Mémoire de Projet de Fin d'Études

Sujet

Réseaux de Neurones Artificiels et Modèles de Markov Cachés

Traitement Automatique de la Parole

Soutenu par :

M. Zakariae EN-NAIMANI

Sous la direction de :

M. Mohamed ETTAOUIL

Année Universitaire 2011-2012

À mes chers parents,

À ma famille,

Remerciements

Au terme de ce travail, je tiens vivement à remercier mon encadrant et cher professeur M. Mohamed ETTAOUIL, d'abord pour l'opportunité qu'il m'a offerte de faire partie de son équipe de travail pendant trois mois. Ensuite, je le remercie pour tous ses précieux conseils et directives, ainsi que toute l'équipe pour sa sympathie, sa patience, sa compréhension et sa disponibilité durant toute cette période.

Je remercie également M. Mohamed Lazaar et Mlle Fidae Harchli , mes chers collègues, qui n'ont épargné ni temps ni effort pour m'aider à concrétiser ce travail.

Table des figures

Table des matières

Introduction

La reconnaissance automatique de la parole donne aujourd'hui lieu à un ensemble important d'applications de nature et de difficulté très variées, concernant quotidiennement des millions de personnes à travers le monde. On peut prévoir que la parole fera de plus en plus partie des interfaces multimédia entre un utilisateur et un système automatique, d'une part grâce à l'amélioration de la robustesse des systèmes de reconnaissance automatique de la parole et, d'autre part, du fait de la sensibilisation croissante du grand public à cette technologie encore peu connue.

Cependant, la reconnaissance automatique de la parole est une problématique aussi intéressante que difficile. Sa difficulté vient justement du fait que la parole est très complexe à traiter et nécessite la prise en compte de plusieurs aspects : acoustique, phonétique, syntaxique etc.

Ce projet à trois chapitres, consiste à aborder la RAP dans ses aspects acoustique et phonétique. Cela dit, nous allons faire un traitement d'extraction des paramètres d'un signal parole qui nous permettra de faire la reconnaissance de ces paramètres en utilisant une approche probabiliste par le modèle de Markov allié à un réseau de neurones de type perceptron multicouches.

En effet, dans le premier chapitre nous allons voir les difficultés que présente le traitement du signal parole, dont la variabilité du signal. Pour remédier à ces difficultés nous procédons à une approche aléatoire, ensuite pour extraire les paramètres représentatifs du signal, nous appliquons l'échantillonnage et la quantification. Enfin la dernière partie du chapitre traite la phase de décodage des informations acoustiques.

Le second chapitre englobe, dans une première partie, une introduction aux réseaux de neurones artificiels. Dans une seconde, nous allons voir en particulier le modèle d'un réseau de neurones nommé perceptron multicouches. Nous allons définir sa topologie, et réaliser son apprentissage par la méthode de rétropropagation du gradient. Et enfin, dans une troisième

partie, nous allons présenter un programme que nous avons développé en langage C, qui réalise l'apprentissage du perceptron multicouche par la méthode de rétropropagation du gradient.

Le troisième est dédié à l'étude des Modèles de Markov, et plus précisément les modèles de Markov cachés qui interviennent au niveau de la reconnaissance d'une séquence qui peut être partiellement erronée. Ce chapitre traite également les différents problèmes associés au MMC et présente les algorithmes liés et leur implémentation dans une application programmée au langage C.

Le quatrième et dernier chapitre, nous allons présenter les différents schémas de couplage neuraux-markoviens proposés dans la littérature. L'objectif du chapitre est de présenter dans un premier temps le mécanisme et l'intérêt d'une combinaison réseaux de neurones artificiels et modèles de Markov cachés, puis dans un second temps les méthodes d'apprentissages de ces modèles hybrides.

Chapitre 1

Reconnaissance Automatique de la Parole (RAP)

1.1 Introduction

La Reconnaissance Automatique de la Parole intéresse un nombre important de scientifiques depuis les années 1950. La parole est le support de communication essentiel de l'être humain, et c'est la raison pour laquelle il tente toujours de le reproduire avec des machines. Cependant, le signal de la parole s'avère très difficile à traiter, de par la complexité des aspects qui s'y mêlent. En effet, l'information portée par le signal peut être analysée de beaucoup de façons. On en distingue généralement plusieurs niveaux de description : acoustique, phonétique, phonologique, morphologique, syntaxique, sémantique et pragmatique. Dans ce mémoire, nous allons mettre l'accent sur l'analyse acoustique.

Ce premier chapitre comporte donc, dans un premier temps, l'étude du signal de la parole et les difficultés qu'il relève, et dans un deuxième l'extraction des paramètres du signal par analyse acoustique et décodage des informations acoustiques.

1.2 Système de reconnaissance de la parole

Un système de reconnaissance de la parole (Figure 1), peut être basé sur deux modèles :

- Le modèle d'analyse acoustique : qui consiste à extraire une suite de vecteurs d'observations qui contiennent les informations caractéristiques et pertinentes du signal de parole, tout en éliminant la redondance.
- Le modèle de décodage des informations : qui consiste à comparer les informations issues de l'analyse acoustique aux informations de références pour déterminer la phrase prononcée.

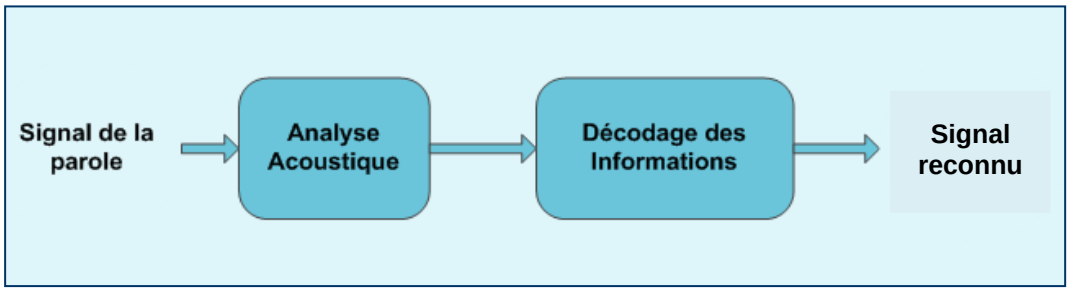

Figure 1 : Système de reconnaissance de la parole

1.3 Signal de la parole

Le signal de parole est une onde acoustique qui se propage dans un milieu donné (en général l'air) et qui est le résultat de la modulation par le conduit vocal d'une onde d'excitation. Contrairement à d'autres vecteurs d'information (image, vidéo, etc.), la parole n'est pas un signal ordinaire. Il détient sa complexité de ses caractéristiques qui rendent difficile sa modélisation.

Nous allons essayer, dans cette partie, de mettre en évidence les plus importantes de ces caractéristiques afin de faire ressortir les problèmes posés lors de son traitement.

1.3.1 Redondance du signal

Le signal de la parole est un signal redondant. Cette redondance des indices dans le signal est informative et elle apporte une robustesse à la transmission. De plus, elle assure la non altération de l'intelligibilité en cas d'absence ou de déformation de ces indices.

1.3.2 Variabilité du signal

La variabilité est l'une des plus importantes caractéristiques du signal de la parole. Ce dernier dépend de l'état de l'appareil phonatoire, qui est conditionné par plusieurs facteurs, et qui influe fortement sur la façon de prononciation des mots. En effet, la voix peut être modifiée pour plusieurs raisons, notamment la fatigue, le rhume, l'évolution d'âge, etc., ce qui confère

au signal de la parole un caractère de variabilité qui ne peut être négligé lors de son traitement.

1.3.3 Effets de coarticulation

Le signal de la parole n'est pas séquentiel. Les mots prononcés subissent l'influence de ceux qui les précèdent et de ceux qui les suivent. Ce qui rajoute au signal un facteur de variabilité supplémentaire.

1.4 Analyse acoustique

L'objectif de cette phase, qui s'inscrit dans la démarche de la reconnaissance automatique de la parole (RAP), est d'extraire des coefficients représentatifs du signal parole. Le signal de parole est transformé en une série de vecteurs de coefficients, qui sont calculés à intervalles temporels réguliers.

Le signal de parole tel que décrit dans la partie **1.1** présente, dans le domaine temporel, une redondance qui rend indispensable un traitement préalable à toute tentative de Cela dit, les méthodes d'analyse acoustique se déclinent en trois phases (Figure 2) :

- Un filtrage analogique en sortie du microphone ;
- Une conversion analogique (numérique) ;
- Un calcul des coefficients.

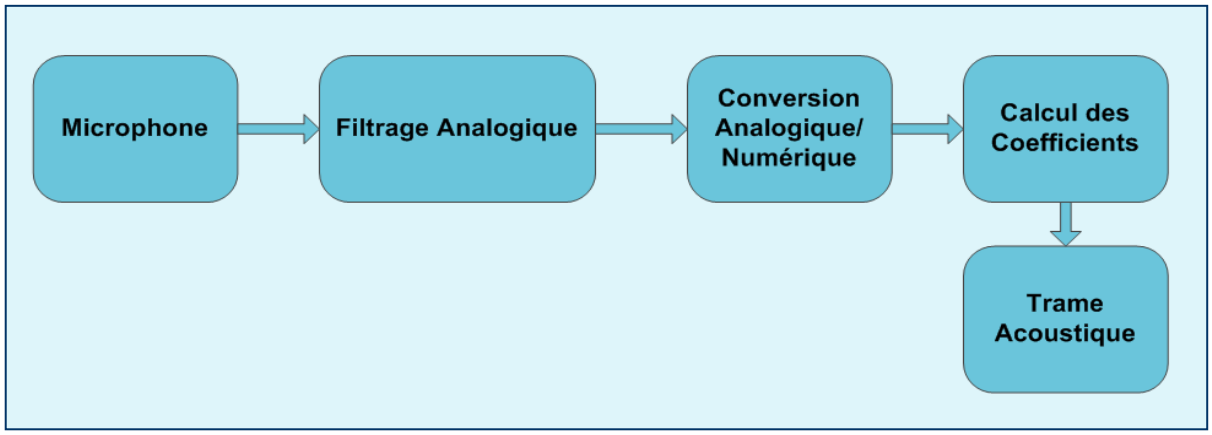

Figure 2 : Analyse acoustique du signal de la parole

1.4.1 Numérisation du signal parole

La plupart des signaux que l'on doit traiter et analyser tels que la parole, audio ou vidéo, sont analogique par nature. Pour des raisons de simplicité, de précision, de stockage de l'information, etc., un traitement numérique est possible et préférable. Pour cela nous allons utiliser des convertisseurs analogiques-numériques (A-N), on peut considérer la conversion (A-N) comme un processus faisant intervenir deux actions successives : l'échantillonnage et la quantification du signal.

Lors du traitement d'un signal analogique nous allons procéder comme suit :

1.4.1.1 Filtrage analogique en sortie du microphone

Comme l'information acoustique pertinente du signal se situe dans la bande passante [50Hz, 8KHz], ce filtrage élimine toutes les composantes du signal en dehors de cette bande passante.

1.4.1.2 Échantillonage

Le signal d'entrée $x(t)$, dont l'amplitude varie au cours du temps est appliqué à un échantillonneur pour être transformé en une suite de valeurs régulièrement espacées (Figure 3). Cette suite de valeurs est représentative du signal dans la mesure où la période d'échantillonnage est compatible avec la rapidité du signal.

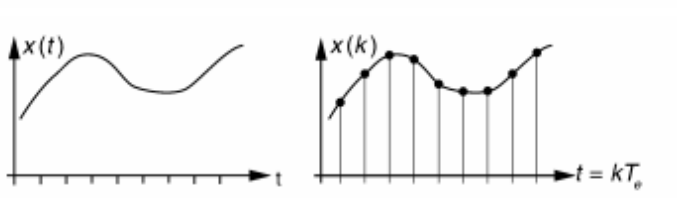

Figure 3 : Signal analogique numérisé

Comme envisagé dans le domaine temporel, la phase de l'échantillonnage (Figure 4) peut être considérée mathématiquement comme un produit du signal analogique $x(t)$ par une suite d'impulsion de Dirac $\delta_{Te}(t)$ qui peut alors être représentée par l'expression :

$$X_e(t) = x(t).\delta_{Te}(t) \qquad (1.1)$$

Où $\delta_{Te}(t)$ est la peigne de Dirac définie par :

$$\delta_{Te}(t) = \sum_{-\infty}^{+\infty} \delta(t - nT_e) \qquad (1.2)$$

L'impulsion de Dirac $\delta(t)$ est définie par :

$$\begin{cases} \text{EMBED Equation.DSMT4} & si\ t \neq 0 \\ \text{EMBED Equation.DSMT4} & \end{cases} \qquad (1.3)$$

Voyons immédiatement deux propriétés importantes concernent l'impulsion de Dirac :

$$\int_{-\infty}^{+\infty} \delta(t).x(t)dt = x(0) \qquad (1.4)$$

Et ce pour tout signal $x(t)$ continu en 0.

Une impulsion en $t = t_0$ sera modélisée par $\delta(t - t_0)$ et on a $\int_{-\infty}^{+\infty} \delta(t - t_0).x(t)dt = x(t_0)$.

Bien entendu, les propriétés ci-dessus impliquent que l'impulsion de Dirac ne peut pas être une fonction ordinaire donc c'est une application multivoque.

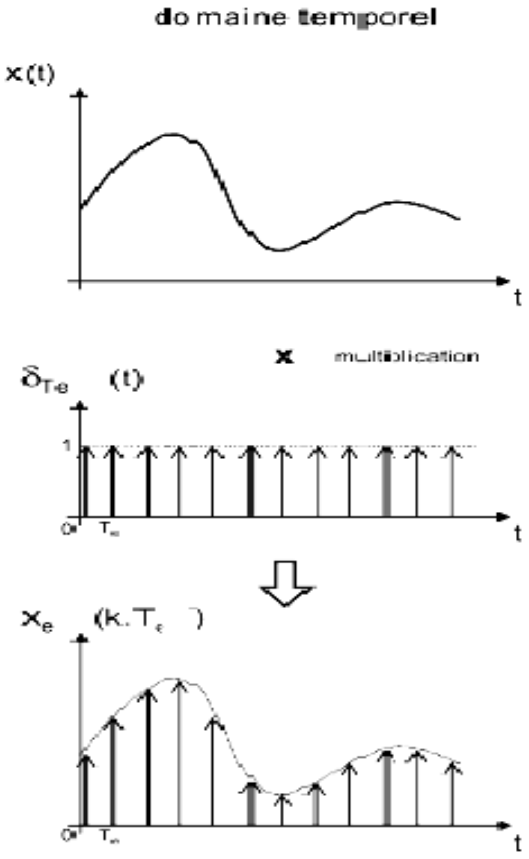

Figure 4 : Échantillonnage d'un signal

- Choix de la période d'échantillonnage

Si on veut respecter la forme du signal, il est important d'avoir des impulsions suffisamment proches les unes des autres. Dans le cas contraire, il n'est plus possible d'avoir les variations les plus rapides du signal à traiter (Figure 5).

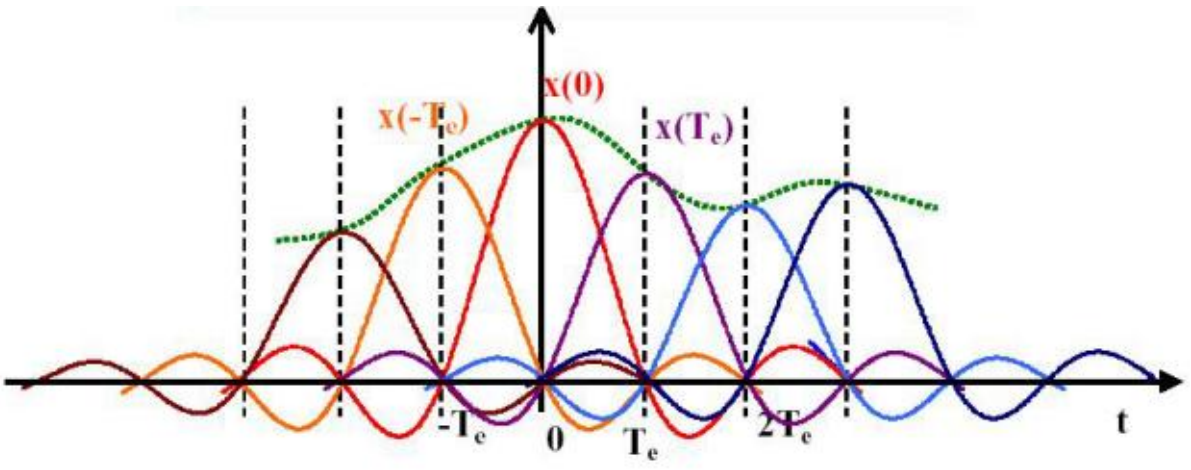

Figure 5 : Ambigüité due à l'échantillonnage

La figure 5 montre que si la période d'échantillonnage T_e est mal choisie, on peut extraire la même information due à deux signaux différents.

En 1948, Shannon a montré que, pour éviter ces problèmes, il suffit de satisfaire :

$$f_e \succ 2 f_{\max} \Leftrightarrow T_e \prec \frac{T_{\min}}{2} \qquad (1.5)$$

Avec :

- f_e : fréquence d'échantillonnage.
- $f_{\max}$: Fréquence maximale dans le signal de parole.
- T_e : Période d'échantillonnage.
- $T_{\min}$: Période minimale dans le signal de parole.

- **Théorème de Shannon**

Un signal $x(t)$ peut être représenté de manière univoque par une suite de valeurs échantillonnées si la fréquence d'échantillonnage f_e est au moins 2 fois plus élevée que la plus grande des fréquences contenues dans le signal.

1.4.1.3 Quantification du signal échantillonné

La quantification d'un signal échantillonné est une étape de numérisation. Elle consiste à remplacer un nombre réel par l'entier le plus proche. La quantification la plus couramment utilisée en audio grand public est de type linéaire, son pas est en rapport avec le nombre de bits alloués à l'opération :

Supposant que le convertisseur Analogique Numérique (CAN) fonctionne avec n bits, le pas de quantification (quantification uniforme) vaut alors :

$$q = \frac{\nabla_{CAN}}{2^n} = \frac{2U_{\max}}{2^n} \qquad (1.6)$$

Où :

- n : nombre de bits.

Il existe plusieurs types de quantification : quantification instantanée, non uniforme, quantification avec compression et expansion, etc.

1.4.2 Méthodes d'analyse acoustique

Il existe trois types de méthodes d'analyse acoustique utilisées dans des systèmes de reconnaissance de la parole :

- Méthode de perception

Obtenu à partir des recherches en physiologie et psycho acoustique, le principe de ce modèle consiste à définir des bandes critiques de perception, correspondant à la distribution fréquentielle de l'oreille humaine. Les coefficients sont les énergies en sortie d'un banc de filtre calibré à partir de ces résultats : cette technique est utilisée dans les vocodeurs à canaux (Figure 6). Dans les systèmes de RAP les modèles de perception restent peu utilisés.

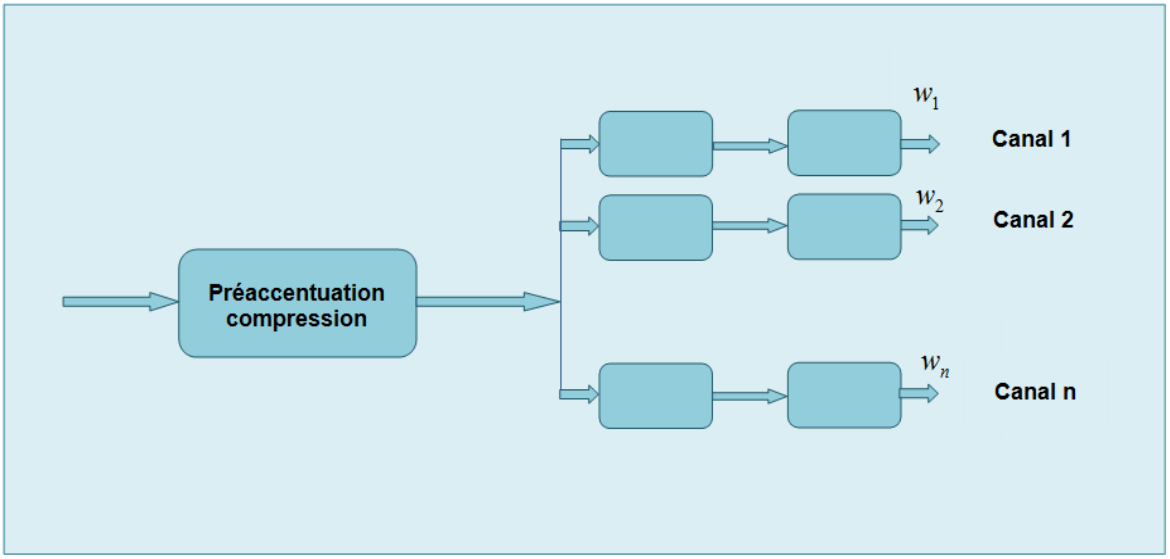

Figure 6 : Schéma classique d'un vocodeur à canaux

L'estimation du spectre est donnée par la suite de valeurs $w_1, w_2, ..., w_n$ correspond aux énergies sortant à un instant donné des n canaux du vocodeur.

- Méthode non paramétrique

L'une des techniques les plus utilisées est basée sur l'analyse par la transformée de Fourier discrète, implémentée grâce à son algorithme de calcul rapide, à savoir la FFT (Fast Fourier Transform), elle permet d'obtenir des spectres en temps réel. En effet, l'analyse par FFT se fait en trois étapes :

- Un filtre de préaccentuation est appliqué à fin d'égaliser les aigus toujours plus faibles que les graves en énergie ;
- Un fenêtrage de type Hamming est effectué sur chaque bloc d'analyse pour limiter les effets de bords lors du spectre ;
- Une FFT est calculée.

- Méthode paramétrique

Les méthodes précédentes ne font pas d'hypothèses sur le signal analysé, par contre les méthodes paramétriques tiennent compte du processus de phonation et supposent que le modèle de production est linéaire. L'onde vocale est modélisée comme la sortie d'un filtre linéaire causal excité par source.

Donc l'analyse par prédiction linéaire LPC (Linear Prediction Coding) peut être simplifiée, en supposant que le filtre est linéaire tout Pôles et se réduit à un modèle autorégressif (AR).

Dans le domaine temporel, sous l'hypothèse AR, le signal obéit à la relation suivante :

$$x(n) = \sum_{i=1}^{p} a_i x(n-i) + e_n \qquad (1.7)$$

Où : - e_n : correspond à la source ;

- a_i : sont les coefficients de prédiction.

Chaque échantillon $x(n)$ peut être prédit à partir des p échantillons précédents.

Soit $\hat{x}(n)$ la prédiction de $x(n)$ qui est donnée par :

$$\hat{x}(n) = \sum_{i=1}^{p} a_i x(n-i) \qquad (1.8)$$

En appliquant la transformée en z, on obtient :

$$x(z) = \frac{e(z)}{A(z)} \qquad (1.9)$$

Avec :

$$A(z) = 1 + a_1 z^{-1} + ... + a_p z^{-p} \qquad (1.10)$$

Les coefficients a_i décrivent la fonction de transfert $\frac{1}{A(z)}$ du conduit vocal. En utilisant les équations précédentes, a_i est considérée comme une erreur de prédiction :

$$e_n = x(n) - \hat{x}(n) \qquad (1.11)$$

L'estimation du modèle est basée sur la détermination des coefficients a_i optimaux par minimisation de la variance σ^2 de l'erreur de prédiction supposée indépendante du temps.

1.5 Décodage des informations acoustiques

1.5.1 Généralités

En reconnaissance automatique de la parole nous distinguons trois approches : l'approche analytique, l'approche globale et l'approche statistique. La première approche utilise une segmentation a priori du signal en unités de taille phonétique, la deuxième approche et la troisième approche consistent à identifier globalement un mot ou une phrase en les comparant, soit à des références acoustiques en utilisant la comparaison dynamique DTW (Dynamic Time Warping) pour l'approche globale, soit à des modèles Markoviens de référence dans l'approche statistique. Dans ce projet, nous nous intéressons à l'approche statistique de la parole.

1.5.2 Approche probabiliste ou statistique de la RAP

Dans le cadre d'une application de la reconnaissance automatique de la parole, trois facteurs principaux interviennent (Figure 7) :

- Le locuteur, qui à partir d'un message m (suites de mots) qu'il veut transmettre produit un signal acoustique $s(t)$;
- L'analyseur acoustique, qui à partir du signal $s(t)$ produit une paramétrisation sous forme d'une suite de vecteurs (séquence d'observations o) contenant l'information pertinente pour la reconnaissance ;
- Un décodeur dont le rôle consiste à déterminer, à partir de la séquence d'observations O, la séquence de mots $\hat{m}$ qui correspond au message m.

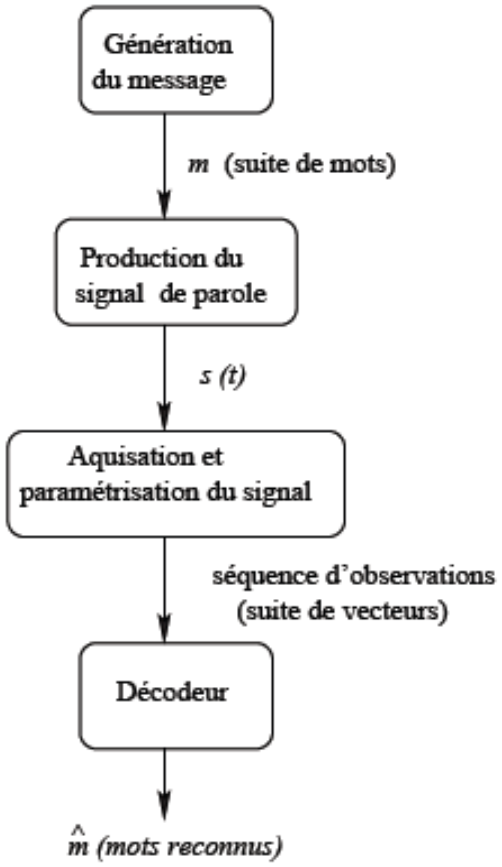

Figure 7 : Les composantes principales du processus de la RAP

La reconstitution d'un message m inconnu à partir d'une séquence d'observations O, consiste à retrouver, parmi tous les messages possibles, celui qui selon toute vraisemblance, correspond à O. L'utilisation de la règle de Bayes permet de décomposer la probabilité $P(m|o)$ en deux composantes :

$$\hat{m} = \arg_m \max P(m \mid o) = \arg_m \max \frac{P(m)P(o \mid m)}{P(o)} \qquad (1.12)$$

Le dénominateur est constant pour tous les messages possibles, donc on peut l'omettre et $\hat{m}$ sera alors écrit sous la forme suivante :

$$\hat{m} = \arg_m \max P(m)P(o \mid m) \qquad (1.13)$$

Ainsi, l'étape de reconnaissance consiste à déterminer la suite de mots $\hat{m}$ qui maximise le produit des deux termes $P(m)$ et $P(o \mid m)$. Le premier terme représente la probabilité a priori d'observer la suite de mots m indépendamment du signal. Cette probabilité est déterminée par le modèle de langage. Le deuxième terme indique la probabilité d'observer la séquence de vecteurs acoustiques O sachant une séquence de mots m. Cette probabilité est estimée par le modèle acoustique. La qualité d'un tel système de reconnaissance de la parole peut être caractérisée par la précision et la robustesse des deux modèles qui permettent de calculer ces deux termes $P(m)$ et $P(o \mid m)$.

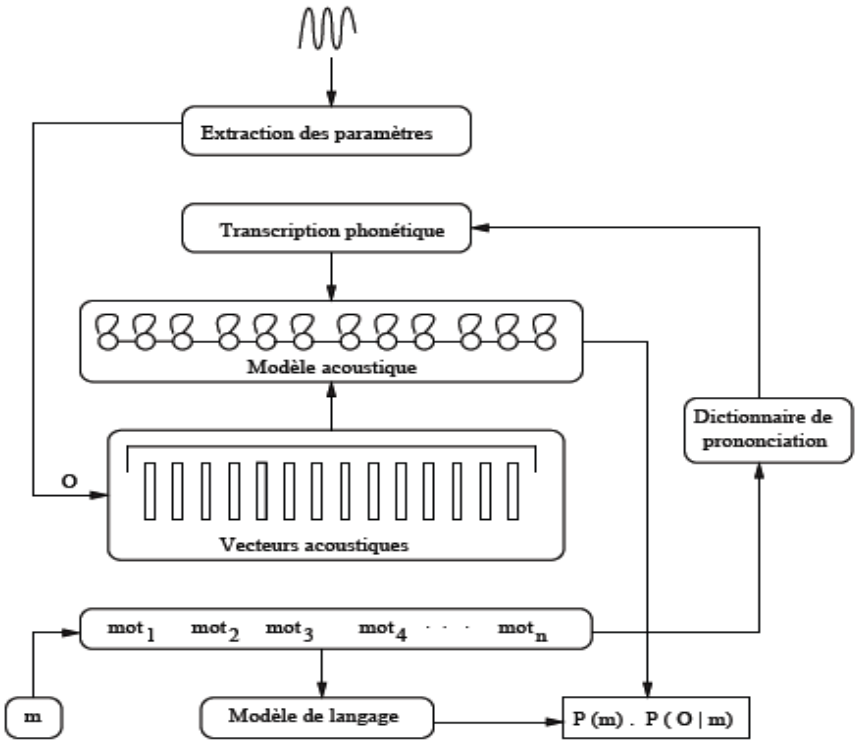

Figure 8 : L'approche probabiliste de la Reconnaissance Automatique de la Parole

L'outil statistique le plus utilisé et le plus performant, de nos jours, pour la modélisation acoustique est fondé sur les modèles de Markov cachés (MMC). Les différentes étapes nécessaires à la reconnaissance d'une hypothèse donnée sont illustrées par la figure 8. Tout d'abord le signal de parole est subdivisé pour construire une séquence de vecteurs acoustiques. En utilisant ces vecteurs, le modèle acoustique se charge, à partir des MMC de phonèmes appris sur un corpus d'apprentissage, de construire la suite des phonèmes hypothèses du signal prononcé. Un seul modèle MMC, représentant l'hypothèse, sera construit par la concaténation de l'ensemble des MMC de phonèmes qui la compose et génère ainsi la probabilité du signal $s(t)$, ce qui définit la probabilité $P(o \mid m)$. Ainsi, à partir du dictionnaire des prononciations, la suite des mots hypothèses sera déterminée. Cette suite de mots sera évaluée par le modèle de langage pour estimer la probabilité $P(m)$. En principe, ce processus est

répété pour toutes les hypothèses possibles. Le système donne Enfin la meilleure hypothèse comme résultat de la reconnaissance.

L'espace de toutes les séquences de mots m augmente très rapidement avec la taille du vocabulaire. Il convient donc de restreindre la recherche à l'espace des séquences de mots les plus plausibles. Les applications récentes en reconnaissance de la parole utilisent souvent des modèles de langage stochastiques. Un modèle de langage est un automate à états finis dont les états représentent les mots du vocabulaire et les arcs les probabilités conditionnelles des transitions. Ces probabilités sont apprises sur des corpus de textes de l'application en question.

Considérons le cas d'une séquence m constituée de la suite des mots m_i avec $i \in \{1,...,L\}$. La probabilité est donnée par :

$$P(m) = P(m_1 m_2 ... m_L) = P(m_1) \prod_{i=2}^{L} P(m_i \mid m_1 m_2 ... m_{i-1}) \qquad (1.14)$$

1.6 Conclusion

Dans ce chapitre, nous avons donné la définition d'un signal parole et les difficultés qui y sont liées. De plus, nous avons présenté les méthodes pour extraire les paramètres représentant ce signal. Ces étapes étant l'échantillonnage et la quantification et quelques méthodes d'analyse acoustique. Et finalement nous sommes passés au décodage des informations obtenues.

Chapitre 2

Réseaux de Neurones Artificiels (RNA)

2.1 Introduction

Un réseau de neurones artificiel est une structure composée d'entités capables de calcul et interagissant entre eux, à savoir les neurones. Il permet de traiter, par le biais de l'informatique, des problèmes de différentes natures que les outils classiques ont du mal à résoudre. En effet, son fonctionnement s'inspire de celui des cellules neuronales animales, et est donc différent des méthodes de calcul analytiques que l'on utilise ordinairement. Il s'avère très puissant dans des problèmes de reconnaissance, classification, approximation ou prévision.

Dans ce chapitre, après avoir énoncé les différentes caractéristiques d'un réseau de neurones, nous allons nous intéresser au cas particulier du perceptron multicouche.

2.2 Généralités sur les RNA

2.2.1 Modèles d'un neurone

2.2.1.1 Neurone biologique

Le neurone biologique est une cellule vivante spécialisée dans le traitement des signaux électriques. Les neurones sont reliés entre eux par des liaisons appelées axones. Ces axones vont eux mêmes jouer un rôle important dans le comportement logique de l'ensemble. Ces axones conduisent les signaux électriques de la sortie d'un neurone vers l'entrée (synapse) d'un autre neurone.

Les neurones font une sommation des signaux reçus en entrée et en fonction du résultat obtenu vont fournir un courant en sortie.

Un neurone (Figure 9) se compose de trois parties :

- La somma : ou cellule d'activité nerveuse, au centre du neurone ;
- L'axone : attaché au somma qui est électriquement actif, ce dernier conduit l'impulsion conduite par le neurone ;

- Dendrites : électriquement passives, elles reçoivent les impulsions d'autres neurones.

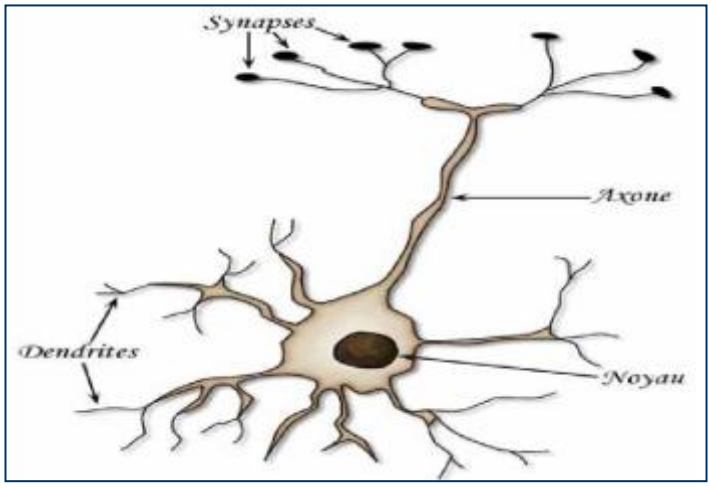

Figure 9 : Modèle d'un neurone biologique

2.2.1.2 Neurone formel

Un neurone formel (Figure 10) est constitué d'un intégrateur qui effectue la somme pondérée de ses entrées $X = (x_1, x_2, ..., x_N)$. Le résultat de cette somme, traduit par la relation (2.1), est ensuite transformé par une fonction de transfert f qui produit la sortie du neurone $y_j = f(v_j)$.

$$v_j = \sum_{i=1}^{N} w_{j,i} x_i - b_j \qquad (2.1)$$

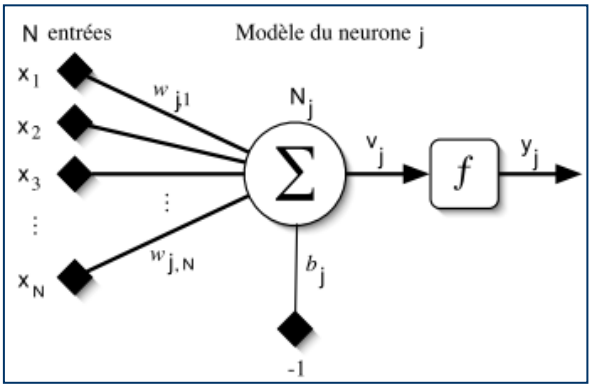

Figure 10 : Modèle d'un neurone formel

Cette sortie correspond à une somme pondérée des poids w_{ji} où $i = 1,...,N$ et des entrées moins ce qu'on nomme le biais (seuil d'activation) b_j du neurone j. Le résultat v_j s'appelle « niveau d'activation du neurone j ». Lorsque le niveau d'activation atteint ou dépasse le seuil b_j, alors l'argument de f devient positif (ou nul). Sinon, il est négatif.

2.2.2 Structure d'un réseau de neurones

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité.

2.2.2.1 Réseau monocouche

La structure d'un réseau monocouche (Figure 11) est telle que des neurones organisés en entrée soient entièrement connectés à d'autres neurones organisés en sortie par une couche modifiable de poids.

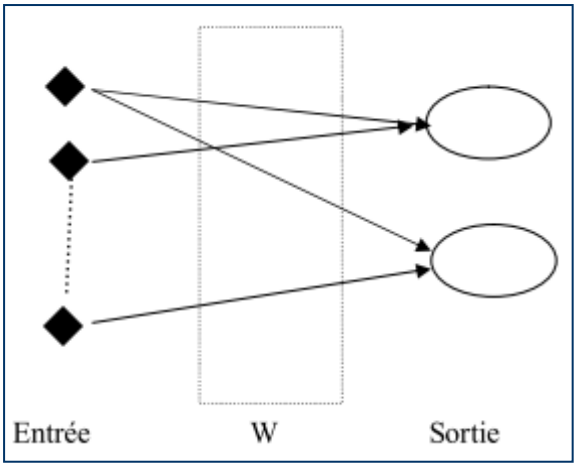

Figure 11 : Modèle d'un RNA monocouche

2.2.2.2 Réseau multicouche

La structure d'un réseau multicouche (Figure 12) est telle que les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche, et les connexions ne se font qu'avec les neurones de couches avales. Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet

d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc de définir les concepts de neurone d'entrée et neurone de sortie. Par extension, on appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelées couches cachées.

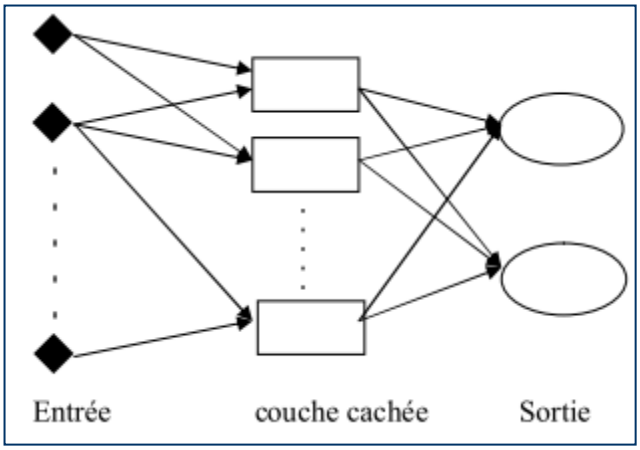

Figure 12 : Modèle d'un RNA multicouche

2.2.2.3 Réseau à connexions locales

Il s'agit d'une structure multicouche, mais qui à l'image de la rétine conserve une certaine topologie. Chaque neurone entretient des relations avec un nombre réduit et localisé de neurones de la couche avale. Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique (Figure 13).

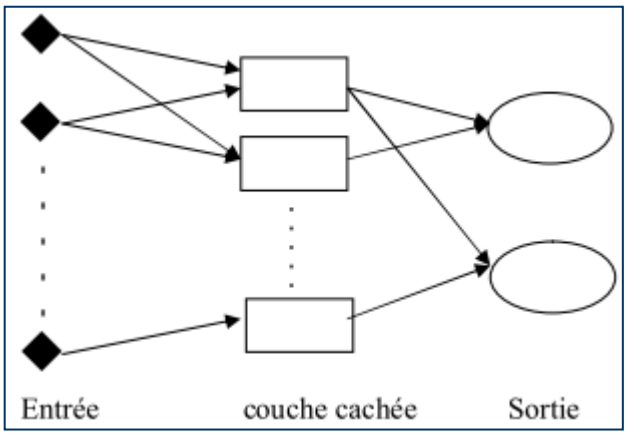

Figure 13 : Modèle d'un RNA à connexions locales

2.2.2.4 Réseau à connexion complète

C'est la structure d'interconnexion la plus générale (Figure 14). Chaque neurone est connecté à tous les neurones du réseau (et à lui-même).

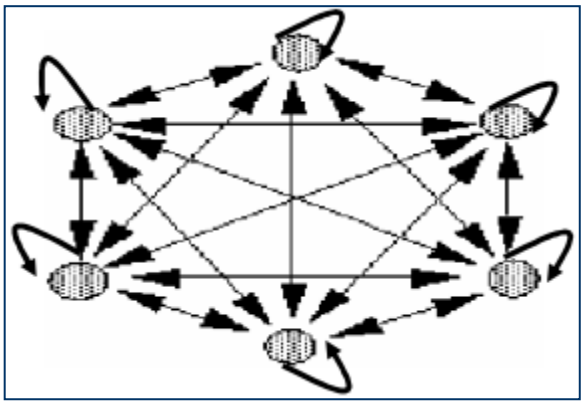

Figure 14 : Modèle d'un RNA à connexion complète

2.2.3 Modèles des réseaux de neurones

2.2.3.1 Modèle de Hopfield

Le modèle de Hopfield fut présenté en 1982. Ce modèle très simple est basé sur le principe des mémoires associatives. C'est d'ailleurs la raison pour laquelle ce type de réseau est dit associatif (par analogie avec le pointeur qui permet de récupérer le contenu d'une case mémoire).

Le modèle de Hopfield utilise l'architecture des réseaux entièrement connectés et récurrents (dont les connexions sont non orientées et où chaque neurone n'agit pas sur lui-même). Les sorties sont en fonction des entrées et du dernier état pris par le réseau.

2.2.3.2 Modèle de Kohonen

Ce modèle a été présenté par Kohonen en 1982 en se basant sur des constatations biologiques. Il a pour objectif de présenter des données complexes et appartenant généralement à un espace discret de grandes dimensions dont la topologie est limitée à une ou deux dimensions. Les cartes de Kohonen sont réalisées à partir d'un réseau à deux couches, une en entrée et une en sortie. Notons que les neurones de la couche d'entrée sont entièrement

connectés à la couche de sortie. Les neurones de la couche de sortie sont placés dans un espace d'une ou de deux dimensions en général, chaque neurone possède donc des voisins dans cet espace. Et qu'enfin, chaque neurone de la couche de sortie possède des connexions latérales récurrentes dans sa couche (le neurone inhibe les neurones éloignés et laisse agir les neurones voisins).

2.2.3.3 Modèle perceptron

Le mécanisme perceptron fut inventé par le psychologue FRANK Rosenblat à la fin des années 50. Il représentait sa tentative d'illustrer certaines propriétés fondamentales des systèmes intelligents en général. Le réseau dans ce modèle est formé de trois couches : Une couche d'entrée (la rétine), fournissant des donnés à une couche intermédiaire, chargée des calculs, cela en fournissant la somme des impulsions qui lui viennent des cellules auxquelles elle est connectée, et elle répond généralement suivant une loi définie avec un seuil, elle-même connectée à la couche de sortie (couche de décision), représentant les exemples à mémoriser. Seule cette dernière couche renvoie des signaux à la couche intermédiaire, jusqu'à ce que leurs connexions se stabilisent.

2.2.3.4 Modèle Adaline

L'adaline (Adaptatif Linear Neurone) de Widrow et Hoff est un réseau à trois couches : la première est la couche d'entrée, la seconde est la couche cachée et la dernière est la couche de sortie. Ce modèle est similaire au modèle de perceptron, seule la fonction de transfert change, mais reste toujours linéaire : $f(x) = x$. Les modèles des neurones utilisés dans le perceptron et l'adaline sont des modèles linéaires.

Séparation linéaire : on dit que deux classes A et B, sont linéairement séparables si on arrive à les séparer par une droite coupant le plan en deux.

Le problème est résolu avec les réseaux multicouches, car il peut résoudre toute sorte de problèmes qu'ils soient linéairement séparables ou non.

2.2.4 Apprentissage

L'apprentissage est l'une des phases importante pour la conception d'un réseau de neurones, car il permet un système RNA d'apprendre son environnement, et d'améliorer sa performance.

2.2.4.1 Apprentissage par correction d'erreur

La première règle que l'on peut utiliser est fondée sur la correction de l'erreur calculée en sortie. Soit d_j la sortie désirée du neurone j, alors y_j et d_j seront généralement différents et il est naturel de calculer l'erreur e_j, donnée par la relation (2.2), entre ce qu'on obtient et ce qu'on voudrait obtenir.

$$e_j(t) = (d_j - y_j)(t) \qquad 1 \le j \le N \tag{2.2}$$

On peut l'écrire sous forme matricielle :

$$e(t) = (d - y)(t) \tag{2.3}$$

Donc l'apprentissage par correction des erreurs consiste à minimiser un indice de performance q basé sur les signaux d'erreur e_j, dans le but de faire converger les sorties du réseau avec ce que nous voulons. Un critère très populaire est la somme des erreurs quadratiques :

$$q(e(t)) = \sum_{j=1}^{N} {e_j}^2(t) \tag{2.4}$$

2.2.4.2 Apprentissage supervisé

L'apprentissage dit supervisé est caractérisé par la présence d'un « professeur » qui possède une connaissance approfondie de l'environnement dans lequel évolue le réseau de neurones. En pratique, les connaissances de ce professeur prennent la forme d'un ensemble de N couples de vecteurs d'entrée et de sortie que nous noterons $\{(x_1, d_1); (x_2, d_2); ...; (x_N, d_N)\}$ où x_j désigne une entrée et d_j la cible de cette entrée, c'est-à-dire les sorties désirées du réseau.

Chaque couple (x_j, d_j) correspond donc à un cas d'espèce de ce que le réseau devrait produire (la cible) pour un stimulus donné. Pour cette raison, l'apprentissage supervisé est aussi dit d'apprentissage par des exemples.

2.2.4.3 Apprentissage compétitif

L'apprentissage compétitif consiste à faire entrer en compétition les neurones d'un réseau pour déterminer lequel sera actif à un instant donné. Contrairement aux autres types d'apprentissage où, généralement, tous les neurones peuvent apprendre simultanément et de la même manière, l'apprentissage compétitif produit un « vainqueur » ainsi que, parfois, un ensemble de neurones « voisins » du vainqueur, et seuls ce vainqueur et, potentiellement, son voisinage bénéficient d'une adaptation de leur poids. On dit alors que l'apprentissage est local car limité à un sous-ensemble des neurones du réseau.

Une règle d'apprentissage compétitif comporte les éléments suivants :

- Un ensemble de neurones identiques (même type) sauf pour les valeurs de leurs poids ;
- Une limite imposée à la « force » d'un neurone ;
- Un mécanisme permettant aux neurones d'entrer en compétition pour le droit de répondre à un certain sous-ensemble des stimuli d'entrée, de manière à ce qu'un seul neurone de sortie soit actif à la fois.

2.2.4.4 Apprentissage non supervisé

L'apprentissage non supervisé est caractérisée par l'absence complète de professeur, c'est-à-dire n'existe pas une erreur comme dans le cas supervisé. Cet apprentissage est basé sur un ajustement des poids du réseau jusqu'à devenir stable.

2.3 Perceptron multicouches

Le perceptron multicouche est un réseau orienté de neurones artificiels organisé en couches et où l'information voyage dans un seul sens, de la couche d'entrée vers la couche de sortie.

La figure 15 donne l'exemple d'un réseau contenant une couche d'entrée, deux couches cachées et une couche de sortie. La couche d'entrée représente toujours une couche virtuelle associée aux entrées du système. Elle ne contient aucun neurone. Les couches suivantes sont des couches de neurones. Dans l'exemple illustré, il y a 3 entrées, 4 neurones sur la première couche cachée, trois neurones sur la deuxième et quatre neurones sur la couche de sortie. Les sorties des neurones de la dernière couche correspondent toujours aux sorties du système.

Dans le cas général, un perceptron multicouche peut posséder un nombre de couches quelconque et un nombre de neurones (ou d'entrées) par couche également quelconque.

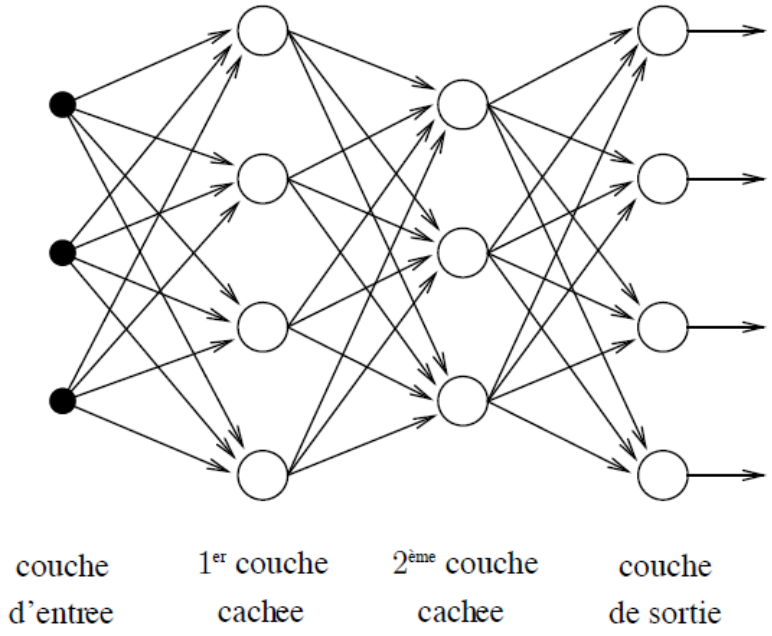

Figure 15 : Exemple de réseau de type perceptron multicouche

Les neurones sont reliés entre eux par des connexions pondérées. Ce sont les poids de ces connexions qui gouvernent le fonctionnement du réseau et « programment » une application de l'espace des entrées vers l'espace des sorties à l'aide d'une transformation non linéaire. La création d'un perceptron multicouche pour résoudre un problème donné passe donc par l'inférence de la meilleure application possible telle que définie par un ensemble de données d'apprentissage constituées de paires de vecteurs d'entrées et de sorties désirées.

Cette inférence peut se faire, entre autre, par l'algorithme dit de rétro propagation.

2.3.1 Apprentissage par rétropropagation du gradient

Soient $X = (x_1, x_2, ..., x_N)$ les entrées, $d = (d_1, d_2, ..., d_m)$ le vecteur des sorties désirées et $Y = (y_1, y_2, ..., y_m)$ vecteur des sorties calculées de notre réseau PMC. Où N est le nombre des entrées et m le nombre de sorties du réseau. L'algorithme de rétropropagation consiste à mesurer l'erreur entre les sorties désirées et les sorties calculées.

2.3.1.1 Cas de la couche de sortie

L'objectif de l'algorithme de rétropropagation procède à l'adaptation des poids des connexions du réseau de manière à minimiser la somme des erreurs sur tous les neurones de sortie. En commençant par la couche de sortie. Soit l'erreur calculée e_j, donnée par la relation (2.5) pour le neurone de sortie j.

$$e_j(t) = (d_j - y_j)(t) \tag{2.5}$$

Avec :

- $d_j(t)$: la sortie désirée du neurone j ;
- $y_j(t)$: la sortie calculée à l'instant t.

Soit E la somme des erreurs quadratiques observées sur l'ensemble C des neurones de sortie. Elle est donnée par :

$$E(t) = \frac{1}{2}\sum_{j \in C} e_j^{\,2}(t) \tag{2.6}$$

La sortie $y_j(t)$ est donnée par la relation (2.7) :

$$y_j(t) = f(\vartheta_j(t)) = f(\sum_{i=0}^{N} w_{j,i} y_i(t)) \tag{2.7}$$

Avec :

- f : la fonction d'activation du neurone ;
- $\vartheta_j(t)$: la somme pondérée des entrées du neurone j ;
- $w_{j,i}$: le poids de la connexion entre le neurone i de la couche précédente et le neurone j de la couche aval;
- $y_i(t)$: la sortie du neurone i.

On suppose que la couche précédente contient de 1 à N neurones, et que le poids $w_{j,0}$ correspond au biais du neurone et que l'entrée $y_0(t) = -1$.

Pour corriger l'erreur observée, il s'agit de modifier les poids $w_{j,i}$ dans le sens opposé au gradient $\frac{\partial E(t)}{\partial w_{j,i}}$ de l'erreur.

Par la règle des dérivées partielles, on obtient :

$$\frac{\partial E(t)}{\partial w_{j,i}} = \frac{\partial E(t)}{\partial e_j(t)} \cdot \frac{\partial e_j(t)}{\partial y_j(t)} \cdot \frac{\partial y_j(t)}{\partial \vartheta_j(t)} \cdot \frac{\partial \vartheta_j(t)}{\partial w_{j,i}} \tag{2.8}$$

Et on exprime la variation de poids $\Delta w_{j,i}$ sous la forme :

$$\Delta w_{j,i} = -\eta \frac{\partial E(t)}{\partial w_{j,i}} \tag{2.9}$$

Avec :

- $0 \leq \eta \leq 1$: le taux d'apprentissage.

Donc après le calcule de chaque dérivée partielle nous allons obtenir :

$$\frac{\partial E(t)}{\partial w_{j,i}} = e_j(t) y_j(t) \left[1 - y_j(t)\right] y_i(t) \tag{2.10}$$

Et la règle dite du "delta" pour la couche de sortie s'exprime par :

$$\Delta w_{j,i} = -\eta \frac{\partial E(t)}{\partial w_{j,i}} = \eta \delta_j(t) y_i(t) \tag{2.11}$$

Avec :

$$\delta_j(t) = -e_j(t) y_j(t) \left[1 - y_j(t)\right] \tag{2.12}$$

2.3.1.2 Cas d'une couche cachée

Considérons maintenant le cas d'une couche cachée, de même l'objectif sera toujours d'adapter les poids de la couche courante en minimisant la somme des erreurs sur les neurones de la couche de sortie. Les indices i et j désigneront respectivement un neurone sur la couche précédente et un neurone sur la couche courante, et k servira maintenant à désigner un neurone sur la couche suivante.

Reprenons l'expression (2.12) de la dérivée partielle de l'erreur totale $E(t)$ par rapport à ω_{ji} mais ne dérivant plus par rapport à l'erreur $e_j(t)$ car celle-ci est maintenant inconnue :

$$\frac{\partial E(t)}{\partial w_{j,i}} = \frac{\partial E(t)}{\partial y_j(t)} \cdot \frac{\partial y_j(t)}{\partial \vartheta_j(t)} \cdot \frac{\partial \vartheta_j(t)}{\partial w_{j,i}} \tag{2.12}$$

De même après le calcul de chaque dérivée partielle nous allons obtenir :

$$\frac{\partial E(t)}{\partial w_{j,i}} = -y_j(t)\left[1 - y_j(t)\right]\left[\sum_{k \in C} \delta_k(t)\omega_{kj}\right] y_i(t) \tag{2.13}$$

Et :

$$\Delta w_{j,i} = -\eta \frac{\partial E(t)}{\partial w_{j,i}} = \eta \delta_j(t) y_i(t) \tag{2.14}$$

Avec :

$$\delta_j(t) = -y_j(t)\left[1 - y_j(t)\right]\left[\sum_{k \in C} \delta_k(t)\omega_{kj}\right] \tag{2.15}$$

- **Algorithme de rétro-propagation**

L'algorithme de rétro-propagation standard se résume donc à la série d'étapes suivantes :

Algorithme de rétropropagation

1. Initialiser tous les poids à de petites valeurs aléatoires dans l'intervalle $[-0.5, 0.5]$.
2. Normaliser les données d'entraînement.
3. Permuter aléatoirement les données d'entraînement.
4. **Pour chaque donnée d'entraînement n** :

   Calculer les sorties observées en propageant les entrées vers l'avant.

   Ajuster les poids en rétro-propageant l'erreur observée.

$$\omega_{ji}(t) = \omega_{ji}(t-1) + \Delta\omega_{ji}(t) = \omega_{ji}(t-1) + \eta\delta_j(t)y_i(t) \qquad (2.16)$$

Où :

$$\delta_j(t) = \begin{cases} e_j(t)y_j(t)\left[1-y_j(t)\right] & \text{si j} \in \text{couche de sortie} \\ y_j(t)\left[1-y_j(t)\right]\left[\sum_{k\in C}\delta_k(t)\omega_{kj}\right] & \text{si j} \in \text{couche cachée} \end{cases} \qquad (2.17)$$

Avec :

- $0 \le \eta \le 1$: le taux d'apprentissage.

5. Répéter les étapes 3 et 4 jusqu'à un nombre maximum d'itérations ou jusqu'à ce que la racine de l'erreur quadratique moyenne soit inférieure à un certain seuil.

2.3.2 Implémentation de l'algorithme de rétropropagation

Dans le cadre de l'application de l'algorithme de rétropropagation, nous avons réalisé un programme en langage C qui implémente cet algorithme.

Dans ce qui suit, nous allons traiter un exemple de réseau de neurones multicouche tel que : Le vecteur $X_{or} = [x_1 = 1, x_2 = 0]$ représente l'entrée, et $d = [d_1 = 1]$ la sortie désirée.

Pour cela, nous allons utiliser un réseau multicouche de 2 couches : la première couche est composée de trois neurones, et la seconde d'un seul neurone (Figure 16).

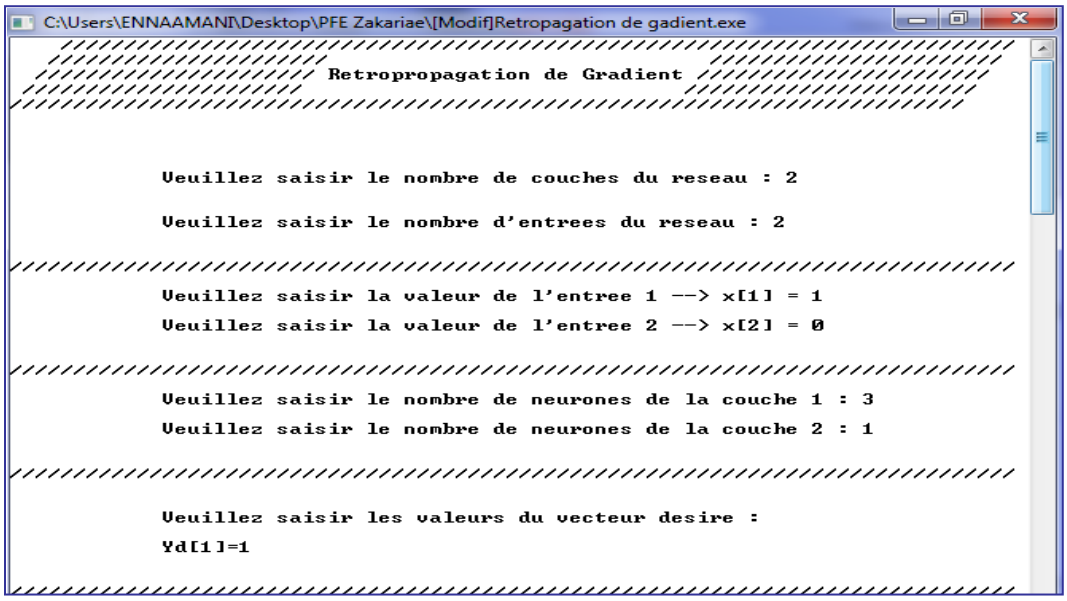

Figure 16 : Perceptron multicouche utilisé dans l'application

Après 500 itérations (Figure 17) nous constatons que l'erreur quadratique diminue à chaque itération et se fixe à 0.001203. Nous observons également que la sortie calculée est égale à 0.990836 proche de la sortie désirée 1. Notons que dans la première itération l'erreur quadratique est égale à 0.085413 et la sortie calculée vaut 0.767606.

Après 1000 itérations (Figure 18) nous constatons que l'erreur quadratique diminue à chaque itération et se fixe à 0. 000570. Nous observons également que la sortie calculée est égale à 0.993772 proche de la sortie désirée 1. Notons que dans la première itération l'erreur quadratique est égale à 0.085413 et la sortie calculée vaut 0.767606.

Après 10000 itérations (Figure 19) nous constatons que l'erreur quadratique diminue à chaque itération et se fixe à 0. 000052. Nous observons également que la sortie calculée est égale à 0.998154 proche de la sortie désirée 1. Notons que dans la première itération l'erreur quadratique est égale à 0.085413 et la sortie calculée vaut 0.767606.

Figure 17 : Résultats du modèle après 500 itérations

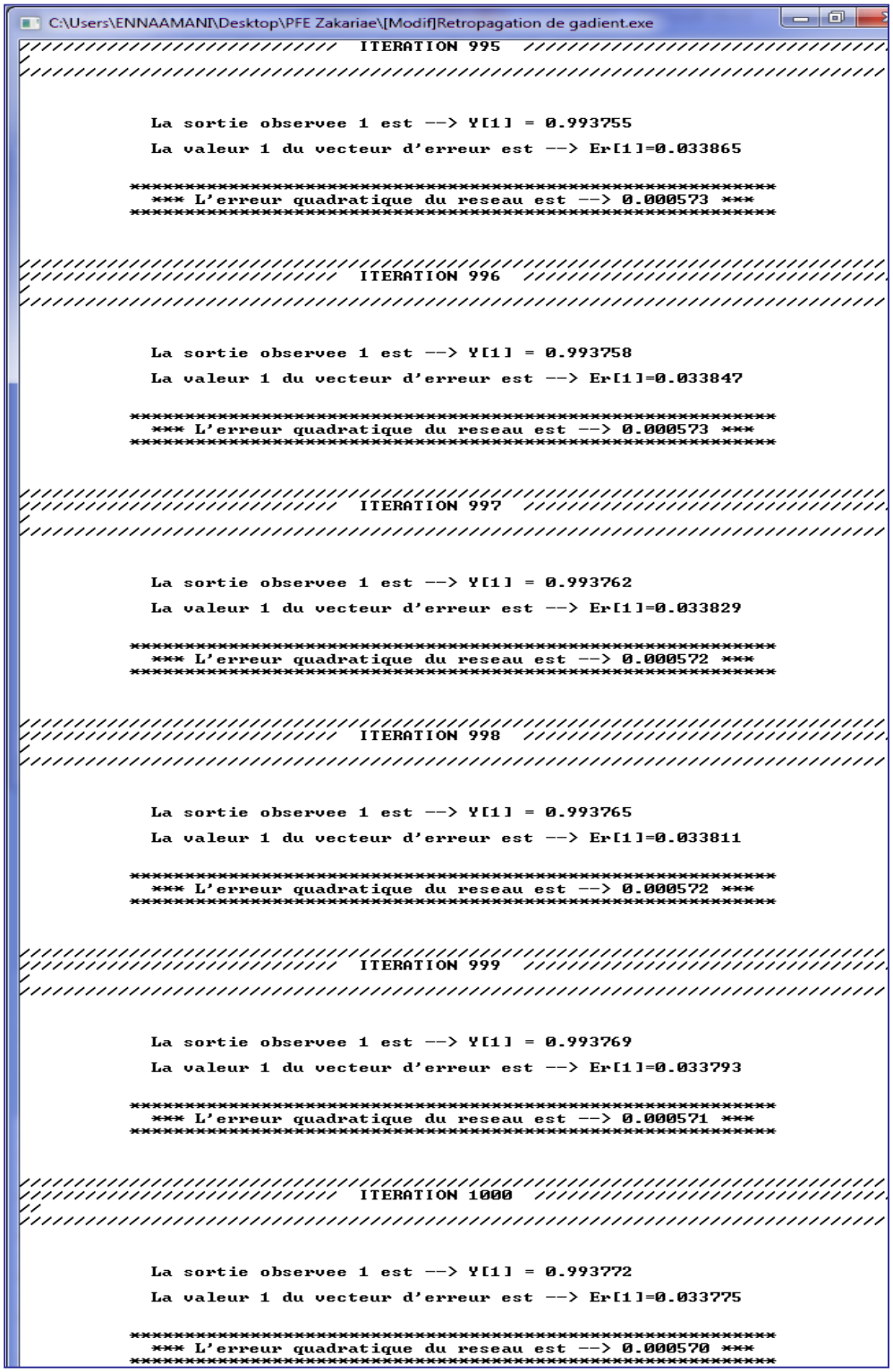

Figure 18 : Résultats du modèle après 1000 itérations

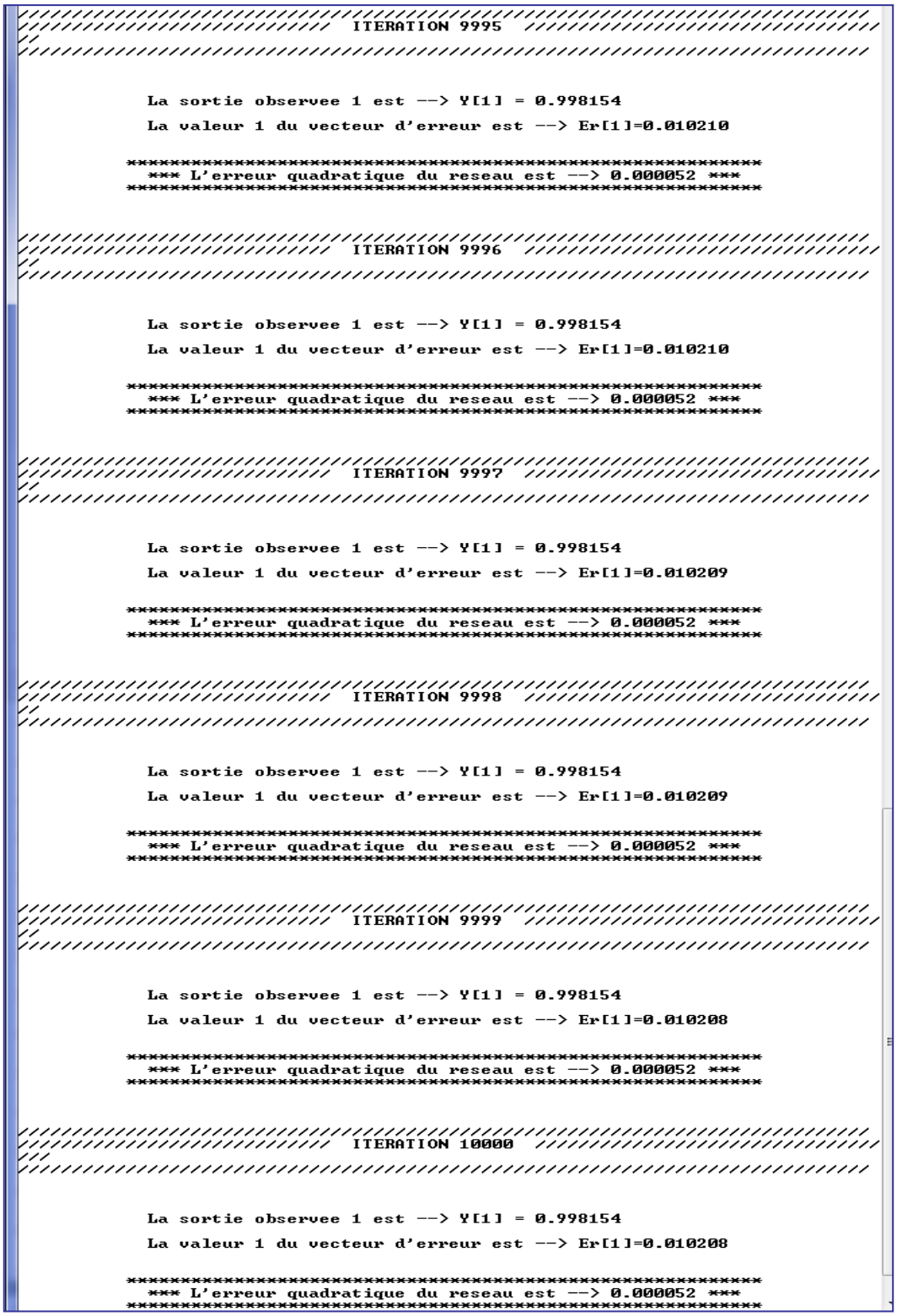

Figure 19 : Résultats du modèle après 10000 itérations

2.4 Conclusion

Nous avons présenté dans cette partie les concepts généraux d'un réseau de neurones, et en particulier le modèle perceptron multicouches. Nous avons également présenté un programme réalisé en langage C de PMC et de son apprentissage par rétropropagation de gradient qui est justement appliqué dans la reconnaissance automatique de la parole.

Chapitre 3

Modèle de Markov Caché (MMC)

3.1 Introduction

A l'heure actuelle, il n'existe pas encore d'applications commerciales de méthode efficace de description du signal vocal sur laquelle on peut baser la reconnaissance. La technique qui s'est avérée la meilleure pour le moment est la recherche d'une similitude entre des chaines de vecteurs mémorisées représentant par exemple des mots et celle qu'on déduit du signal enregistré. Les modèles de Markov cachés sont une approche prometteuse dans différents domaines d'applications où on envisage de traiter des données quantifiées qui peuvent être partiellement erronées comme par exemple la reconnaissance d'images ou la reconnaissance de la parole.

Dans ce chapitre, nous allons d'abord présenter les modèles de Markov dans le cas général ; ensuite, nous allons voir les deux modèles de Markov Observable et Caché avec les algorithmes qui leur sont liés, pour enfin présenter un programme développé avec le langage C qui implémente le Modèle de Markov Caché.

3.2 Terminologie et définitions

Dans cette partie, nous allons présenter un nombre de définitions et théorèmes relatifs aux chaînes de Markov.

Définition 1 :

Un processus stochastique (ou aléatoire) est une famille de variables aléatoires $X_t : \{X_t, t \in T\}$, où t représente le plus souvent, un parcours de temps. Si T est discret, on parle de suite stochastique, sinon on parle d'un processus dans le cas continu.

Définition 2 : (Chaînes de Markov à espace d'états discret)

Soient $T = \{t_1, t_2, ..., t_n, ...\}$ et $Q = \{q_1, q_2, ..., q_n, ...\}$ un ensemble d'états fini ou infini.

Une chaîne de Markov est un processus aléatoire $\{X_t, t \in T\}$ tel que si pour tout instant u, pour toute valeur $X_u = i$ donnée (on dit que le processus est passé par l'état q_u à l'instant

u). La probabilité pour que le processus prenne la valeur y un instant quelconque t avec $(t > u)$ qui ne dépend pas des valeurs prises par le processus avant l'instant u est donnée par la relation (3.1).

$$P(X_t = j \mid X_s, \forall s \prec u, X_u = i) = P(X_t = j \mid X_u = i) \qquad (3.1)$$

On dit que le processus est sans mémoire.

Définition 3 : (Chaîne de Markov homogène)

Une chaîne est dite homogène si, pour tout intervalle $[u,t]$, la probabilité $P(X_t = j \mid X_u = i)$ dépend seulement de la longueur $t-u$ de l'intervalle.

Alors,

$$\forall n, P(X_{t+n} = j \mid X_{u+n} = i) = P(X_t = j \mid X_u = i) \qquad (3.2)$$

En particulier, cette probabilité est égale à :

$$P(X_{t-u} = j \mid X_0 = i) \qquad (3.3)$$

3.3 Modèle de Markov

Nous allons voir dans cette partie un outil puissant pour induire un concept de nature statistique à partir seulement de séquences d'apprentissage appartenant à ce concept : les modèles de Markov. Il existe deux modèles de Markov : le modèle de Markov observable (MMO) et le modèle de Markov caché (MMC).

Généralement, un modèle de Markov (Figure 20) est vu comme un ensemble discret d'états et de transitions reliant ces états entre eux. Formellement, il peut être défini par l'ensemble des paramètres $\lambda = (Q, O, P, B, \prod)$.

Où :

- Q : ensemble d'états fini du modèle de cardinal N ;

- O : ensemble d'observations fini du modèle de cardinal M ;
- $P = \left(p_{i,j} \right)_{\substack{1 \le i \le N \\ 1 \le j \le N}}$: matrice des probabilités de transition sur l'ensemble des états du modèle. La probabilité de transition est la probabilité de choisir la transition $p_{i,j}$ pour accéder à l'état q_j, étant donné un processus à l'état q_i ;
- $B = \left(b_j(o_t) \right)_{\substack{1 \le j \le N \\ 1 \le t \le |T|}}$: matrice des probabilités d'émission de l'observation o_t dans l'état q_j ;
- $\prod = (\pi_1, \pi_2, ..., \pi_N)$: distribution initiale des états $\pi_j = P(q_0 = j), \forall j \in [1, N]$, où q_0 représente l'état initial du modèle.

Le but de ce modèle est de trouver la probabilité d'obtenir une certaine séquence d'observations.

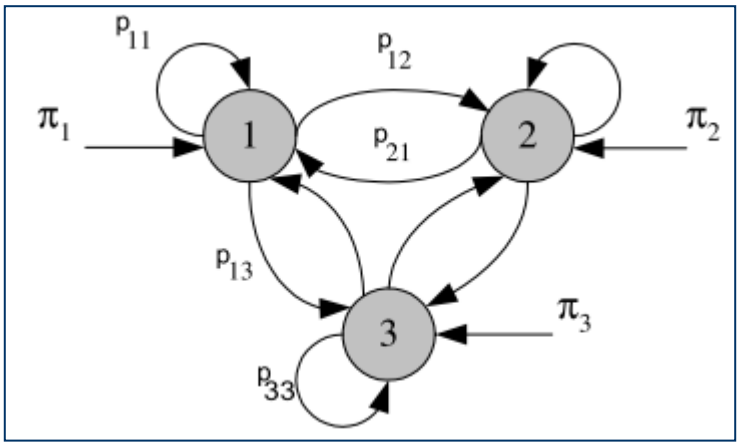

Figure 20 : Modèle de Markov à 3 états

3.3.1 Modèle de Markov Observale (MMO)

3.3.1.1 Définition

On dit qu'un modèle de Markov est observable si l'espace d'états est le même que l'espace d'observations c'est-à-dire $O = Q$.

Donc on peut l'écrire sous la forme :

$$\lambda_o = (Q, P, \prod) \tag{3.10}$$

De plus, à chaque pas de temps t on connaît l'état q_t. Et soit une séquence d'observations O qui correspond à la séquence d'états de l'ensemble Q.

La probabilité d'obtenir une certaine séquence d'observations étant donné ce modèle est :

$$P(O = Q \mid \lambda_o) = P(q_1) * \prod_{t=2}^{T} P(q_t \mid q_{t-1}) = \pi_{q_1} p_{q_1 q_2} p_{q_2 q_3} ..... p_{q_{T-1} q_T} \tag{3.11}$$

3.3.1.2 Exemple

Considérons un système météorologique (Figure 21).

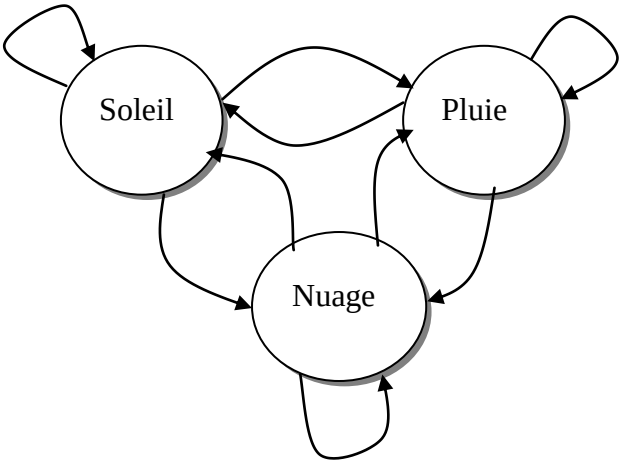

Figure 21 : Système météorologique

Avec :

- $N = 3$: le nombre d'états.
- $Q = O = \{q_1, q_2, q_3\}$ avec : $q_1 =$ pluvieux ; $q_2 =$ nuageux ; $q_3 =$ ensoleillé.
- $\Delta t =$ une journée.

Cet exemple est proposé par Météo France, et grâce à des mesures passées on pourrait calculer la matrice de transition entre les états $P = (p_{ij})$.

Aujourd'hui $(t = 1)$, il fait beau $P(q_1 = 3) = 1$. Donc la question résolue par ce genre de modèles est de calculer la probabilité que le temps des 15 prochains jours soit :

Soleil-soleil-pluie-pluie-nuage-nuage-nuage-soleil-soleil-soleil-soleil-soleil-pluie-pluie-pluie.

Solution

Il est facile de trouver la solution à la main. On va chercher d'abord les paramètres de modèle de Markov observable.

$$P = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

Et :

$$\prod = \begin{bmatrix} 0.3 & 0.5 & 0.2 \end{bmatrix}$$

Soit une séquence d'observations :

$O =$ s-s-p-p-n-n-n-s-s-s-s-s-p-p-p.

Alors :

$$P(O) = \pi_s . p_{ss} . p_{sp} . p_{pp} . p_{pn} . p_{nn} . p_{nn} . p_{ns} . p_{ss} . p_{ss} . p_{ss} . p_{ss} .. p_{sp} . p_{pp} . p_{pp}$$

D'où :

$$P(O) = 0.000000906$$

3.3.2 Modèle de Markov Caché (MMC)

3.3.2.1 Définition

On dit qu'un modèle de Markov est caché (MMC) si l'espace d'états est différent de l'espace d'observations c'est-à-dire $O \neq Q$.

Donc on peut l'écrire sous la forme :

$$\lambda_c = (Q, O, P, B, \prod) \tag{3.12}$$

3.3.2.2 Problèmes fondamentaux des MMC

Soient λ_c un modèle de Markov caché et O une séquence d'observations. L'objectif est de trouver un modèle λ qui maximise la probabilité $P(\lambda | O)$ (probabilité qu'un modèle λ génère une séquence O). Malheureusement, il n'est pas possible d'accéder directement à cette probabilité. Mais on peut calculer la probabilité qu'un modèle donné génère une certaine séquence de vecteurs acoustiques $P(O | \lambda)$.

En utilisant la loi de Bayes, il est possible de lier ces deux probabilités par :

$$P(\lambda | O) = \frac{P(O | \lambda).P(\lambda)}{P(O)} \tag{3.13}$$

Avec :

- $P(O | \lambda)$: probabilité de la séquence d'observations O sachant le modèle λ_c ;
- $P(\lambda)$: probabilité a priori du modèle ;
- $P(O)$: probabilité a priori de la séquence des vecteurs d'observations.

Puisque $P(O)$ est considérée constante, le problème de maximiser $P(\lambda | O)$ revient à maximiser $P(O | \lambda).P(\lambda)$. Pour cela, il faut résoudre les trois problèmes fondamentaux des MMC : problème d'évaluation, problème de décodage et problème d'apprentissage.

3.3.2.3 Problème d'évaluation

On peut se donner une séquence d'observations O et le modèle λ_c, la question à poser est comment calculer efficacement $P(O | \lambda)$?

Pour répondre à cette question on va passer par plusieurs étapes.

Premièrement, on va calculer la probabilité d'observer la séquence O pour une séquence d'états Q qui est égale à :

$$P(O|Q,\lambda_c) = b_{q_1}(o_1).b_{q_2}(o_2)...b_{q_T}(o_T) \quad (3.14)$$

Or, la probabilité de la séquence Q s'écrit sous la forme :

$$P(Q|\lambda) = \pi_{q_1} p_{q_1 q_2} p_{q_2 q_3} ... p_{q_{T-1} q_T} \quad (3.15)$$

Donc la probabilité conjointe du chemin Q et des observations O est :

$$P(O,Q|\lambda) = P(Q|\lambda).P(O|Q,\lambda) \quad (3.16)$$

Alors :

$$P(O|\lambda) = \sum_Q P(O,Q|\lambda) \quad (3.17)$$

C'est-à-dire :

$$P(O|\lambda) = \sum_{q_1,...,q_T} \pi_{q_1} b_{q_1}(o_1) p_{q_1 q_2} b_{q_2}(o_2) ... p_{q_{T-1} q_T} b_{q_T}(o_T) \quad (3.18)$$

Pour une machine à N états, ce calcul direct nécessite $(2T-1)*N^T$ multiplications et $N^T - 1$ additions, ce qui le rend trop complexe et impossible à implémenter. Il existe heureusement un algorithme rapide et efficace dit avant-arrière qui donne une solution pour mener efficacement ce calcul.

- **Algorithme avant-arrière**

Soit, la probabilité avant (3.19), d'observer la séquence $o_1, o_2, ..., o_t$ et d'être à l'état i à l'instant t sachant le modèle λ. Cette probabilité est calculée d'une manière récursive.

$$\alpha_t(i) = P(o_1, o_2, ..., o_t, q_t = i|\lambda) \quad (3.19)$$

Algorithme de calcul de la fonction avant (Forward) : α

Pour $i = 1,...,N$

$\alpha_1(i) \leftarrow \pi_i b_i(o_1)$

Fin pour

$t \leftarrow 1$

Tant que $t \prec T$ **faire**

$j \leftarrow 1$

Tant que $j \leq N$ **faire**

$$\alpha_{t+1}(j) \leftarrow b_j(o_{t+1}) \left[\sum_{i=1}^{N} \alpha_t(i).p_{ij} \right]$$

$j \leftarrow j + 1$

Fin Tant que

$t \leftarrow t + 1$

Fin Tant que

$$P(O \mid \lambda) \leftarrow \sum_{i=1}^{N} \alpha_T(i)$$

Cette récursion dépend du fait que la probabilité d'être à l'état j au temps $t+1$ et d'observer o_{t+1} peut être déduite en sommant les probabilités avant pour tous les états prédécesseurs de j pondérées par les probabilités de transition p_{ij} .

L'algorithme avant étant présenté, la partie qui suit va traiter l'algorithme arrière.

La probabilité arrière $\beta_t(j)$ est définie par :

$$\beta_t(j) = P(o_{t+1}, o_{t+2}, ..., o_T, q_t = j \mid \lambda) \qquad (3.20)$$

C'est la probabilité d'observer la séquence $o_{t+1}, o_{t+2}, ..., o_T$ sachant qu'on est à l'état i au temps t et qu'on a le modèle λ . Cette probabilité est aussi calculée d'une manière récursive.

Algorithme de calcul de la fonction arrière (Backward) : β

Pour $i = 1,...,N$

$\beta_T(i) \leftarrow 1$

Fin pour

$t \leftarrow T$

Tant que $t \succ 1$ **faire**

$j \leftarrow 1$

Tant que $j \leq N$ **faire**

$$\beta_t(i) \leftarrow \sum_{j=1}^{N} p_{ij}.b_j(o_{t+1}).\beta_{t+1}(j)$$

$$j \leftarrow j - 1$$

Fin Tant que

$$t \leftarrow t - 1$$

Fin Tant que

$$P(O \mid \lambda) \leftarrow \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$$

3.3.2.4 Problème de décodage

Soient O une séquence d'observations et un modèle λ. Le problème de décodage revient à la recherche d'une séquence d'états optimale qui va générer O. C'est à dire maximiser $P(Q \mid O, \lambda)$, ce qui revient à maximiser $P(Q, O \mid \lambda)$.

Pour trouver ce résultat, appliquons l'algorithme de Viterbi.

Alors pour trouver $Q = \{q_1, q_2, ..., q_T\}$ pour une séquence d'observations $O = \{o_1, o_2, ..., o_T\}$, on définit la variable intermédiaire $\delta_t(i)$ (3.21) qui est la probabilité du meilleur chemin amenant à l'état i à l'instant t, en étant guidé par les t premières observations :

$$\delta_t(i) = \underset{q_1,...,q_{t-1}}{Max}\ P(q_1, q_2, ..., q_t = i, o_1, o_2, ..., o_t \mid \lambda) \qquad (3.21)$$

Par récurrence, on calcule :

$$\delta_{t+1}(j) = \left[\underset{i}{Max}\, \delta_t(i)\, p_{ij} \right] b_j(o_{t+1}) \qquad (3.22)$$

En gardant trace, lors du calcul, de la suite d'états qui donne le meilleur chemin amenant à l'état i à t dans un tableau ψ .

On a donc l'algorithme qui fournit en sortie la valeur P^* de la probabilité de l'émission de la séquence par la meilleure suite d'états $\left\{q_1^*, q_2^*, ..., q_T^*\right\}$.

La fonction *Argmax* permet de mémoriser l'indice i entre 1 et N, avec lequel on atteint le maximum des quantités $\delta_t(i)\, p_{ij}$.

Algorithme de Viterbi

Pour $i = 1, ..., N$

$\delta_1(i) \leftarrow \pi_i b_i(O_i)$
$\psi_1(i) \leftarrow 0$

Fin pour

$t \leftarrow 2$

Tant que $t \le T$ **faire**

$j \leftarrow 1$

Tant que $j \le N$ **faire**

$\delta_t(j) \leftarrow Max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij} \right] b_j(O_t)$

$\psi_t(j) \leftarrow ArgMax_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij} \right]$

$j \leftarrow j + 1$

Fin Tant que

$t \leftarrow t + 1$

Fin Tant que

$P^* \leftarrow Max_{1 \le i \le n} \left[\delta_T(i) \right]$

$q_T^* \leftarrow ArgMax_{1 \le i \le N} \left[\delta_T(i) \right]$

$t \leftarrow T$

Tant que $t \geq 1$ **faire**

$q_t^* \leftarrow \psi_{t+1}(q_{t+1}^*)$

$t \leftarrow t-1$

Fin tant que

3.3.2.5 Problème d'apprentissage

Cette phase est comme toutes les méthodes d'apprentissage. Elle consiste à trouver une méthode pour ajuster les paramètres du modèle $\lambda = (P, B, \prod)$ jusqu'à trouver un modèle $\lambda^* = (P^*, B^*, \prod^*)$ qui permet de générer notre séquence d'observations O d'une vraisemblance maximale. Ce problème n'a pas de solution analytique connue et il n'existe pas de technique optimale pour estimer les paramètres du modèle. On peut choisir λ telle que $P(O \mid \lambda)$ soit localement maximale en utilisant une procédure itérative telle que la méthode de Baum-Welch ou la technique du gradient. Dans la suite de notre mémoire nous allons utiliser la technique de Baum-Welch.

Pour cela nous allons définir la probabilité $\xi_t(i, j)$ qui représente la probabilité d'être à l'état i au temps t et de faire une transition à l'état j au temps $t+1$ sachant O et λ.

$$\delta_t(i) = \underset{q_1,\ldots,q_{t-1}}{Max}\ P(q_1, q_2, \ldots, q_t = i, o_1, o_2, \ldots, o_t \mid \lambda) \qquad (3.23)$$

Et :

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j \mid O, \lambda) \qquad (3.24)$$

D'après les définitions des probabilités avant et arrière on a :

$$\xi_t(i,j) = \frac{\alpha_t(i).p_{ij}.b_j(o_{t+1}).\beta_{t+1}(j)}{P(O|\lambda)} \tag{3.25}$$

De plus, on sait $\gamma_t(i)$, donc il existe une relation entre $\xi_t(i,j)$ et $\gamma_t(i)$ tel que :

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{3.26}$$

Et $t \in \{1,2,...,T-1\}$

L'algorithme de Baum-Welch estime les nouveaux paramètres du MMC comme suit :

$$\overline{\pi}_i = \gamma_1(i) \qquad 1 \le i \le N \tag{3.27}$$

Où $\overline{\pi}_i$ est la ré-estimation de π_i.

$$\overline{p}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(j)} \qquad 1 \le i \le N \qquad 1 \le j \le N \tag{3.28}$$

Où $\overline{p}_{ij}$ la ré-estimation de p_{ij} est le rapport du nombre de transitions de l'état i vers j sur le nombre de transitions partant de l'état i.

$$\overline{b}_j(k) = \frac{\sum_{t=1, o_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \qquad 1 \le j \le N \tag{3.29}$$

Où $\overline{b}_j(k)$ la ré-estimation de $b_j(k)$ est le rapport du nombre de fois d'être à l'état i en observant sur le nombre de fois étant dans l'état i.

Algorithme de Baum-Welch

Fixer les valeurs initiales $\lambda_0 = (P, B, \prod)$

Définir le MMC de départ avec $\lambda_0 = (P, B, \prod)$.

$p \leftarrow 0$

Tant que la convergence n'est pas réalisée **faire**

On procède le MMC λ_p.

Calculer pour ce modèle, sur l'ensemble d'apprentissage, les valeurs : $\xi_t(i, j)$ et $\gamma_t(j)$ avec $1 \le i, j \le N \;\; 1 \le t \le T-1$.

En déduire : $\overline{\prod}$, $\overline{B}$ et $\overline{P}$.

Le MMC courant défini par $\lambda_p = (\overline{P}, \overline{B}, \overline{\prod})$.

$p \leftarrow p+1$

Fin tant que

Nous avons obtenu un nouveau modèle ré-estimé $\overline{\lambda} = (\overline{P}, \overline{B}, \overline{\prod})$ qui nous permet d'affirmer l'une ou l'autre de ces propositions :

- Le modèle initial λ définit un point critique de la fonction de vraisemblance, dans ce cas $\overline{\lambda} = \lambda$.
- On a $P(O \mid \overline{\lambda}) \succ P(O \mid \lambda)$, donc le modèle $\overline{\lambda}$ est meilleur que le modèle λ.

3.3.2.6 Implémentation des algorithmes du Modèle de Markov Caché

On va considérer l'exemple de modèle de Markov observable (Figure 22).

Les états et les transitions sont les mêmes qui l'exemple traité dans le chapitre 2, mais on ajoute de plus :

$O = \{o_1, o_2, o_3, o_4\}$ avec : $o_1 =$ café ; $o_2 =$ sport ; $o_3 =$ étude ; $o_4 =$ sommeil.

Et :

$$\Pi = (\pi_1, \pi_2, \pi_3)$$

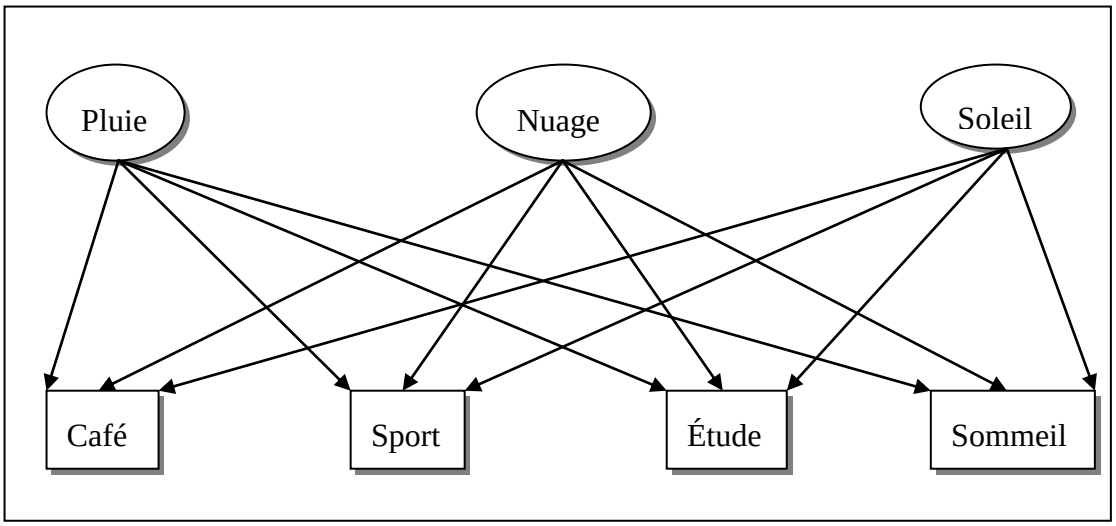

Figure 22 : Exemple d'application du MMO

Maintenant le problème à résoudre est de trouver le meilleur modèle qui va générer la séquence d'observations :

Café-sport-sommeil-étude-étude-étude-sommeil-sommeil-sport-café

Solution

Il est difficile de trouver la solution à la main. Pour cela, nous avons développé un programme en langage C qui effectue tous les calculs nécessaires.

Trouvons d'abord les paramètres de modèle MMC initial λ_0 :

$$P = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.20 & 0.20 & 0.20 \\ 0.10 & 0.20 & 0.10 \\ 0.40 & 0.40 & 0.40 \\ 0.30 & 0.20 & 0.30 \end{bmatrix}$$

Et :

$$\prod = \begin{bmatrix} 0.3 & 0.5 & 0.2 \end{bmatrix}$$

Soit $O = \begin{bmatrix} 1 & 0 & 2 & 3 \end{bmatrix}$ une séquence d'observations, alors :

$$P(O \mid \lambda_0) = 0.003199$$

Si on prend comme ensemble d'apprentissage cette seule séquence, l'application de l'algorithme Baum-Welch doit augmenter sa probabilité de reconnaissance.

Et pour cela nous allons itérer le code jusqu'à la convergence, alors le dernier modèle obtenu à la convergeance est le meilleur modèle qui génère cette séquence d'observation.

Donc on va trouver les resultats suivantes :

Après une réestimation,

On trouve le MMC λ_1 (Figure 23) :

$$\prod = \begin{bmatrix} 0.2014 & 0.6591 & 0.1393 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.4158 & 0.2587 & 0.3253 \\ 0.2163 & 0.5568 & 0.2267 \\ 0.0994 & 0.0781 & 0.8224 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.2595 & 0.2660 & 0.2247 \\ 0.2302 & 0.3858 & 0.0983 \\ 0.2493 & 0.2013 & 0.3091 \\ 0.2608 & 0.1467 & 0.3677 \end{bmatrix}$$

$$P(O \mid \lambda_0) = 0.005857$$

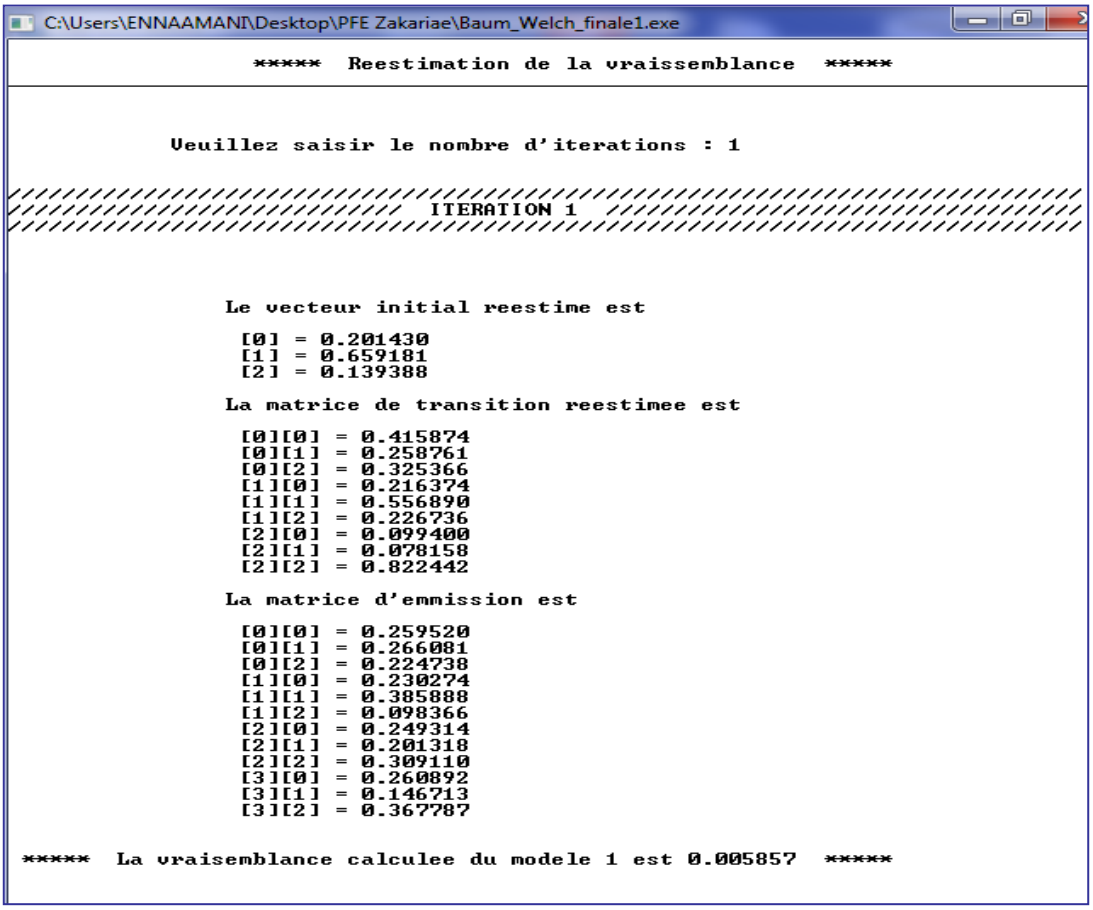

Figure 23 : Résultat du modèle MMC après 1 itération

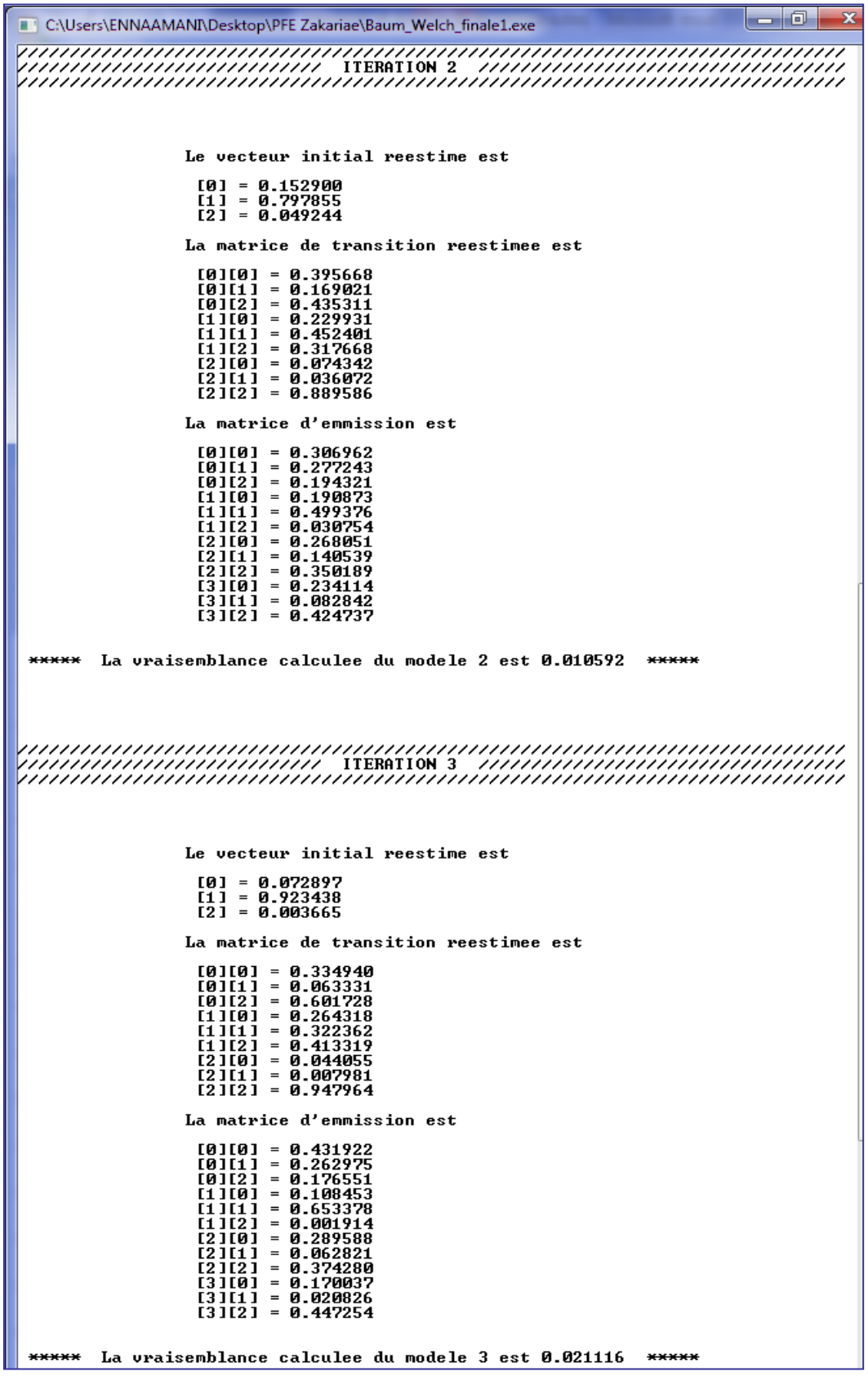

Figure 24 : Résultat du modèle MMC après 3 itérations

Après quatre itérations on a,

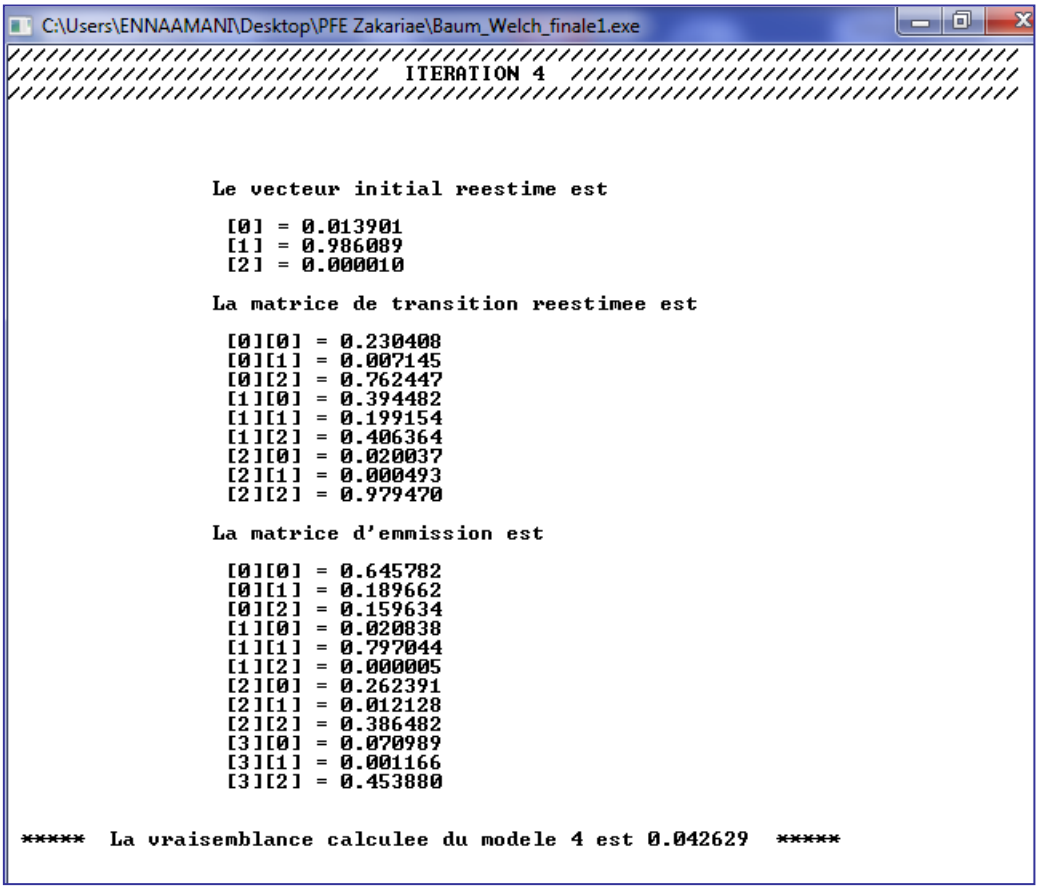

Figure 25 : Résultat du modèle MMC après 4 itérations

On trouve alors le MMC λ_4 :

$$\prod = \begin{bmatrix} 0.0139 & 0.9860 & 0.0000 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.2304 & 0.0071 & 0.7624 \\ 0.3944 & 0.1991 & 0.4063 \\ 0.0200 & 0.0004 & 0.9794 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.6457 & 0.1896 & 0.1596 \\ 0.0200 & 0.7970 & 0.0000 \\ 0.2623 & 0.0121 & 0.3864 \\ 0.0709 & 0.0011 & 0.4538 \end{bmatrix}$$

$$P(O \mid \lambda_4) = 0.042629$$

Après dix itérations on a,

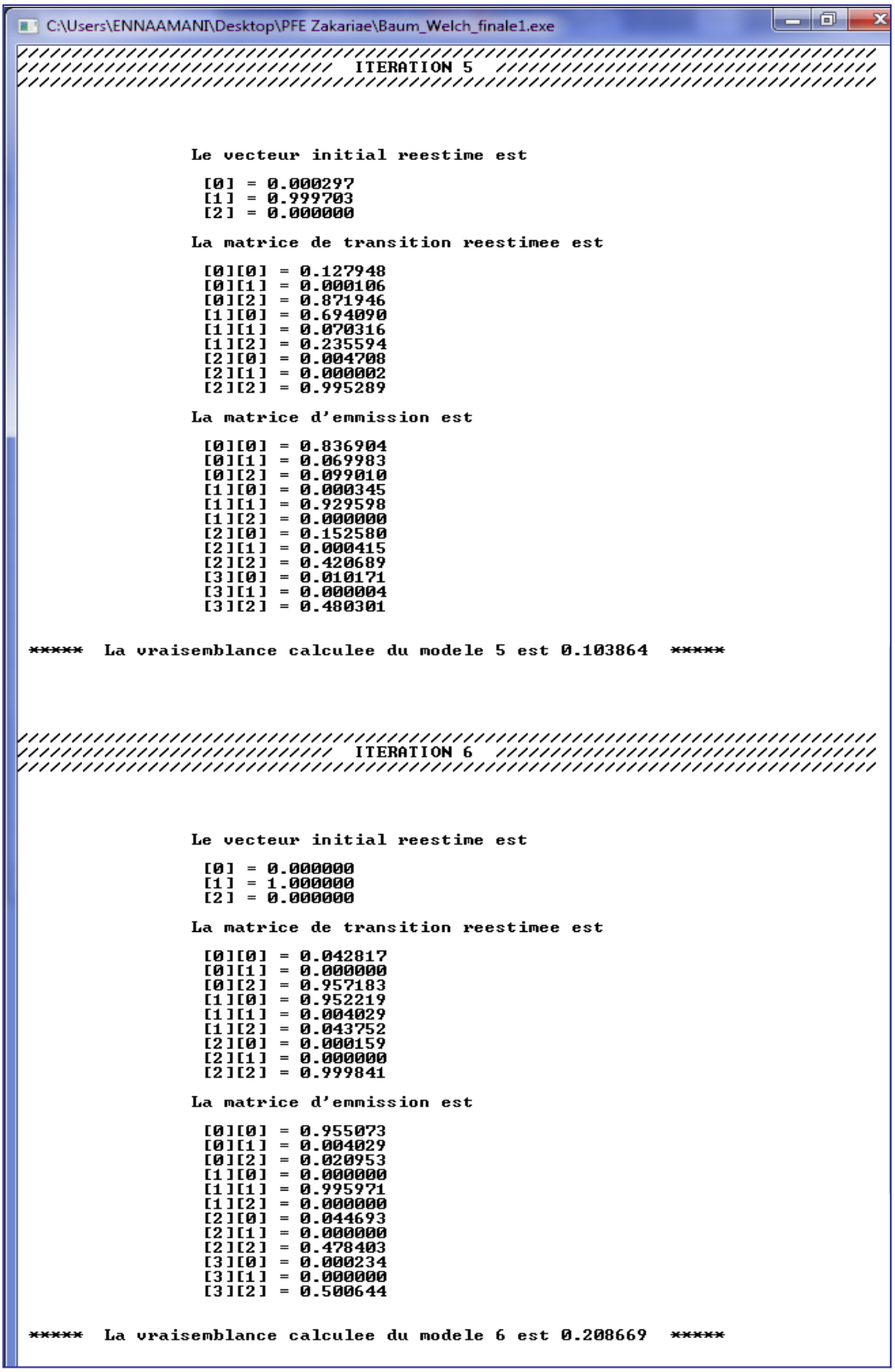

Figure 26 : Résultat du modèle MMC après 6 itérations

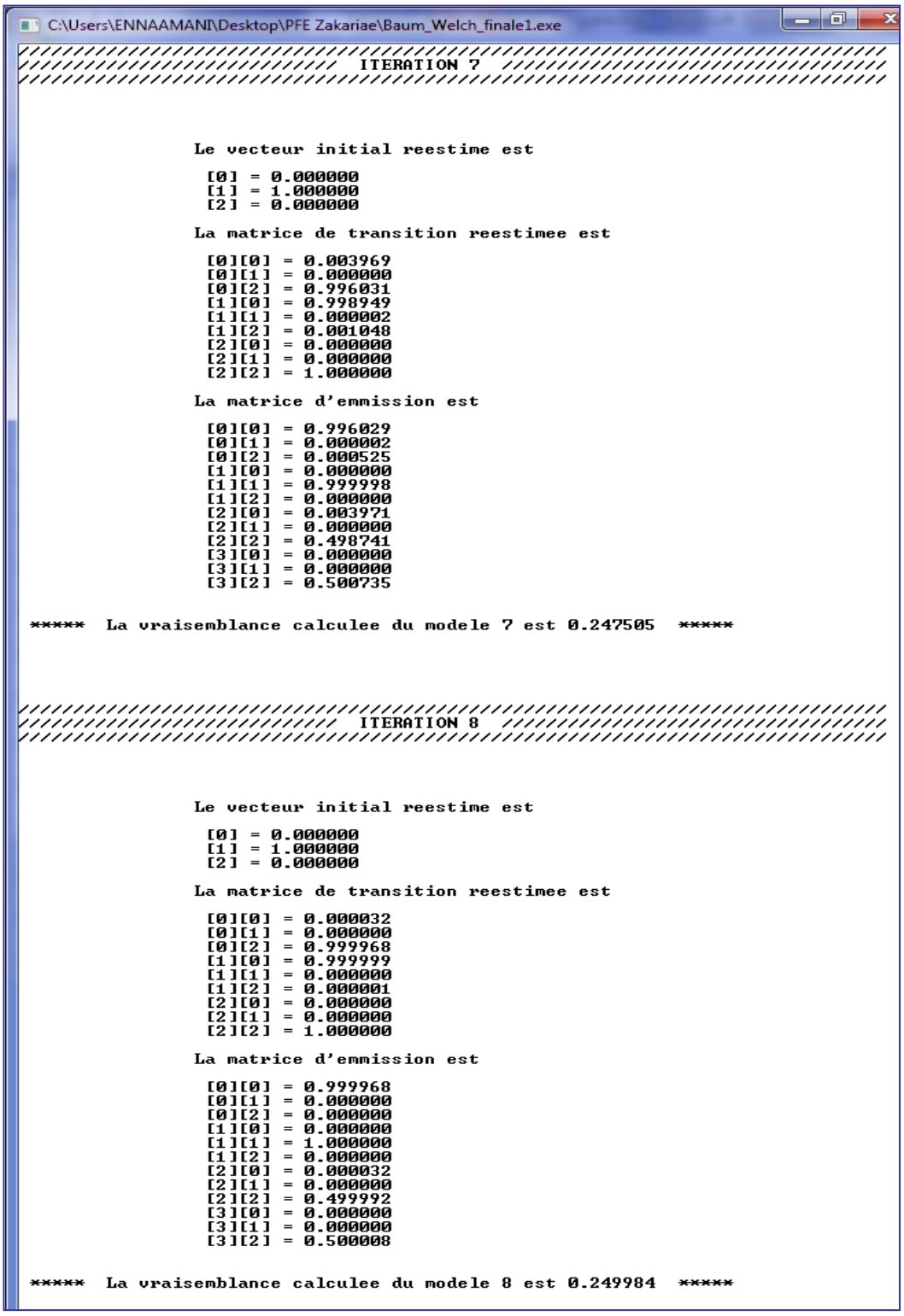

Figure 27 : Résultat du modèle MMC après 8 itérations

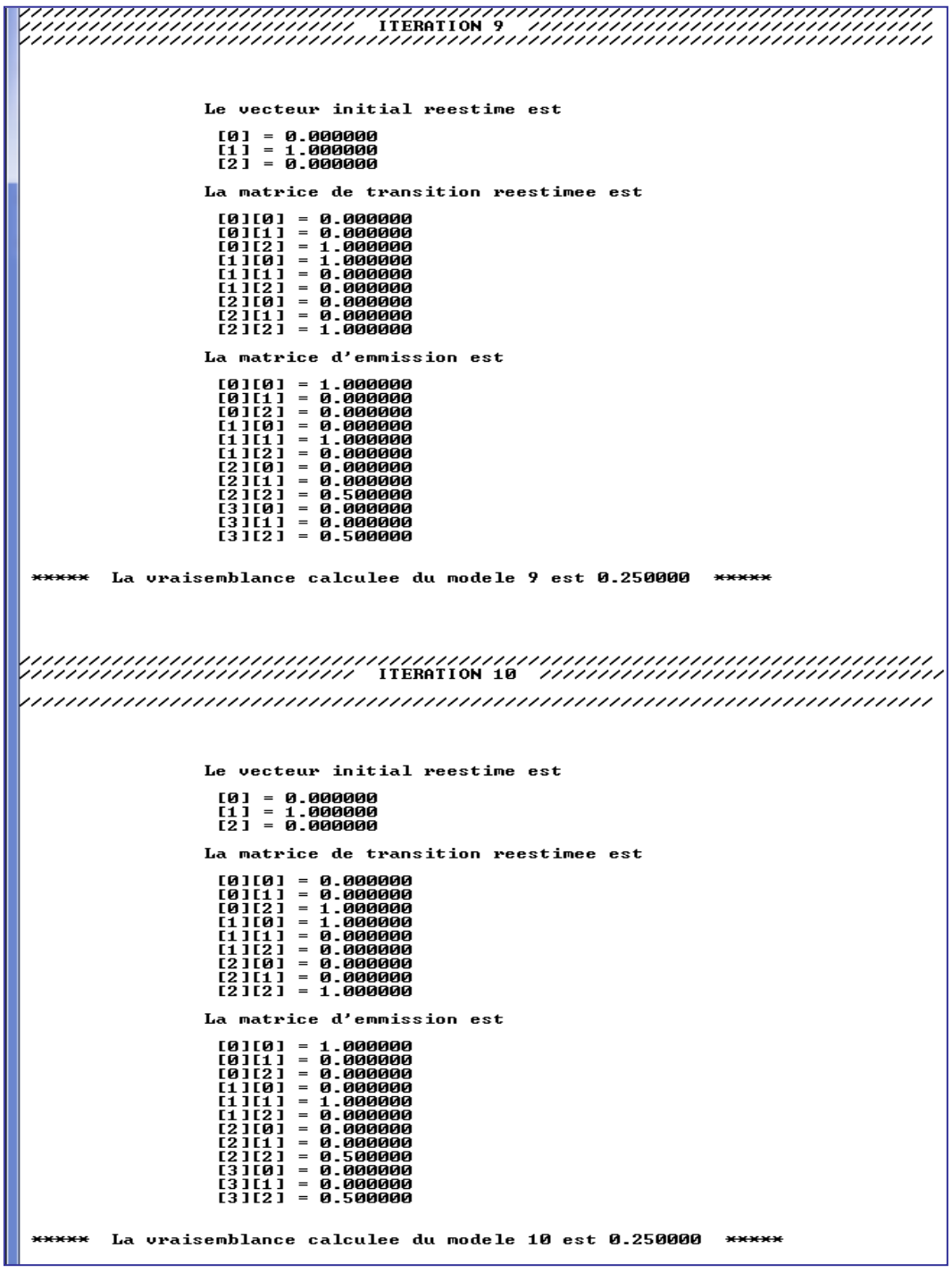

Figure 28 : Résultat du modèle MMC après 10 itérations

Donc après dix itérations, la convergence est réalisée. On va trouver le modèle λ_{10} suivant **:**

$$\prod = \begin{bmatrix} 0.0000 & 1.0000 & 0.0000 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.0000 & 0.0000 & 1.0000 \\ 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.5000 & 0.0000 \\ 0.0000 & 0.0000 & 0.5000 \end{bmatrix}$$

$$P(O \mid \lambda_{10}) = 0.250000$$

Après que tous ces calculs soient réalisés, nous constatons que le MMC étudié ne génère pas la séquence d'apprentissage avec la probabilité 1. Ceci vient du fait que le nombre d'états est trop petit pour réaliser un apprentissage avec la probabilité 1. Ce qui nous ramène à supposer qu'il faut jouer sur d'autres paramètres pour obtenir la probabilité maximale.

3.4 Conclusion

Dans ce chapitre nous avons décrit les deux modèles de Markov : modèle de Markov observable et modèle de Markov caché. Nous nous sommes intéressé plus particulièrement au modèle de Markov caché. Nous avons traité des problèmes qui sont liés à ce modèle pour enfin aboutir à une application de ce modèle réalisée en langage C.

Chapitre 4

Modèle hybride RNA et MMC pour la RAP

4.1 Introduction

Ce chapitre présente la technique hybride d'un modèle de Markov caché et d'un réseau de neurones artificiels, il est réparti en trois parties.

Dans la première partie, un bref survol des différentes approches de l'hybride RNA et les modèles de Markov caché (MMC). La deuxième partie décrit l'architecture du système proposé. Et la dernière, présente la description détaillée de cette architecture.

4.2 Nécessité de l'approche hybride RNA et MMC

L'idée générale de ces hybridations est que les réseaux de neurones sont très puissants pour capturer des structures locales (par exemple spatiales) mais pas très puissants pour capturer la structure séquentielle (même avec les réseaux récurrents), alors que les MMCs ont la caractéristique inverse : les deux se complètent donc bien. On utilise le réseau de neurones pour détecter des traits locaux, le MMC est aussi utilisé pour modéliser la structure séquentielle de ces traits locaux, et on utilise le MMC pour modéliser la structure séquentielle de ces traits (et pour incorporer facilement des connaissances à priori sur cette structure, ce qui n'est pas si facile à faire avec des réseaux de neurones). Cette idée a très bien marché pour la reconnaissance de l'écriture. Elle est aussi utilisée en reconnaissance de la parole, ainsi, il semble intéressant de tenter de combiner les capacités respectives des MMC et des réseaux de neurones, pour produire de nouveaux modèles hybrides performants qui puisent leur source dans les deux formalismes. Cependant, cette combinaison n'est pas simple à réaliser.

4.3 Types de couplage

Nous présenterons dans ce qui suit les principales approches proposées dans la litérature. Celles-ci peuvent être regroupées en trois types de couplages (figure 15) :

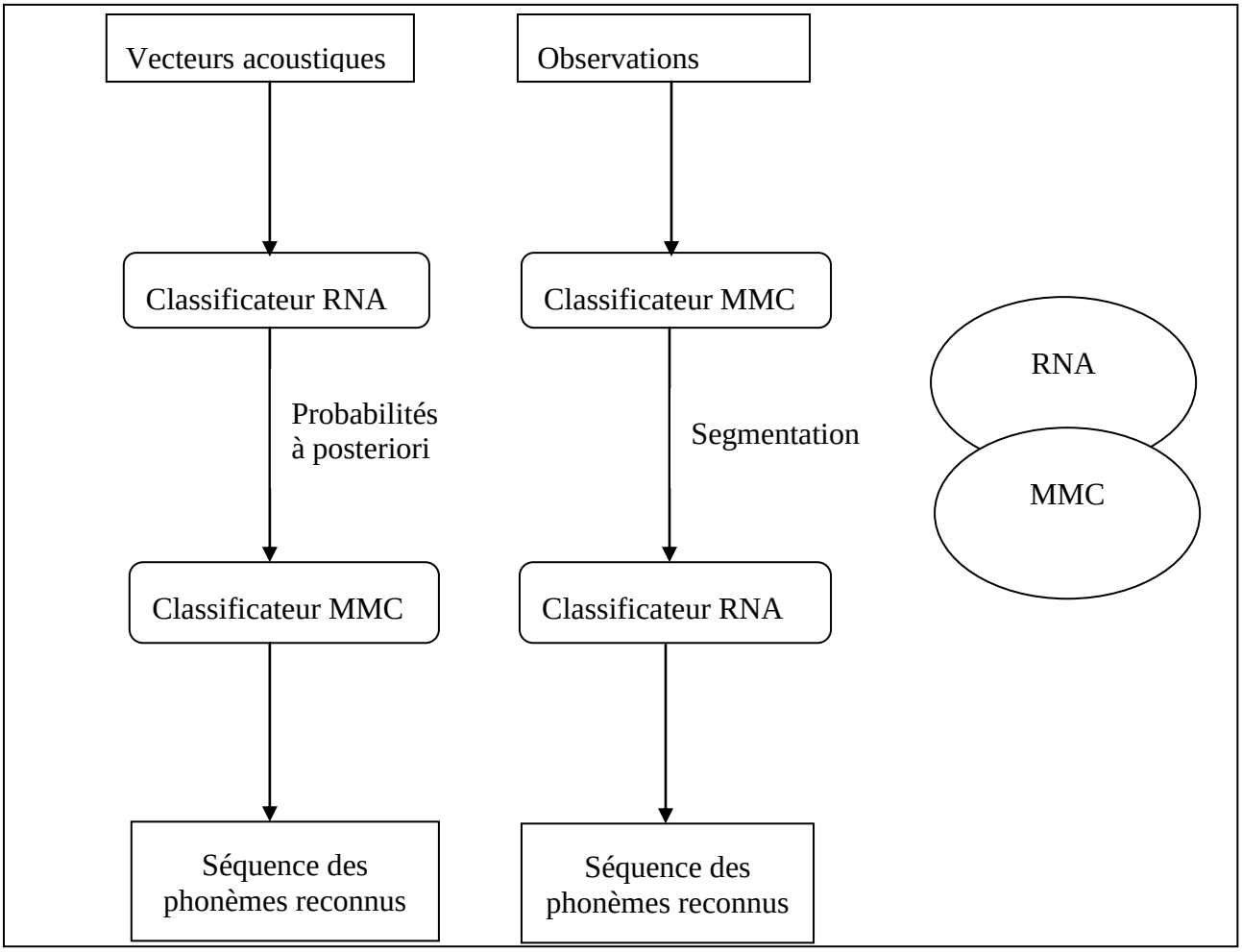

Figure 29 : modèle Amont, modèle Aval et modèle unificateur

- Réseau de neurones en amont d'un MMC :

Une première solution est d'utiliser le modèle neuronal avant le modèle probabiliste dans un modèle hybride du type RNA/MMC. Le premier sert à estimer les probabilités a posteriori $P(C_i / O(t))$ d'apprentissage du vecteur d'entrée (vecteur des paramètres acoustiques $O(t)$ à l'instant t) à une classe C_i.

- Réseau de neurones en aval d'un MMC :

Une deuxième solution est d'utiliser le modèle neuronal après le modèle probabiliste dans un modèle hybride que nous désignons par MMC/RNA le classificateur neuronal est alors utilisé pour étiqueter la segmentation issue de la classification probabiliste des vecteurs acoustiques de paroles.

- Modèles tentant d'unifier les deux théories

Une autre voie d'hybridation consiste à concevoir des modèles s'appuyant sur les deux théories des modèles de Markov et des réseaux de neurones artificiels de façon a progresser vers un formalisme unique. Dans cette catégorie, il est démontré que l'algorithme d'apprentissage de MMC avant-arrière est équivalent à l'algorithme de rétropropagation du gradient d'erreur.

Dans toute la suite, on propose une hybridation de type RNA/MMC, le réseau de neurones artificiels vu comme un classificateur a C classes, le réseau neuronal proposé est de type PMC (perceptron multicouche).

4.4 Chaîne de traitement

Le processus de reconnaissance automatique de la parole étant relativement complexe, le processus sera donc décomposé en sous-problèmes que l'on appellera modules. Chaque module traite une phase de traitement et fournit des résultats acceptables pour la phase suivante on propose dans la suite une hybridation de type PMC/MMC capable de reconnaitre une séquence des phonèmes prononcés, suivie par un module de décision capable de décider si la suite des phonèmes est bien prononcée.

La chaîne est composée de quatre modules (figure.16)

- Le module de paramétrisation transforme le signal de la parole en une suite de vecteurs acoustiques $O = (o_1, o_2, ..., o_T)$ (chapitre 1).
- La deuxième étape est constituée d'un réseau de neurones artificiels. Ce dernier permet d'estimer localement la probabilité qu'un vecteur acoustique appartienne à une classe représentant un phonème. Par la suite, chacun des états MMC sera associé à une

de ces classes, le réseau de neurones artificiels est donc vu comme un classificateur à p classes.

- L'avant dernier module est un modèle de Markov caché (MMC). Il permet d'exploiter globalement les probabilités en trouvant le chemin optimal à travers les probabilités estimées lors de l'étape précédente.
- Finalement, un module de décision permettra d'évaluer un mot donné au niveau des phonèmes qui le constituent (chaque mot est construit par un certain nombre de phonèmes). Il repose sur des bases statistiques faisant appel aux tests d'hypothèses et aux mesures de confiance basées sur les probabilités a posteriori ou les rapports de vraisemblance.

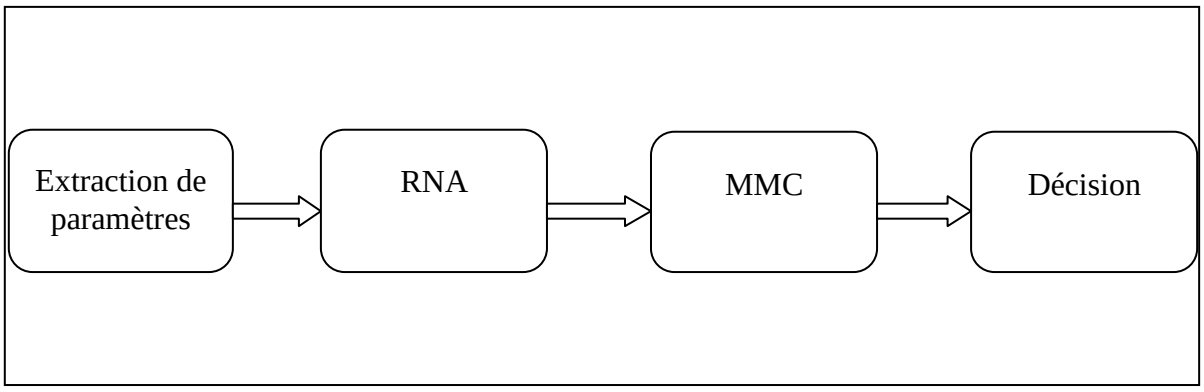

Figure 30 : chaîne de traitement

4.5 Modèle hybride RNA/MMC

Les réseaux de neurones présentent beaucoup d'avantages, mais ils ne peuvent modéliser une évolution temporelle à long terme. Ils sont donc mal adaptés au traitement de signaux séquentiels tels que la parole. Par contre, les modèles de Markov modélisent très bien ce genre de problèmes grâce à la topologie imposée. Une combinaison des deux systèmes semble donc appropriée afin de tirer parti de ces avantages. Le réseau de neurones artificiels est utilisé en tant qu'estimateur des probabilités d'émission pour le modèle de Markov.

Dans l'approche hybride RNA/MMC, les réseaux de neurones artificiels (PMC) sont utilisés comme estimateurs des probabilités à posteriori. Chaque classe étant associée à l'un des $\lambda_i \quad i = 1,..., p$ décrivant l'ensemble des modèles MMC. Le réseau de neurones artificiels

représente alors T neurones à son entrée (T présente le nombre des vecteurs acoustiques issus de la phase de paramétrisation), et p neurones de sortie (p le nombre de classes c-à-d, le nombre des MMC utilisés).

A partir d'une séquence d'obsérvations, le réseau de neurones va calculer les probabilités a posteriori $P(C_i \mid o_t)$ des états des MMC en fonction de l'observation. Selon la règle de Bayes, on peut exprimer la probabilité d'une classe C_i sachant l'entée o_t , par l'équation suivante :

$$P(C_i \mid o_t) = P(q_i \mid o_t) = \frac{P(o_t \mid q_i).P(q_i)}{P(o_t)}$$

Le terme $P(o_t)$ est indépendant de la classe, il peut donc être ignoré pour la classification.

$P(q_i)$ représente la probabilité a priori de la classe, elle permet d'obtenir des vraisemblances normalisées c'est-à-dire $\frac{P(q_i \mid o_t)}{P(q_i)} = \frac{P(o_t \mid q_i).P(o_t \mid q_i)}{P(o_t)}$. Et $P(o_t \mid q_i)$ est la probabilité de l'observation o_t dans l'état q_i c'est-à-dire dans la notation de (chapitre 3) est $b_i(o_t)$. Et c'est à partir des vraisemblances normalisées $b_i(o_t)$, que le MMC effectue la tâche de reconnaissance.

4.6 Apprentissage séparé

La première solution est d'effectuer un apprentissage au niveau d'unité de base. C'est l'approche la plus simple à réaliser. Cependant, comme le rôle du classifieur est d'estimer les probabilités de reconnaissance au niveau d'unité de base, elle nécessite une segmentation préalable de la base des mots en unités de base. Par ailleurs, cette procédure demande plusieurs itérations du processus apprentissage/reconnaissance/étiquetage pour obtenir une segmentation correcte et des résultats satisfaisants.

Ensuite, les probabilités de transitions des états des MMC peuvent être estimées en utilisant les algorithmes de Baum-Welch ou avant-arrière à partir de données mots comme dans apprentissage classique de MMC.

Un tel apprentissage pose différentes difficultés.

La première est de transformer les probabilités a posteriori estimées par le RNA en vraisemblances normalisées.

Selon le théorème de Bayes, la vraisemblance de l'observation o_t pour le modèle λ s'écrit : $P(o_t \mid \lambda) = \frac{P(\lambda \mid o_t).P(o_t)}{P(\lambda)}$

Or, il est difficile d'estimer la densité de probabilité $P(o_t)$. Il est souvent défini une pseudo vraisemblance normalisée :

$$\hat{P}(o_t \mid \lambda) = \frac{P(\lambda \mid o_t)}{P(\lambda)}$$

Le second problème est de savoir traiter les entrées non rencontrées pendant l'apprentissage car elles ne correspondent à aucune étiquette réelle d'unité de base. Une technique est d'utiliser en sortie du réseau de neurones une classe de rejet.

4.7 conclusion

Dans ce chapitre nous avons décrit les différents types de couplages d'hybridation d'un modèle de RNA et le modèle de Markov caché. Nous avons traiter plus particulièrement l'approche RNA|MMC.

Conclusion

Nous avons décrit dans ce mémoire les modèles les plus utilisés en reconnaissance de la parole. Ces modèles sont à la base de la plupart d'outils de reconnaissance de la parole existants. Néanmoins, la mise en œuvre de ces modèles nécessite des hypothèses contraignantes qui peuvent être adoucies en utilisant conjointement les modèles de Markov cachés et un réseau de neurones artificiels. Ces modèles sont appelés des modèles hybrides et permettent d'obtenir de meilleures performances que les modèles MMC classiques. Les avantages dans l'utilisation cumulée des modèles de Markov cachés et des réseaux de neurones se reflètent clairement dans les expériences réalisées sur des corpus de parole isolée et continue pour de petits et grands vocabulaires. Mais il existe quand même quelques inconvénients, ces quelques inconvénients peuvent être contournés.

L'entraînement du réseau est un processus compliqué. Il arrive souvent que le réseau ait tendance à se spécialiser uniquement dans les exemples qui lui sont présentés, perdant ainsi son caractère de généralisation. Pour pallier à ce problème, une partie de la base d'entraînement est généralement utilisée pour tester le réseau.

Le problème principal associé à l'utilisation des réseaux de neurones en reconnaissance automatique de la parole, est lié à la phase d'entraînement. En effet, cette phase est beaucoup plus longue que celle correspondant aux méthodes discrètes et continues. Ceci est principalement dû à l'algorithme de rétropropagation du gradient utilisé lors de l'entraînement de réseaux à grande architecture.

Bibliographie

[1] L. R. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition, Proceedings of the IEEEE, vol. 77, no 2, Feb 1989.

[2] Adrien VERGÉ, Réseaux de neurones artificiels, 2009.

[3] Etienne TISSERAND - Jean PAUTEX, Analyse et traitement des signaux, Méthodes et applications au son et à l'images- DUNOD.

[4] app, Apprentissage artificiel.

[5] G. Dreyfus, J.-M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria Apprentissage statistique, réseaux de neurones, cartes topologiques et Machines à vecteurs supports.

[6] Emilie POISSON, Architecture et Apprentissage d'un Système Hybride Neuro-Markovien pour la Reconnaissance de l'Écriture Manuscrite En-Ligne- Thèse de Doctorat.

[7] Christian GAGNÉ, Modèles de Markov cachés, 2009.

[8] Olivier Lévêque, Probabilités et Calcul Stochastique, 2005.

[9] Intelligence Artificielle et Informatique Théorique- Cepadués Editions.

[10] Les réseaux de neurones- presse universitaire de Grenoble.

[11] Marc PARISEAU, Le perceptron multicouche et son algorithme de rétropropagation des erreurs, 2004.

[12] Marc PARISEAU, Réseaux de neurones, 2004.

[13] Méthodes d'estimation rapide de vraisemblance – Rapport bibliographique.

[14] Joël Le Roux, Modèles de Markov cachés, 2003.

[15] Mohamed ETTAOUIL et Hassan LAASSAIRI, Combination des modèles de Markov cachés et modèles de Neurones multicouches, Application à la RAP, 2005.

[16] Mohamed ETTAOUIL et Mohamed BADAOUI, Système hybride, Réseaux de Neurones et Modèle de Markov Caché, Application à la RAP, 2005.

[17] Précis de recherche opérationnelle.

[18] Programmation mathématiques- Tome 1.

[19] Programmation mathématiques- Tome 2.

[20] Reconnaissance automatique de la parole à partir de segments acoustiques et de modèles de Markov cachés.

[21] Claude BARRAS, Reconnaissance de la parole continue : adaptation au locuteur et contrôle temporel dans les modèles de Markov cachés- Thèse de Doctorat.

[22] Reconnaissance de mots connectés indépendamment du locuteur par des méthodes statistique- Thèse de Doctorat.

[23] Laurent BUNIET, Traitement automatique de la parole en milieu bruité : étude de modèles connexionnistes statiques et dynamiques.

[24] Traitement de la parole, ouvrage presses polytechniques et universitaires Romandes, collection électricité.

[25] Une Approche théorique de l'apprentissage connexionniste ; Application à la reconnaissance de la parole. Thèse de Doctorat centre d'Orsay.

[26] Yassine BENAYAD, Détection des mots clés dans un flux de parole, 2003.

[27] ALAIN BOUCHER - NGHIEM Anh Tuan, Reconnaissance d'écriture manuscrite, 2005