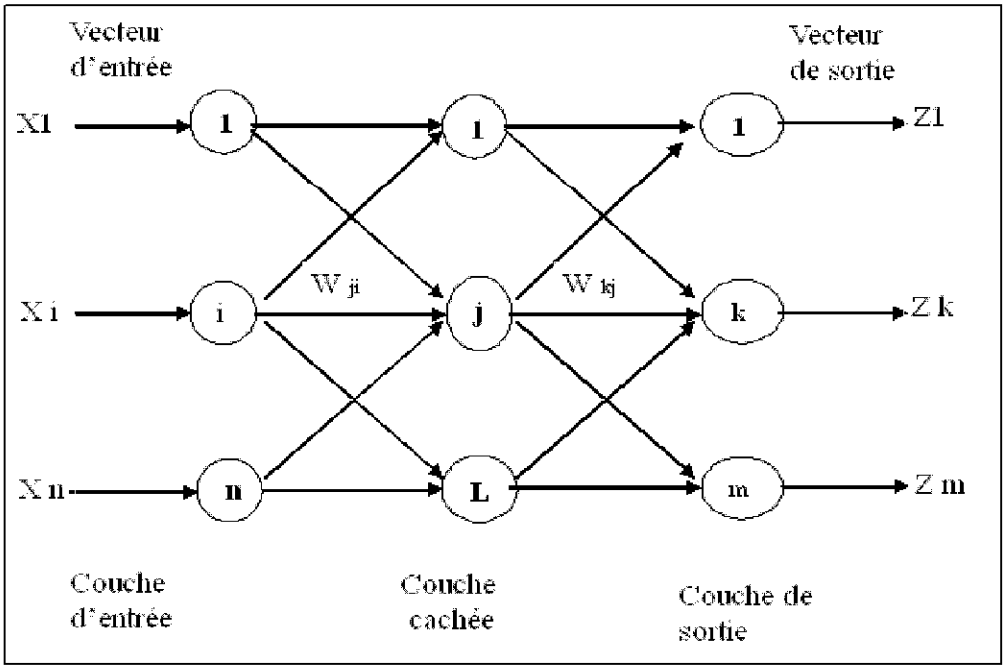

Figure II.10 : *Le réseau à couche cachée*

W_{ji} : poids reliant le nœud i au nœud j

W_{kj} : poids reliant le nœud j au nœud k

n : le nombre de cellules de la couche d'entrée

L : le nombre de cellules de la couche cachée

m : le nombre de cellules de la couche de sortie

F j : la fonction d'activation du neurone j de la couche de cachée

Z_k : sortie de cellule k de la couche cachée

F_k : la fonction d'activation de la cellule k de la couche de sortie

D_k : la sortie désire de la cellule k

err (k) : erreur de la cellule k de la couche de sortie

Ce chapitre a fait l'objet de l'approche proposée pour la classification des locuteurs. Deux méthodes ont été retenues :

- Les modèles cachés de Markov;
- Algorithme de Rétropropagation du gradient des réseaux de neurones.

III.1 INTRODUCTION

Nous avons présenté dans les chapitres précédents notre approche de classification ainsi que les prétraitements effectués sur notre banque de données. Le but de l'étape de prétraitement, est d'extraire l'information essentielle caractérisant la signature acoustique d'une personne à identifier, et de la présenter sous la forme la plus concentrée possible au classifieur.

Il est important de remarquer que le problème traité dans ce travail est la classification supervisée, c'est-à-dire pour laquelle l'utilisateur connaît et le système utilise l'information concernant l'appartenance aux classes des vecteurs d'apprentissages. Nous disposons donc d'un ensemble de vecteurs étiquetés, qui vont être présentés au classifieur afin que celui-ci puisse en déduire une règle de décision optimisant un certain critère.

III.2 CONSTRUCTION DE LA BANQUE DE DONNÉES

Nous proposons une banque de donnée composée de six (06) classes qui sont : classe 1 : Imane, classe 2 : Linda, classe 3 : Ouiza, classe 4 : Ghani, classe 5 : Mourad, classe 6 : Toufik. Le résultat du traitement appliqué est un ensemble de matrices de dimension (8*60) ; c à d, 60 vecteurs de 8 éléments chacun. Les vecteurs de chaque matrice (classe) sont pris de façon aléatoire.

III.3 LES PARAMÈTRES DES CLASSIFIEURS

Nous utilisons un perceptron à une seule couche cachée. La fonction d'activation est la sigmoïde. Le nombre de neurones en entrée est 8 (taille du vecteur). Le nombre de neurones en sortie est 6 (6 classes). Le nombre de neurones dans la couche cachée est déterminé par tâtonnement. Concernant les modèles cachés de Marcov, nous avons choisi 05 états. Les matrices sont initialisées de façon aléatoire.

Les vecteurs d'entrée appartiennent à (06) classes différentes réparties sur des matrices de (8*60) chacune. Dans la phase de l'apprentissage, nous prenons d'une manière aléatoire 45 (75% de l'ensemble) vecteurs parmi les 60 possibles, les 15 (25% de l'ensemble) restants de chaque matrice sont retenus pour la généralisation (le test).

Les performances de classification, sont déterminées par :

- Le diagramme de l'erreur quadratique et la matrice de confusion pour la méthode neuronale.
- La matrice de confusion pour les modèles cachés de Marcov.

III.4 ÉVALUATION DES PERFORMANCES DE CLASSIFICATION

III.4.1 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 rétropropagée, c'est-à-dire propagée de la sortie vers l'entrée en recalculant les valeurs des poids synaptiques (mise à jour des poids). Cette procédure est itérée plusieurs fois (par fois jusqu'à 100000 itérations). A chaque introduction d'un vecteur, l'erreur est calculée est comparée à la valeur 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.

III.4.2 La matrice de confusion

Elle est de dimension n * n (où n est le nombre de classe de sortie, (égal à 6 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 III.1. Idéalement cette matrice ne devait contenir que des " 1 " sur la diagonale et des " 0 " partout ailleurs.

Nous définissons le paramètre Pcc, qui représente la probabilité de classification correcte qui est donné par la moyenne de la diagonale de la matrice de confusion.

	C1	C2	C3	C4	C5	C6
C1	C1/C1	C2/C1	C3/C1	C4/C1	C5/C1	C6/C1
C2	C1/C2	C2/C2	C3/C2	C4/C2	C5/C2	C6/C2
C3	C1/C3	C2/C3	C3/C3	C4/C3	C5/C3	C6/C3
C4	C1/C4	C2/C4	C3/C4	C4/C4	C5/C4	C6/C4
C5	C1/C5	C2/C5	C3/C5	C4/C5	C5/C5	C6/C5
C6	C1/C6	C2/C6	C3/C6	C4/C6	C5/C6	C6/C6

Figure III.1 : *Matrice de confusion, n = 6.*

III.5 PRÉSENTATION DES RÉSULTATS OBTENUS

III.5.1 Quelques résultats obtenus par la Rétropropagation

Dans ce qui suit, nous considérons :

It : le nombre d'itérations ;

nn : le nombre de neurones de la couche cachée;

a : le pas d'apprentissage ;

mom : le momentum ;

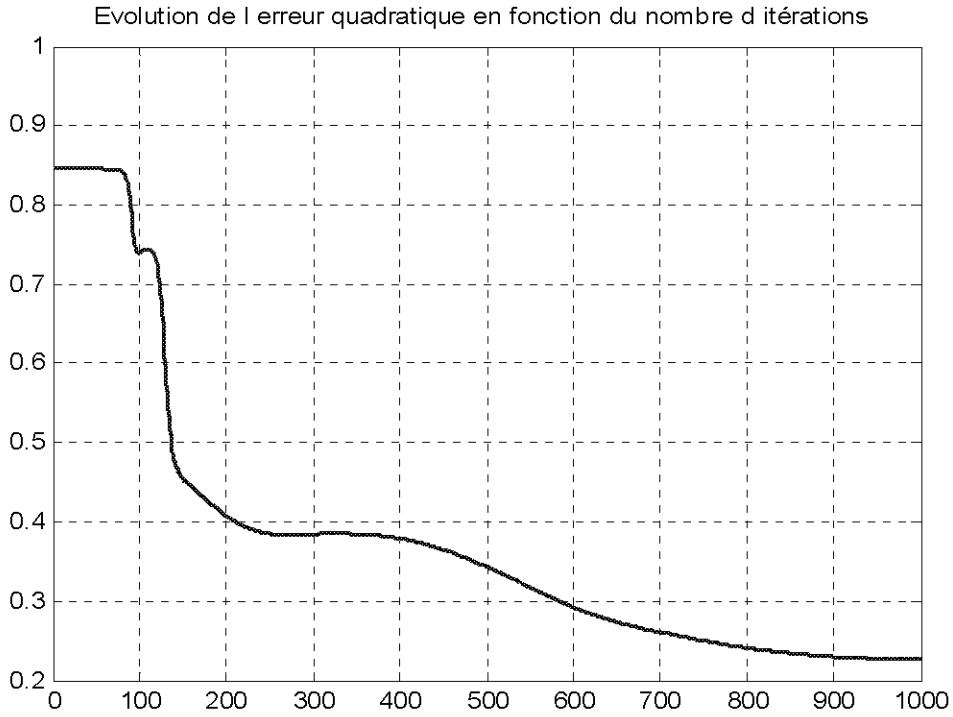

Figure III.2 : *Evolution de l'erreur quadratique.*
It = 1000 , a = 0.2 , mom = 0,999 , nn = 5

0.4889	0.2222	0.0667	0	0.1111	0.1111
0.3556	0.3111	0.0444	0.0444	0.1111	0.1333
0.0889	0.0889	0.5333	0.0444	0.1333	0.1111
0.0444	0.0444	0.1778	0.5111	0.2222	0
0	0	0.0444	0.0667	0.7556	0.1333
0.2000	0.0889	0.0889	0.0222	0.1556	0.4444
		Pcc =0.5074			

Figure III.3 : *Le résultat du test avec 75 % des données d'apprentissage. Apprentissage de Figure III.2*

0.5333	0.2667	0	0	0.0667	0.1333
0.1333	0.2000	0.0667	0.1333	0.2000	0.2667
0.0667	0.1333	0.5333	0.0667	0.2000	0
0	0.0667	0.2667	0.4000	0.2000	0.0667
0	0.0667	0.0667	0.0667	0.7333	0.0667
0.0667	0	0.1333	0	0.1333	0.6667
		Pcc =0.5111			

Figure III.4 : *Le résultat de la classification avec 25 % des données de test. Apprentissage de Figure III.2*

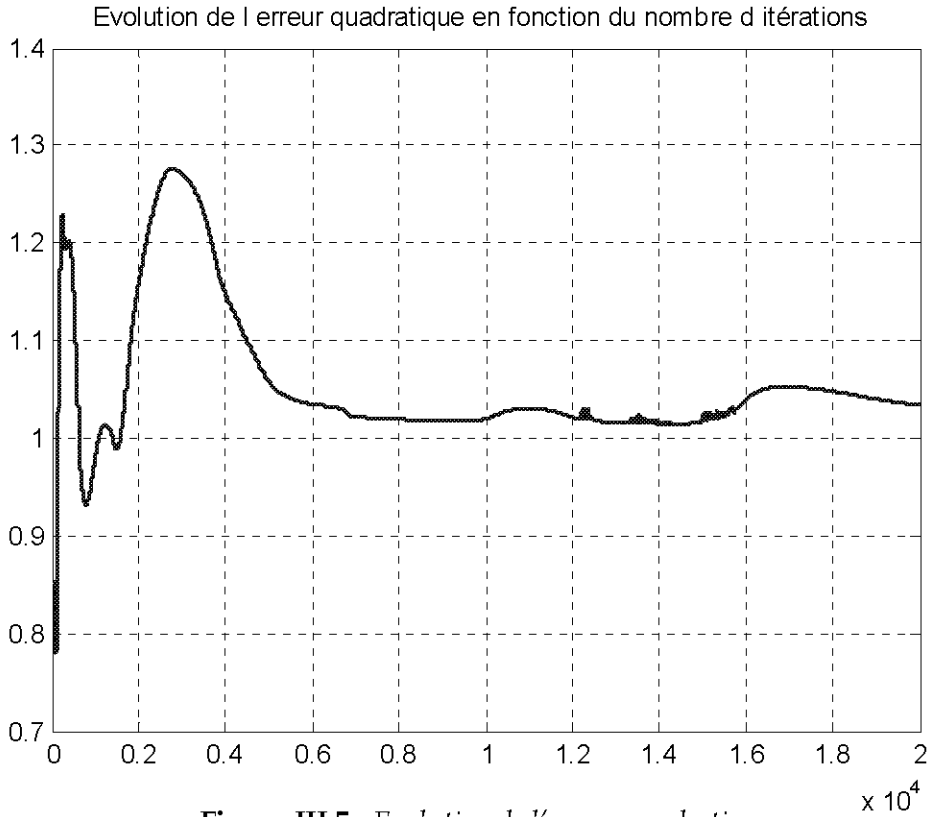

Figure III.5 : *Evolution de l'erreur quadratique.*
It = 20000 , a = 0.2 , mom = 0,998 , nn = 10

0.7333	0.0889	0.0667	0.0667	0.0222	0.0222
0.0667	0.7333	0.0444	0.0222	0.0444	0.0889
0.0667	0.1111	0.5778	0.1111	0.0667	0.0667
0.0667	0.0222	0.1111	0.6444	0.1111	0.0444
0	0.0444	0.0222	0	0.8444	0.0889
0.0667	0.0889	0.0222	0.0222	0.1111	0.6889

Pcc =0.7037

Figure III.6 : *Le résultat du test avec 75 % des données d'apprentissage. Apprentissage de Figure III.5*

0.2667	0.6000	0.0667	0	0	0.0667
0.1333	0.4000	0.0667	0.1333	0.0667	0.2000
0.1333	0.1333	0.3333	0.1333	0.2000	0.0667
0	0	0.4000	0.3333	0.2667	0
0	0	0.1333	0.1333	0.6000	0.1333
0.1333	0.2000	0	0.1333	0	0.5333

Pcc=0.4111

Figure III.7 : *Le résultat de la classification avec 25 % des données de test. Apprentissage de Figure III.5*

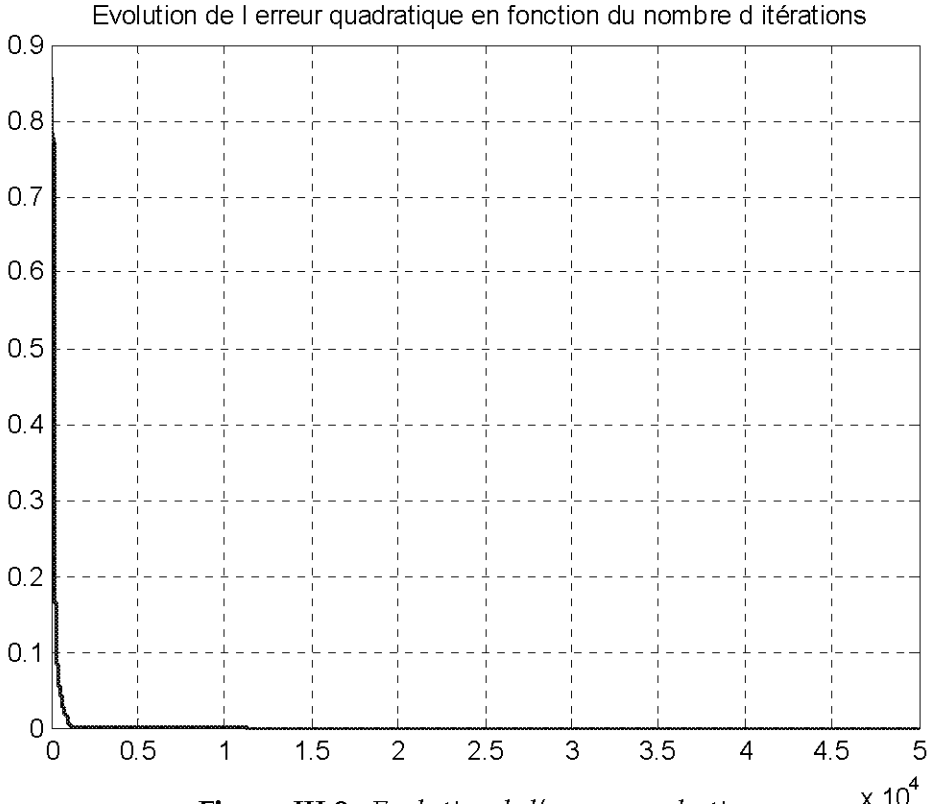

Figure III.8 : *Evolution de l'erreur quadratique.*
It = 50000 , a = 0.2 , mom = 0,998 , nn = 12

0.7333	0.1333	0.0222	0.0222	0.0222	0.0667
0.0667	0.6667	0.0444	0.0222	0.0889	0.1111
0.0222	0.0667	0.6889	0.0667	0.0889	0.0667
0.0222	0.0222	0.0889	0.7556	0.0889	0.0222
0	0.0222	0.0222	0.0667	0.8222	0.0667
0.0889	0.0444	0.0222	0.0444	0.0444	0.7556

Pcc =0.7370

Figure III.9 : *Le résultat du test avec 75 % des données d'apprentissage. Apprentissage de Figure III.8*

0.1333	0.3333	0	0	0.2667	0.2667
0.1333	0.2000	0.1333	0.1333	0.0667	0.3333
0.1333	0.1333	0.4667	0.1333	0.1333	0
0	0.1333	0.1333	0.6667	0	0.0667
0	0.1333	0	0.0667	0.6000	0.2000
0.3333	0.0667	0.0667	0	0.1333	0.4000

Pcc =0.4111

Figure III.10 : *Le résultat de la classification avec 25 % des données de test. Apprentissage de Figure III.8*

D'après la Figure III.2, l'erreur diminue mais reste toujours importante (0,23). En utilisant les données de l'apprentissage, nous obtenons Pcc = 0,5074 (Voir Figure III.3). Ce résultat est prévisible car l'apprentissage est mal fait (erreur = 0,23).

Par conséquent, les données sont mal classées. Ceci peut être interprété par le nombre insuffisant d'itérations. Par conséquent, nous devons augmenter le nombre d'itérations.

D'après les Figures III.6, III.7, III.9 et III.10, nous constatons que les vecteurs sont toujours mal classés malgré le nombre important d'itérations (50000 itérations). Un cas particulier pour la classe 5 où l'on note une probabilité supérieure à 0,60.

Ceci peut être interprété par le mauvais choix :

- des paramètres caractéristiques (ondelettes, les coefficients MFCC);
- du type du classifieur;
- des paramètres du réseaux de neurones tel le momentum, le pas d'apprentissage,……

Notant que pour trouver les paramètres optimaux, il faut tâtonner et essayer plusieurs valeurs de façon empirique.

III.5.2 Quelques résultats obtenus par les HMM

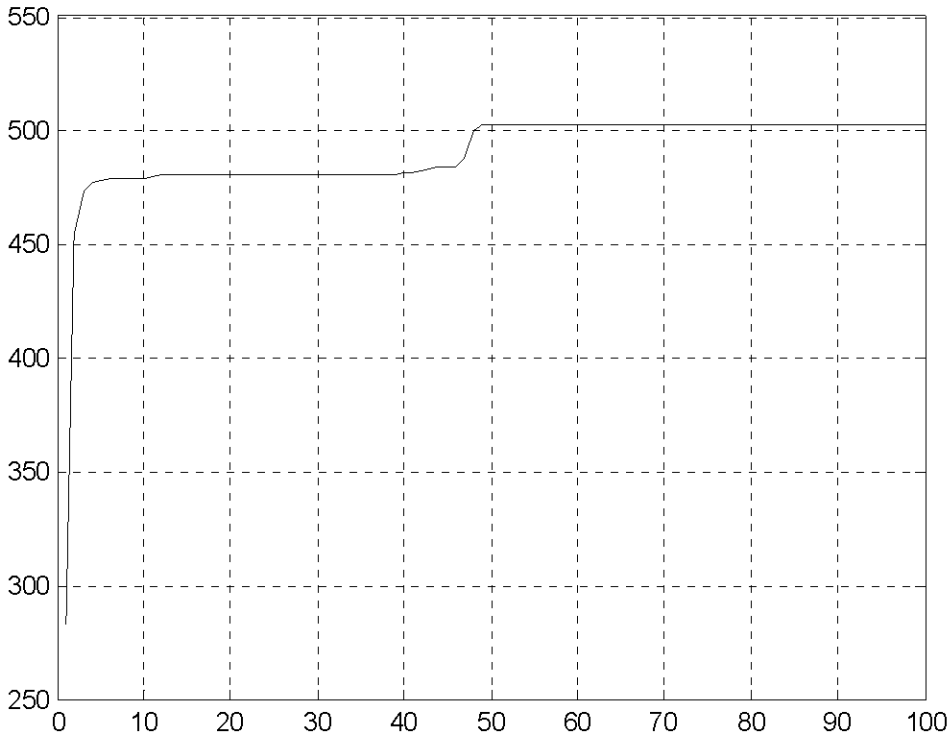

Figure III.11 : *Apprentissage du modèle 1 (classe 1).*

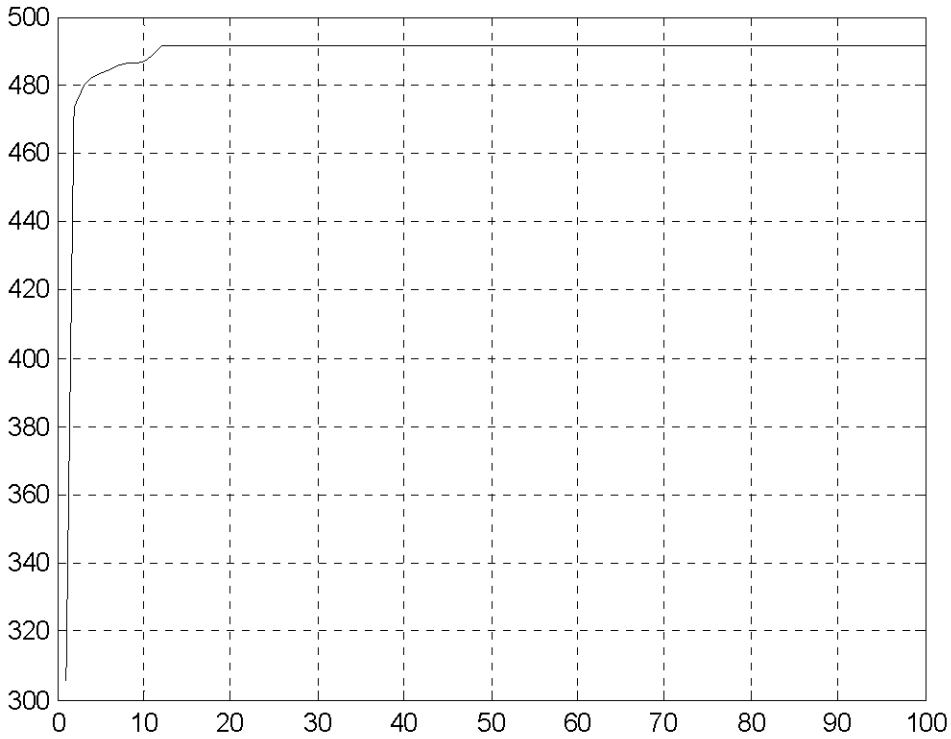

Figure III.12 : *Apprentissage du modèle 2 (classe 2).*

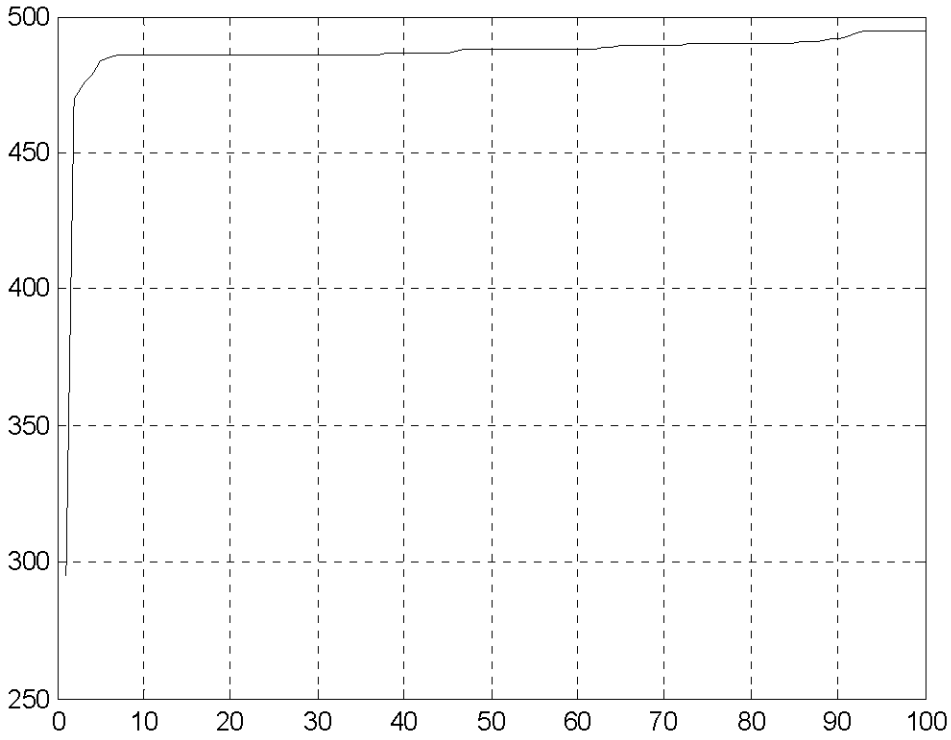

Figure III.13 : *Apprentissage du modèle 3 (classe 3).*

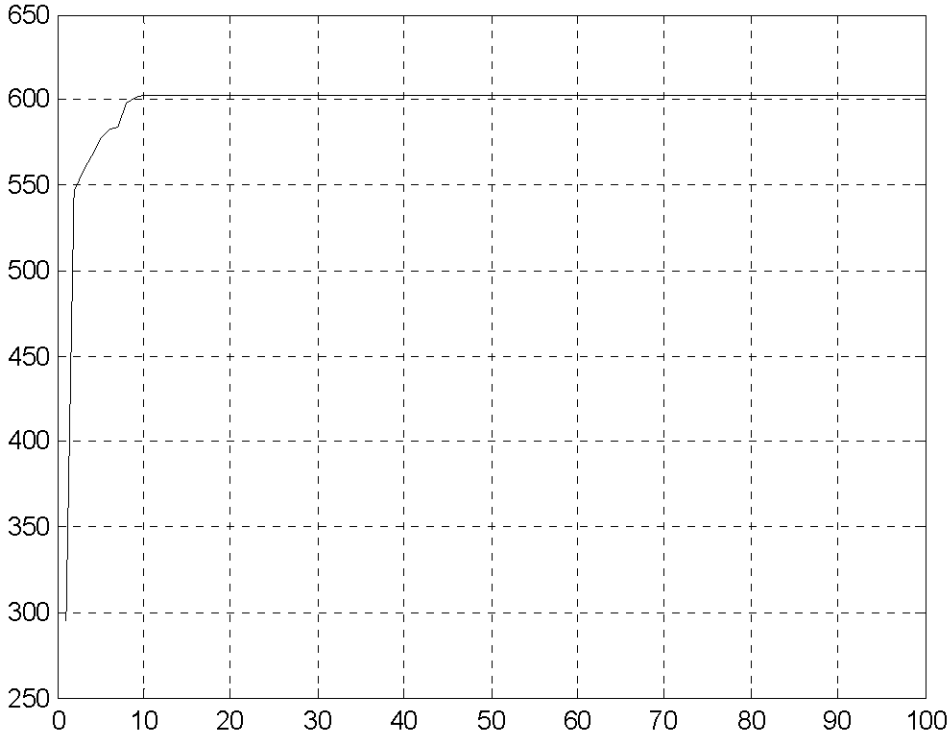

Figure III.14 : *Apprentissage du modèle 4 (classe 4).*

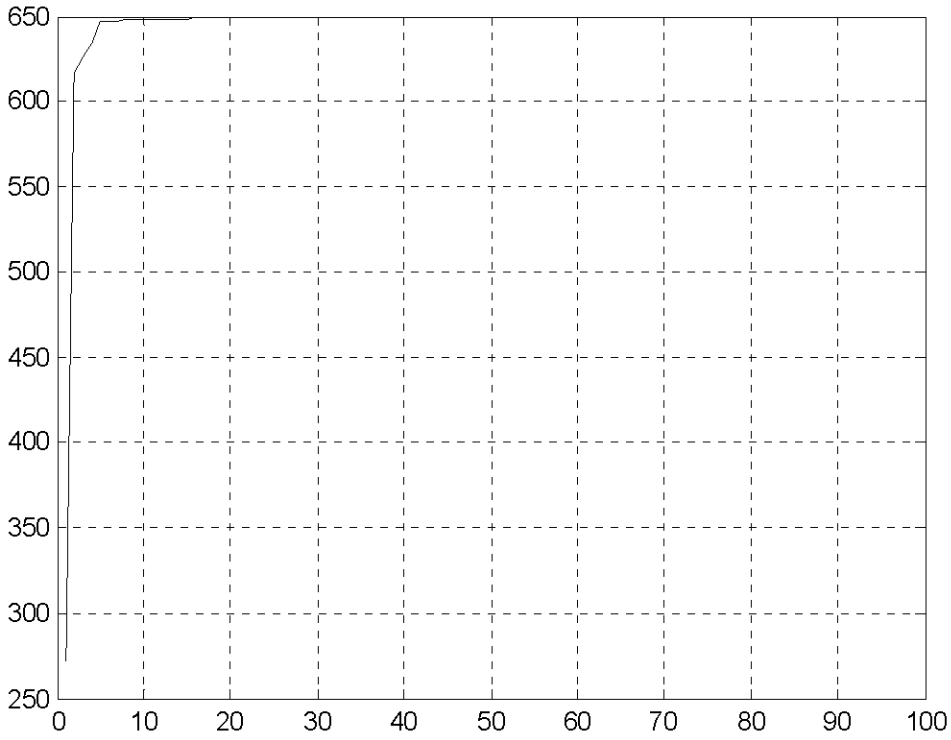

Figure III.15 : *Apprentissage du modèle 5 (classe 5).*

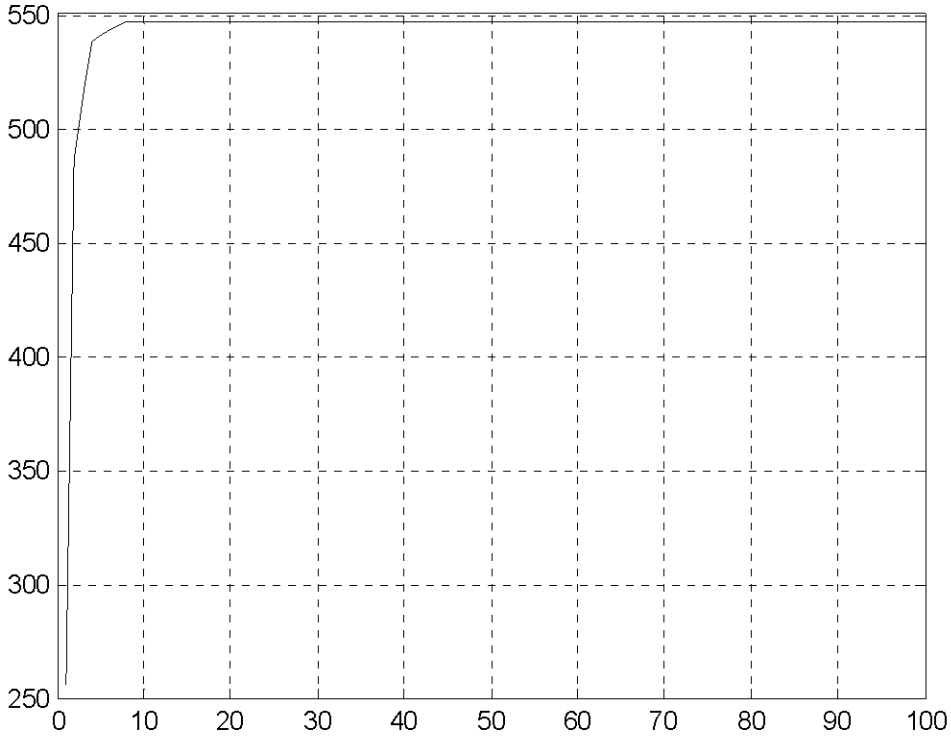

Figure III.16 : *Apprentissage du modèle 6 (classe 6).*

0.8912	0.0011	0	0	0.1076	0.0001
0	1	0	0	0	0
0	0	0.6666	0	0.3334	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

Pcc =0.9263

Figure III.17 : *Le résultat du test avec 75 % des données d'apprentissage. Apprentissage de Figure III.11,……, Figure III.16*

0.8912	0.0011	0	0	0.1077	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

Pcc =0.8152

Figure III.18 : *Le résultat de la classification avec 25 % des données de test. Apprentissage de Figure III.11,……, Figure III.16*

III.6 CONCLUSION

Les résultats obtenus montrent que la méthode des HMM donne de meilleurs résultats que la méthode neuronale en terme de probabilité de classification correcte.

D'après les valeurs des Pcc obtenues, les données que nous avons utilisées ne sont pas discriminantes. Ceci peut être expliqué par des données bruitées, mauvaises conditions d'enregistrement (type de microphone, sonorisation de la salle). De plus, le choix des MFCC comme paramètres caractéristiques, n'est peut être pas judicieux.

De plus la méthode de Rétropropagation est empirique. Il faut exécuter plusieurs fois pour aboutir aux paramètres optimaux (le momentum, le pas d'apprentissage, le nombre d'itérations). L'initialisation des poids synaptiques neuronaux ainsi que les matrices HMM est aléatoire.

BILAN

Le travail proposé a pour thème l'application des réseaux de neurones et les modèles cachés de Marcov pour la reconaissance supervisée de locuteurs.

Concernant l'étape de prétraitement, nous avons utilisé la pondération Hamming, décomposition en ondelettes et l'analyse cepstrale. En effet, cette étape est la plus importante car l'objectif est l'extraction de l'information la plus pertinente.

Dans l'étape de classification, nous avons utilisé un perceptron à une seule couche cachée. Nous avons utilisé également l'algorithme de la Rétropropagation du gradient de l'erreur pour l'apprentissage et un HMM continu pour modéliser les vecteurs.

Les résultats obtenus par les HMM sont satisfaisants. Ceci montre que l'approche statistique modélise au mieux la variabilité intra classes. Contrairement aux réseaux de neurones qui donnent de mauvais résultats.

PERSPECTIVES

La première perspective concerne la banque de données utilisées. En effet, il convient d'utiliser des enregistrements effectués dans de meilleures conditions.

De plus, il serait intéressant d'utiliser une ondelette différente de la Daubechies 1 pour la compression des données.

Concernant la méthode de classification, nous proposons d'utiliser le mélange de gaussiennes (Gaussian Mixture Models - GMM) pour l'apprentissage.

Par souci d'amélioration des performances, nous proposons des méthodes de classification hybrides, exemple Rétropropagation / HMM. Ceci consiste en l'estimation des matrices HMM par les réseaux de neurones au lieu de les initialiser de façon aléatoire.

Enfin, le locuteur à identifier est inconnu en réalité. Par conséquent nous proposons la classification non supervisée en utilisant les réseaux de neurones compétitifs tels que les cartes auto organisatrices (SOM).

BIBLIOGRAPHIE

[1]. YASSINE MAMI, " *Reconnaissance de locuteurs par localisation dans un espace de locuteurs de référence* ". Thèse de doctorat, Octobre 2003. Ecole nationale supérieure des télécommunications -Paris, France.

[2]. CORINNE FREDOUILLE, "*Approche statistique pour la reconnaissance automatique du locuteur : Informations dynamiques et normalisation bayesienne des vraisemblances* ".Thèse de doctorat, Octobre 2000. Université d'Avignon et des pays de Vaucluse - France.

[3]. YASSINE BEN AYED, " *Détection de mots clés dans un flux de parole* ".Thèse de doctorat, Décembre 2003.Ecole national supérieure des télécommunication -Paris, France.

[4]. PIERRE JOURLIN, " *Approche bimodale du traitement automatique de la parole : Application a la reconnaissance du message et du locuteur* ". Thèse de doctorat, Avril 1998. Université d'Avignon et des pays de Vaucluse -France.

[5]. DONGSUK YUK, " *Robust Speech Recognition Using Neural Networks and Hidden Markov Models* ". Thèse de doctorat, Octobre 1999. University of New Jersey.

[6]. SEBASTIEN DEMANGE, " *Contribution a la reconnaissance automatique de la parole avec données manquantes* ". Thèse de doctorat, Novembre 2007. Université Henri Poincaré -Nancy 1, France.

[7]. AUGUSTE-REGIS et AMVAME-BEKALE, " *Reconnaissance multimodale du locuteur* ". Projet d'application, Juillet 2000. Ecole de technologie supérieure, Université du Québec.

[8]. LAWRENCE R. RABINER , "*A tutorial on Hidden Marcov Models and selected applications in speech recognition* ", Proceedings of the IEEE, Vol. 77,N° 2, February 1989.

Document PDF :

[9].TRAITEMENT DE LA PAROLE : compression et codage de la parole.
http://scgwww.epfl.ch/courses

[10].TRAITEMENT DE LA PAROLE : Signal de parole Production – Perception – Analyse.
http://catalogue.ircam.fr/sites/Voix/decrire/index.html

[11]. Introduction au Traitement Automatique de la Parole.
http://tcts.fpms.ac.be/cours/1005-08/speech/

Logiciels téléchargés sur Internet pour l'enregistrement des voix :

- Praat : www.praat.org
- GoldWave : http://www.goldwave.com