ORSAY
n° d'ordre: 2311

UNIVERSITE DE PARIS-SUD
CENTRE D'ORSAY

THESE

présentée
pour obtenir

Le grade de Docteur en Sciences
de l'Université Paris XI Orsay

par

Laurence DEVILLERS

Sujet

Reconnaissance de parole continue avec un système hybride neuronal et markovien

soutenue le 6 Novembre 1992 devant la Commission d'examen composée de

Mr.	Gérard LIGOZAT	Président
Mr.	Patrick GALLINARI	Rapporteur
Mr.	Jean-Pierre TUBACH	Rapporteur
Mr.	Alex WAIBEL	Examinateur
Mr.	Gérard CHOLLET	Examinateur
Mr.	Jean-Sylvain LIENARD	Examinateur

Ce travail a été mené dans le cadre d'un contrat européen ESPRIT POLYGLOT en collaboration avec le laboratoire de recherche en parole de Philips à Aix-la-Chapelle .

Remerciements

Je tiens à remercier les membres de mon jury de thèse,

Monsieur Gérard Ligozat, Professeur à Paris XI, qui a présidé mon jury de thèse,

Monsieur Alexander Waibel, Professeur à Karlruhe (Allemagne), Chercheur à CMU (USA) et ATR (Japon), dont les travaux innovateurs sur les réseaux neuronaux appelés TDNNs ("Time Delays Neural Network) ont guidé de nombreuses recherches, école que j'ai suivie durant cette thèse. Je le remercie pour son soutien, ses conseils scientifiques et son humour pétillant.

Monsieur Jean-Pierre Tubach, Professeur à l'ENST, rapporteur de cette thèse, pour ses critiques constructives tant sur le plan scientifique que formel,

Monsieur Patrick Gallinari, Professeur à Paris VI, rapporteur de cette thèse, pour son aide précieuse et son enthousiasme,

Monsieur Gérard Chollet, Chercheur à l'ENST, pour son soutien et nos discussions,

Monsieur Jean-Sylvain Liénard, Directeur de recherche au LIMSI, qui a accepté de diriger cette thèse, merci pour sa confiance,

Je tiens à remercier Monsieur Joseph Mariani, Directeur du LIMSI, Madame Françoise Néel, Directeur du groupe Communication Homme-Machine, pour leur soutien durant ces années et pour le dynamisme scientifique et l'esprit d'équipe qui règne au laboratoire,

Enfin, je tiens à remercier tout particulièrement

Christian Dugast, Docteur-Ingénieur à Philips (Aix-la-Chapelle-Allemagne), pour la collaboration fructueuse menée durant le projet européen Polyglot et pour son amitié,

Pascal Blanchet et Gilles Adda pour leur soutien scientifique et moral tout au long de cette thèse,

Léon Bottou, Patrick Haffner, Fréderick Bimbot, Jean-François Jodouin pour nos discussions,

Et toute l'équipe chaleureuse du LIMSI avec qui j'ai vécu cette expérience importante. Merci à tous, à Martine, Martine et Martine, à Arno, Philippe, Bertrand, Dominique, Hélène, Sophie, Anne, Sophie-Anne, Jean-Claude, Fredérique, Gaël, Christian, Vincent, Mr Demars, Jean-Luc, Lori, Christophe, Michèle, Maxine, à Bernard, Christophe, Françoise, Mr et Mme Chastagner... et à tous les autres, anciens et nouveaux ...

A Jean-luc,
à mes parents,

Résumé

L'objet de cette étude est d'évaluer les performances de réseaux neuronaux de type TDNNs *"Time Delay Neural Networks"* pour la reconnaissance automatique de parole continue. La question posée est de savoir si ces méthodes sont une alternative aux méthodes classiques de reconnaissance de la parole telles que les modèles markoviens ou plus vraisemblablement un complément.

Un système de reconnaissance automatique de parole continue a pour but de transformer un signal acoustique en phrases graphémiques. Ce travail porte principalement sur l'etude de méthodes d'apprentissage pour la phase de décodage "acoustico-phonétique" du signal. Une phase d'apprentissage sur des données acoustiques dont on connait la transcription phonétique permet de mémoriser des correspondances entre parties de signal et unités phonétiques : phones, diphones ou mots. L'unité choisie dans cette étude est le phone, unité minimale qui permet de différencier la prononciation de deux mots. Lors de la phase de reconnaissance, l'énoncé inconnu est étiqueté phonétiquement, puis les unités phonétiques sont intégrées en mots enfin en phrases en utilisant différentes sources de connaissances. Ces connaissances sont par exemple un dictionnaire de transcription des mots de phonème en graphème, un modèle de langage, etc...

L'aspect discriminant des critères d'apprentissage des réseaux de neurones est un des côtés les plus attractifs de ces modèles, qualité qui fait en général défaut aux modèles markoviens. Les réseaux de neurones de type MLPs "Multi-Layer Perceptrons" ou TDNNs utilisant la rétro-propagation du gradient ont cependant deux inconvénients majeurs qui sont le temps prohibitif d'apprentissage et le manque de mécanisme de gestion des distorsions temporelles et d'intégration de connaissances de différents niveaux. Le TDNN par rapport au MLP prend en compte un contexte temporel sur les différentes couches et a la propriété d'invariance par translation temporelle grâce au partage des matrices de connexions. Il ne possède cependant pas non plus de mécanisme d'alignement temporel permettant de gérer des problèmes de distorsion temporelle et de reconnaissance de parole continue.

Les axes principaux de ce travail répondent à ces problèmes. D'une part, La conception de structures de réseaux neuronaux modulaires permet l'apprentissage de grandes bases de données avec des temps d'apprentissage raisonnable. D'autre part, la coopération entre réseaux

neuronaux et modèles markoviens dans des systèmes hybrides a pour but de combiner les qualités de discrimination des réseaux et de modélisation des systèmes de Markov et de montrer la complémentarité de ces deux approches. L'intégration des sorties des réseaux dans un formalisme HMMs "Hidden Markov Models" permet d'utiliser les possibilités des modèles markoviens pour gérer l'aspect temporel des formes et intégrer au système de reconnaissance des connaissances de différents niveaux : lexicales, syntaxiques,etc...

La modularité apporte une solution au problème d'apprentissage de grandes bases de données avec des réseaux neuronaux. Nous avons proposé deux architectures modulaires originales. L'une explore les capacités d'une architecture modulaire n'utilisant pas une partition de classes particulière. L'autre est une architecture hiérarchique de sous-réseaux. Les architectures neuronales ont été ensuite intégrées dans des systèmes hybrides.

La première architecture que nous proposons, appelée FE-TDNN pour "*Features Extracted TDNNs*" permet d'éliminer l'utilisation a priori d'une partition de classes particulière. Les sous-réseaux classifient des sous-ensembles de phonèmes comprenant des recouvrements entre les classes d'où l'appellation de macro-classes croisées. L'architecture est composée de sous-réseaux de macro-classes qui extraient différents indices comme le voisement, le mode d'articulation, la place de l'articulation... Le réapprentissage des sous-réseaux dans une structure globale permet de détecter une forme si plusieurs indices sont détectés simultanément. Des travaux complémentaires sont cependant nécessaires pour apporter des résultats concluants.

Le deuxième type d'architecture étudié est un arbre hiérarchique de réseaux, appelé HT-TDNN pour "*Hierarchical TDNNs with Traps*". La tâche globale est scindée en sous-tâches, chacune d'elles étant résolue par un sous-réseau. Un réseau de macro-classes correspondant à la partition choisie sert de noeud principal à l'arborescence. L'originalité de cette construction vient de l'ajout aux sorties de chaque réseau "feuille" d'une sortie de rejet appelée trappe qui permet de donner au réseau une connaissance globale de toutes les formes à reconnaître. Chaque réseau a alors la connaissance des phonèmes qui ne sont pas dans sa propre classe et il n'est plus nécessaire de faire un réapprentissage global de tous les sous-réseaux. La structure HT-TDNN est extrêmement souple. Elle élimine la limite du nombre de sorties et du nombre de sous-réseaux qui existe dans une structure avec réapprentissage global. Il devient donc possible de multiplier le nombre de sous-réseaux appris dans des contextes précis ou correspondant à des sous-réseaux discriminants des diphones et de les intégrer ensuite très simplement. Cette architecture a permis d'obtenir de nettes améliorations des taux de classification des segments phonétiques par rapport à un réapprentissage global des sous-réseaux. Une architecture simplifiée des HT-TDNNs ne comportant pas de trappes a aussi été testée. Cette architecture

notée H-TDNN pour "Hierarchical TDNNs" offre des résultats un peu moins bons mais elle est en revanche beaucoup moins coûteuse en temps calcul.

Afin de traiter de la reconnaissance automatique de parole continue, nous avons développé des systèmes hybrides alliant les capacités de discrimination des réseaux et la gestion temporelle ainsi que les mécanismes d'intégration des HMMs.

Il existe différents types de coopérations entre modèles markoviens et réseaux de neurones :

- les modèles neuronaux peuvent être utilisés comme pré-traitement de modèles markoviens ;
- ils peuvent participer avec des modèles markoviens à la phase d'apprentissage, on parle alors de coopération optimale;
- enfin, il peut aussi servir à un apprentissage de type "*corrective training*" c'est-à-dire à un post-traitement des modèles markoviens.

Nous avons étudié le premier type d'intégration où les réseaux sont utilisés comme pré-traitement des HMMs. Les scores de sorties des réseaux sont alors considérés comme des approximations de probabilités a posteriori, qui divisés par la probabilité a priori, sont intégrés directement dans un formalisme markovien comme des probabilités d'émission associées aux états ou aux transitions.
Nous avons proposé un autre type de coopération des deux méthodes. Les probabilités issues des deux modèles, connexionniste et markovien sont utilisées en coopération lors de la phase de reconnaissance. Une simple combinaison linéaire des probabilités permet de mesurer l'apport de chacun des modèles.

Les modèles markoviens utilisés dans cette étude ont été développés par un laboratoire de recherche de Philips de Aix-la-Chapelle. Les tests ont été menés pour trois locuteurs de la base de données DARPA RM1 "speaker dependent" de 1000 mots de vocabulaire, ce qui a permis des comparaisons au niveau international avec les meilleurs systèmes actuels.

Les résultats donnés sont obtenus pour les trois locuteurs testés. Les taux de reconnaissance obtenus sur cette base de données significatives, avec un système hybride utilisant la coopération des probabilités issues de HMMs et de H-TDNNs lors de la reconnaissance, ont permis de montrer la complémentarité des deux approches puisqu'on améliore notablement (20%) les performances de modèles markoviens utilisés seuls dont les scores initiaux d'erreurs sur les mots étaient de l'ordre de 3.2%.

Peu de résultats montrent à l'heure actuelle de meilleures performances avec des méthodes neuronales hybrides que celles obtenues avec des systèmes basés sur les modèles markoviens seuls. L'un des l'objectifs de notre étude était justement de construire un système hybride neuronal et markovien, tirant partie des qualités des deux systèmes et offrant de meilleures performances que l'un ou l'autre pris séparément. Nous avons obtenu des résultats particulièrement intéressants en faisant coopérer lors de la reconnaissance les probabilités issues de modèles neuronaux et markoviens. Ces résultats sont d'autant plus encourageants qu'ils ont été obtenus avec une architecture neuronale modulaire assez simple qui devrait pouvoir être optimisée.

Les systèmes hybrides et modulaires sont certainement une voie de recherche prometteuse. Notre travail a été une exploration de cette voie qui s'est ouverte récemment et qui suscite l'intérêt de nombreux chercheurs à l'heure actuelle. De nombreux travaux sont cependant encore nécessaires pour développer d'autres algorithmes et améliorer les structures actuelles, et pour avoir plus de certitudes quant à l'apport des réseaux et des méthodes hybrides dans le domaine de la reconnaissance de la parole.

Table des matières

Partie 3: Reconnaissance de parole continue et Systèmes hybrides

1. Introduction

1.1 Contexte de recherche

On assiste depuis quelques années à un regain d'intérêt pour les méthodes connexionnistes en reconnaissance des formes. Retour historique, car les pionniers présentaient les premiers réseaux il y a 30 ans. Le succès actuel de ces méthodes neuronales est dû à de nouvelles structures et algorithmes. Nous nous intéresserons, dans cette étude, aux réseaux les plus performants à l'heure actuelle, les réseaux LVQ *"Learning Vector Quantization"* [Kohonen 88] ainsi que le perceptron multicouche MLP *"Multi-Layer Perceptron"* [Rumelhart 86] et ses dérivés comme les réseaux à délai temporel TDNN *"Time-Delay Neural Network"* [Waibel 87] .

La reconnaissance de la parole est une application particulièrement difficile de la reconnaissance des formes. Son but est de transformer un signal acoustique en phrases graphémiques. Une phase d'apprentissage sur des données dont on connaît la transcription permet d'apprendre des correspondances entre partie de signal et unités linguistiques : par exemple phones, diphones ou mots. L'unité souvent choisie est le phone, unité minimale qui différencie la prononciation de deux mots. Cette phase correspond à une classification phonétique. Une deuxième phase de traitement consiste ensuite à intégrer les unités phonétiques en phrases lors de la reconnaissance. Plusieurs sources de connaissances peuvent être utilisées : un lexique, une grammaire syntaxique... Cette étude porte principalement sur le décodage acoustico-phonétique. La complexité de la classification phonétique est due en grande partie à l'importante variabilité fréquentielle et temporelle du signal de

parole, intra et inter locuteur ; chaque son élémentaire est déformé par les sons qui l'entourent par un effet de coarticulation.

Les systèmes les plus connus en reconnaissance des formes pour le traitement de données temporelles sont les modèles markoviens cachés, HMMs pour "*Hidden Markov Models*" [Baker 75], [Jelinek 76]. Ils ont permis ces quinze dernières années d'améliorer les performances en reconnaissance de la parole, tant au niveau de la taille des vocabulaires que des applications multi-locuteurs. Plus récemment, les réseaux neuronaux de type TDNN et LVQ ont été appliqués avec succès à des tâches difficiles mais limitées à des phonèmes [Waibel 89], [McDermott 89] et des mots isolés [Franzini 89], [Bottou 90].

L'aspect discriminant des critères d'apprentissage des réseaux de neurones est un des côtés les plus attractifs de ces modèles. Les classifieurs discriminants ont des avantages évidents sur les méthodes non discriminantes lorsqu'il s'agit de problèmes réels. En effet, ils font en général moins d'hypothèses sur la distribution des classes et sont plus robustes aux variabilités des données puisqu'ils sont directement reliés à l'erreur de classification. Les systèmes de reconnaissance basés sur une modélisation HMM sont souvent reliés au critère non discriminant de "*Maximum Likelihood Estimation*" MLE. De récentes recherches ont été menées sur l'utilisation de critères discriminants avec un formalisme markovien, citons par exemple le critère de "*Maximum Mutual Information*" MMI [Brown 87], [Merialdo 88], [Normandin 91] ou de "*Corrective Training*" [Gauvain 92] qui permettent de réestimer, après un apprentissage de type MLE, les paramètres des HMMs. Ces méthodes sont cependant souvent difficiles à mettre en oeuvre.

Les réseaux de neurones MLP, TDNN ou LVQ sont souvent comparés aux modèles Markoviens (HMMs) en reconnaissance de la parole. Cependant, leur principal inconvénient hormis le temps d'apprentissage porte sur la modélisation temporelle. En réalité, la comparaison HMMs et NNs n'a de sens que pour la classification de formes statiques puisque à l'heure actuelle les réseaux ne possèdent pas de mécanisme d'alignement temporel efficace pour intégrer des séquences. Tandis que les modèles markoviens cachés sont typiquement des systèmes capables de gérer des séquences d'événements. L'algorithme de reconnaissance de Viterbi, classiquement utilisé dans les HMMs, permet de faire une approximation de la meilleure séquence d'états des modèles à partir des distributions de probabilités d'émission associées à chaque état. Une véritable comparaison se situe au niveau de la modélisation de ces distributions. Elles peuvent être des mélanges de densités continues "*continuous mixture densities*", des distributions gaussiennes

univariables ou aussi des sorties de classifieurs discrimants tels que les MLPs, TDNNs... On parle alors de système hybride neuronal et markovien. Il existe aussi d'autres possibilités de coopération entre réseaux de neurones et modèles markoviens, par exemple celle d'utiliser un réseau en post-traitement d'un HMM comme module de correction d'erreurs.

A l'heure actuelle, de nombreux systèmes hybrides neuronaux et markoviens, dont le but est d'allier les qualités respectives de discrimination et de gestion temporelle des deux modèles, ont été proposés [Bridle 90], [Franzini, Waibel 90], [Niles 90], [Haffner 90]... Les réseaux de neurones ont malheureusement le défaut de nécessiter des temps d'apprentissage très longs mais ils se prêtent très bien aux implémentations sur des architectures massivement parallèles. Les performances sur des bases de données de taille importante, monolocuteur ou multilocuteur, comme TIMIT, DARPA "*Ressource Management*", ont été principalement obtenues avec des systèmes hybrides réalisés sur des architectures multiprocesseurs telles qu'une machine utilisant 5 "Ring Array Processors" contenant 20 TI TMS320C30 DSPs [Renals 92], ou une architecture de 64 "*transputers*" T800 [Robinson, Fallside 91]... Ces travaux permettent d'obtenir des résultats plus performants que ceux des meilleurs HMMs, avec tout de même d'énormes moyens mis en oeuvre.

De nombreuses recherches sont menées dans ce domaine actuellement. La question que l'on peut se poser est de savoir si les réseaux sont une alternative aux méthodes classiques de reconnaissance de la parole ou plus vraisemblablement un complément.

1.2 Objectif de l'étude

Notre objectif a été d'utiliser au mieux les capacités des réseaux de neurones afin de construire un système hybride qui améliore les performances des modèles markoviens utilisés seuls. Peu d'études ont évalué les résultats des systèmes hybrides neuronaux et Markoviens par rapport à des HMMs très performants utilisant par exemple des mélanges de gaussiennes "*Gaussian mixture HMM classifier*"; c'est le but que nous avons poursuivi.

Le système hybride construit est composé d'un réseau de neurones formels de type perceptron multicouche, MLP pour "*Multi-Layer Perceptron*", pour ses capacités de discrimination et d'un modèle markovien caché pour son pouvoir de modélisation temporelle et d'intégration des connaissances acoustiques, syntaxiques... Le réseau

de neurones développé est une variante du MLP, le TDNN pour "*Time Delay Neural Network*" [Waibel 87]. Les ordinateurs utilisés n'ayant pas de structure multiprocesseur, des architectures de réseaux de neurones modulaires ont été développées pour apprendre une base de données de taille relativement importante. La décomposition de la tâche de classification en sous-tâches est un moyen de faciliter l'apprentissage sur une machine monoprocesseur. Chaque réseau dédié à une sous-tâche est entraîné séparément et réutilisé ensuite avec tous les autres sous-réseaux dans une structure globale. La modularité de la structure apporte une diminution du temps et une amélioration de la qualité de l'apprentissage.

Les axes de ce travail sont l'étude de structures de réseaux neuronaux modulaires permettant l'apprentissage de grandes bases de données sans machine spécialisée et l'étude de la coopération entre un modèle neuronal et markovien dans un système hybride. Ces thèmes de recherche répondent aux principaux inconvénients des réseaux de neurones: le temps d'apprentissage et le pouvoir de modélisation temporelle.

Notre démarche a été de tester tout d'abord les qualités et les limites de différents types de réseaux, LVQ, MLP et TDNN pour une tâche de classification phonétique. Puis l'aspect modulaire des réseaux a été étudié. Enfin, différentes coopérations entre un modèle neuronal et un HMM sont décrits. L'utilisation d'une base de donnée étalon de 1000 mots en américain, "*DARPA Ressource Management Speaker Dependent*" [Price 88], a permis des comparaisons au niveau international avec les meilleurs systèmes actuels. Les tests ont été menés sur trois locuteurs de Darpa (JWS0, CMR0 et BEF0).

Ce travail a débuté en mai 1989 et a été poursuivi dans le cadre d'un projet ESPRIT 2104 "Polyglot", "*A multi-language speech-to-text and text-to-speech system*" en coopération avec Christian Dugast du laboratoire de recherche de Philips à Aix-la-Chapelle (Allemagne) qui a développé les modèles markoviens. Les réseaux de neurones ont été réalisés au LIMSI "Laboratoire Informatique et Mécanique des Sciences de l'Ingénieur" en langage C sur VAX/VMS, puis sur station de travail SUN Sparc/UNIX et sont exploités actuellement sur MicroVAX/Ultrix à Aix-la-Chapelle.

1.3 Plan

Cette thèse comporte 3 parties suivies par une conclusion de l'étude et les perspectives envisagées.

La **première partie** est une description du contexte de l'étude: complexité de l'application, techniques de reconnaissance des formes avec des algorithmes classiques et des réseaux connexionnistes et état de l'art concernant les systèmes hybrides en reconnaissance de la parole. La **deuxième partie** est une suite d'expériences de classification phonétique visant à montrer les principaux avantages et inconvénients des réseaux appliqués à la parole. Afin de gérer le problème du temps d'apprentissage avec des réseaux, nous proposons plusieurs architectures modulaires de réseaux. La **troisième partie** porte sur les systèmes hybrides. Afin de gérer des problèmes de parole continue, on allie la classification statique fourni par un réseau de neurones à la modélisation temporelle obtenue grâce à des HMMs.

Partie 1: Présentation des techniques de classification pour la reconnaissance de la parole

Une présentation des méthodes de reconnaissance de la parole fait l'objet du **chapitre 2** qui permet aussi de situer les modèles connexionnistes parmi les méthodes classiquement utilisées. Après une introduction générale sur les modèles connexionnistes, nous détaillons dans **le chapitre 3** les algorithmes développés dans cette étude. Un état de l'art des applications en reconnaissance de la parole avec des systèmes hybrides neuronaux et markoviens est décrit dans le **chapitre 4.**

Partie 2: Expériences de classification phonétique et Conception d'architectures modulaires

Le **chapitre 5** porte sur l'étude des capacités des modèles connexionnistes pour la classification phonétique et sur les mécanismes de gestion temporelle des réseaux. Les expériences relatées dans ce chapitre ont été principalement effectuées sur une base de données monolocuteur en français d'environ 250 mots [Adda 88]. Cette base est décrite dans l'annexe 1.

Des comparaisons ont été effectuées entre l'algorithme de LVQ2 pour "*Learning Vector Quantization*" qui est une version sophistiquée de la LVQ, une structure de cartes topologiques [Kohonen 84], l'algorithme de rétro-propagation du gradient

d'erreur utilisé avec un réseau perceptron multicouche et des méthodes plus classiquement utilisées en reconnaissance des formes.

Pour traiter des problèmes de reconnaissance de la parole, il est nécessaire de prendre en compte l'aspect temporel des formes. Les formes consécutives sont dépendantes. En exploitant le contexte, on améliore les performances du classifieur. Le TDNN prend en compte un contexte temporel sur les différentes couches et a la propriété d'invariance par translation temporelle grâce au partage des matrices de connexions. Il a une gestion des connexions entre les couches de neurones différente du MLP, ce qui lui confère un potentiel d'abstraction supérieur. Il ne possède cependant pas de mécanisme d'alignement temporel permettant de gérer des problèmes de distorsion temporelle et de reconnaissance de parole continue.

Le **chapitre 6** porte sur l'apprentissage de grandes bases de données avec des réseaux. Les réseaux utilisés dans la suite de cette étude sont de type TDNNs. La base de données testée est la base "*DARPA Ressource Management 1*", les tests effectués étant toujours monolocuteurs.

Nous avons étudié les difficultés inhérentes à l'apprentissage d'une grande base de données de 1000 mots de vocabulaire. Pour un locuteur, on dispose de 600 phrases d'apprentissage soit 25 minutes de parole continue. Cette base en américain contient douze locuteurs, nous en testerons trois. Elle est décrite en détail en annexe 2. Dans le but de construire ensuite des architectures modulaires qui permettent de diminuer le temps nécessaire à l'apprentissage de grandes bases de données, nous avons entraîné des réseaux de petite taille, appelés sous-réseaux, pour la classification de sous-ensembles phonétiques. Différents prétraitements des données ont été testés et ont permis d'améliorer les performances de classification des sous-réseaux. Les problèmes d'optimisation de l'algorithme d'apprentissage de rétropropagation du gradient et de choix de structure des TDNNs sont aussi relatés dans ce chapitre. Ils sont d'autant plus critiques que la taille de la base de données et le nombre de poids de connexions dans le réseau sont importants.

Le **chapitre 7** apporte une solution au problème d'apprentissage de grandes bases de données : la modularité. La notion de modularité permet de concevoir des systèmes correspondant à la décomposition d'une tâche complexe en sous-tâches enchainées séquentiellement ou réalisées en parallèle. Elle peut être vue comme un ajout de connaissances a priori qui permet de diminuer le temps d'apprentissage. Le principal problème consiste à gérer l'ensemble des sous-réseaux pour obtenir une

classification globale. Deux nouvelles architectures modulaires, avec et sans réapprentissage neuronal global, sont décrites dans ce chapitre.

Le premier type d'architecture avec réapprentissage neuronal est inspiré des travaux de A. Waibel [Waibel 88]. Le problème global de classification est scindé en sous-problèmes, chacun d'eux étant résolu par un sous-réseau. Les sous-réseaux sont réutilisés ensemble dans une structure globale et participe à nouveau à un apprentissage. Un réseau de macro-classes correspondant à la partition choisie pour entraîner séparément les sous-réseaux est en général adjoint à la structure. L'architecture que nous proposons, appelée FE-TDNN pour *"Features Extraction TDNNs"* permet d'éliminer l'utilisation a priori d'une partition de classes particulière. Cette architecture est composée de sous-réseaux de macro-classes qui extraient différents indices comme le voisement, le mode d'articulation, la place de l'articulation... Les sous-réseaux classifient des sous-ensembles de phonèmes comprenant des recouvrements entre les classes d'où l'appellation de macro-classes croisées. L'apprentissage de ces sous-réseaux dans une structure globale permet de détecter une forme si plusieurs indices sont détectés simultanément.

Le deuxième type d'architecture étudiée est un arbre hiérarchique de réseaux, appelé H-TDNN pour *"Hierarchical TDNNs"*. Le réseau de macro-classes sert de noeud principal à l'arborescence et permet de sélectionner un sous-réseau spécifique. Une architecture plus complexe, dérivée de cet arbre a été ensuite développée, elle est appelée HT-TDNN *"Hierarchical TDNNs with Traps"*. L'originalité de cette construction vient de l'ajout aux sorties de chaque réseau "feuille" d'une sortie de rejet appelée trappe qui permet de donner au réseau une connaissance globale de toutes les formes à reconnaître. Chaque réseau a alors la connaissance des phonèmes qui ne sont pas dans sa propre classe et il n'est plus nécessaire de faire un réapprentissage global de tous les sous-réseaux. La structure HT-TDNN est extrêmement souple. Elle élimine la restriction du nombre de sorties et du nombre de sous-réseaux qui existe dans une structure avec réapprentissage global. Il devient donc possible de multiplier le nombre de sous-réseaux appris dans des contextes précis ou correspondant à des sous-réseaux discriminants des diphones et de les intégrer ensuite très simplement. Cette architecture a permis d'obtenir de nettes améliorations des taux de classification des segments phonétiques par rapport à un réapprentissage global des sous-réseaux.

Partie 3: Reconnaissance de parole continue et Systèmes hybrides

Des systèmes hybrides TDNN-HMMs ont été développés afin d'utiliser les capacités des deux méthodes. Leur but est double, il s'agit d'une part de rendre les modèles markoviens discriminants et d'autre part d'être en mesure de gérer de la parole continue avec un réseau de neurones.

On distingue plusieurs types de coopérations entre modèles markoviens et réseaux ; Le modèle neuronal peut être utilisé comme pré-traitement d'un modèle markovien [Fallside 90], [Franzini, Waibel 91], [Devillers, Dugast 91], [Bourlard 92], les deux modèles peuvent être entraînés ensemble, on parle alors de coopération optimale [Bridle 90], [Bengio 91], [Haffner 92],, le réseau peut servir à un apprentissage de type "*corrective training*" c'est-à-dire à un post-traitement des HMMs [Niles 91], [Austin 92] et enfin les probabilités issues des deux modèles connexionniste et markovien peuvent être utilisées ensemble dans le critère de décision [Renals 92], [Dugast, Devillers 92]. Nous avons étudié plusieurs types de coopérations dans le **chapitre 8**.

Lorsque les NNs sont utilisés comme pré-traitement des HMMs, les scores peuvent être considérés comme des approximations de probabilités a posteriori, qui divisés par la probabilité a priori, sont intégrés directement dans un formalisme markovien comme des probabilités d'émission associées aux états ou aux transitions.

Un autre type d'intégration a été étudié : la coopération des probabilités issues des HMMs et des NNs lors de la reconnaissance. Cette nouvelle approche diminue de 20% le taux d'erreur des HMMs utilisant des mélanges de densités qui servent de références dans nos expériences et montre tout l'intérêt du codage discriminant des réseaux de neurones.

Conclusion

Nous ferons un bilan de l'apport des réseaux neuronaux par rapport aux modèles classiques dans nos expériences et décrirons les recherches futures envisagées dans le **chapitre 9**.

Partie 1

Méthodes de classification en reconnaissance de la parole

Cette première partie est une présentation des techniques de classification en reconnaissance de la parole, elle regroupe les aspects théoriques et état de l'art du domaine étudié. Une présentation des méthodes de reconnaissance de la parole fait l'objet du ***chapitre 2*** *qui permet de situer les modèles connexionnistes parmi les méthodes classiquement utilisées. Après une introduction générale sur les modèles connexionnistes, nous détaillons dans* ***le chapitre 3*** *les algorithmes développés dans cette étude. Un état de l'art des recherches en reconnaissance de la parole avec des systèmes hybrides neuronaux et markoviens est décrit dans le* ***chapitre 4****.*

2. Reconnaissance automatique de la parole

Un système de reconnaissance automatique de la parole contient plusieurs niveaux de traitements souvent distingués par les appellations de bas niveau et de niveau supérieur. L'expression bas niveau regroupe toutes les phases concernant le traitement du signal et sa transcription en unités symboliques. Les niveaux supérieurs correspondent aux traitements symboliques; accès lexical, analyse syntaxique et sémantique et aux traitements pragmatiques. Dans la première partie de nos expériences qui correspond à la partie 2 de cette étude, nous nous sommes plus particulièrement intéressés aux traitements dits de bas niveau, c'est-à-dire à la reconnaissance acoustique. Nous avons exploré différentes méthodes connexionnistes pour transformer le signal en unités symboliques élémentaires plus courtes que le mot, on parlera de classification phonétique. Dans la deuxième partie de nos expériences qui correspond à la partie 3 de cette étude, nous avons cherché à reconnaître de la parole continue en utilisant des connaissances de plus haut niveau, un lexique et une syntaxe.

Des méthodes de classification permettent d'apprendre et ensuite de reconnaître des unités élémentaires. Les phénomènes d'allongement et de compression temporelle dus au débit du locuteur et à la coarticulation entre les sons sont en général pris en compte par des algorithmes d'alignement temporel. Ils permettent de tolérer une certaine variabilité du signal et d'intégrer les séquences de classes reconnues en mots puis en phrases.

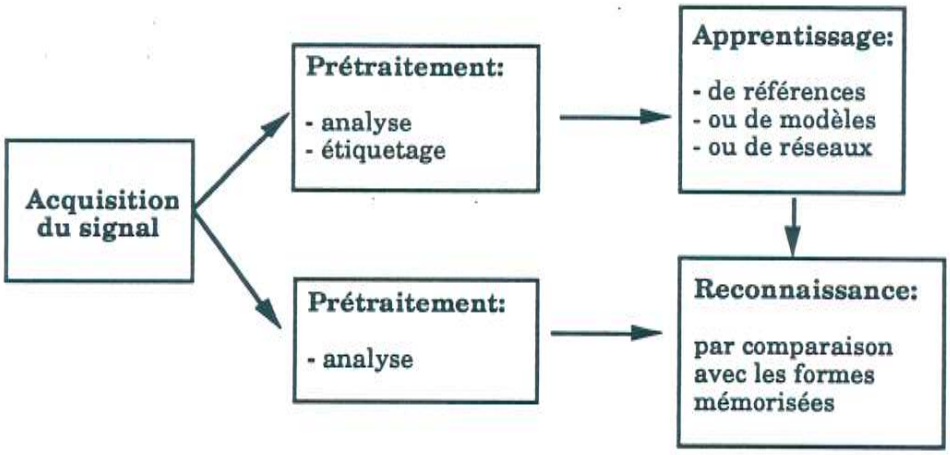

*Fig.. 2.1 : **Système de reconnaissance de formes acoustiques.***

On distingue quatre modules dans un système de reconnaissance de la parole, cf fig.2.1 :

- l'acquisition du signal qui correspond à l'extraction de données numériques. L'échantillonnage du signal permet de passer des données analogiques aux données numériques ;

- les prétraitements qui permettent de réduire la quantité d'information du signal de parole et de faciliter la classification en mettant en évidence des invariants par rapport aux événements acoustiques ;

- l'apprentissage où l'on utilise une méthode de classification pour mémoriser les formes (ce module n'est pas présent dans tous les systèmes). Les "formes" sont représentées par des séquences de vecteurs correspondant à des segments phonétiques ;

- la reconnaissance qui correspond à la phase d'identification de formes inconnues par comparaison avec les connaissances mémorisées par une méthode de classification ou avec les données initiales sans phase d'apprentissage. Un algorithme d'alignement temporel peut aussi être utilisé afin de gérer la dynamique temporelle.

Nous présentons les principales difficultés de la reconnaissance de la parole au paragraphe 2.1, et les différents modules de traitement dans la suite de ce chapitre. L'acquisition du signal et les prétraitements classiquement utilisés en parole sont introduits dans les paragraphes 2.2 et 2.3. Les méthodes générales de classification des formes sont présentées dans le paragraphe apprentissage et reconnaissance, §2.4. Les approches classiques de reconnaissance des formes ainsi que les méthodes

connexionnistes explorées dans nos expériences y sont décrites. Nous détaillerons ensuite dans le paragraphe 2.5 les systèmes à base de modèles markoviens qui sont les méthodes les plus puissantes, à l'heure actuelle, en reconnaissance de la parole. Ces modèles ont servi de référence dans cette étude. Enfin, le paragraphe 2.6 est consacré à la reconnaissance par comparaison dynamique. Les méthodes de classification abordées dans le paragraphe 2.4 n'ont pas, à l'exception des modèles markoviens, de mécanisme d'alignement temporel. Nous présentons les deux algorithmes de comparaison dynamique les plus classiquement utilisés en reconnaissance de la parole, algorithmmes qui sont en fait très proches : le *"Dynamic Time Warping"* ou DTW et l'algorithme de Viterbi utilisé dans les HMMs.

Deux ouvrages sont en général cités comme référence pour le traitement de la parole [Liénard 77], [Calliope 89].

2.1 Principales difficultés en reconnaissance de la parole

La complexité de la reconnaissance phonétique de la parole est due en grande partie à l'importante variabilité fréquentielle et temporelle du signal de parole, intra et inter locuteur. Chaque son élémentaire est déformé par les sons qui l'entourent par un effet de coarticulation. De plus, le débit du locuteur, son accent, son état émotionnel, physique entraînent des modifications du signal de parole. Le signal est aussi déformé par le bruit environant ou la superposition de voix (effet cocktail). Il est important de rappeler que la parole naturelle spontanée comprend des hésitations, des ajouts de mots liaisons, une sysntaxe souvent peu correcte et est encore plus difficile à décoder que la dictée de textes. La complexité de la reconnaissance de la parole est donc très dépendante de l'application. Elle est fonction :

- du mode d'élocution : mots isolés ou parole continue,
- de la taille du dictionnaire : de 10 à plus de 10000 mots de vocabulaire,
- du langage : naturel ou artificiel,
- de l'environnement : bruité ou non,
- de la dépendance au locuteur : mono ou multi-locuteurs.

Ces problèmes imposent différentes techniques. Un système de reconnaissance sera toujours orienté par la tâche à résoudre.

L'élocution, tout à fait artificielle, en mots isolés, diminue la difficulté de la reconnaissance de la parole. La parole est alors mieux articulée, le débit de parole et les phénomènes de coarticulation sont réduits et les pauses permettent de séparer facilement les mots. Par contre, la reconnaissance de parole continue présente toutes les difficultés, de débit, de coarticulation... Les mots ne sont pas séparés par des silences, ce qui affecte les phonèmes et les mots courts. Les applications en parole continue comptent en général un vocabulaire plus vaste que des applications en mots isolés et nécessitent l'utilisation d'unités élémentaires de taille inférieure au mot. On parle alors de reconnaissance analytique en opposition à la reconnaissance globale où l'on cherche à reconnaître directement les mots.

L'intégration des unités en mots puis en phrases oblige à des analyses de plus haut niveau, à un accès lexical puis à une analyse syntaxique.

Les bases de données de parole utilisées dans cette étude sont dépendantes du locuteur, cf annexe 1 et 2. Elles ont été enregistrées dans des conditions de laboratoire sans bruit additionnel. La principale difficulté de ces applicationss réside en la reconnaissance de parole continue. De plus la reconnaissance d'une base de données telle que Darpa, cf annexe 2, oblige à gérer une base de données de taille relativement importante.

L'unité choisie dans notre application est le phone qui est la plus petite entité acoustique permettant de différencier deux mots. Le choix d'unités inférieures au mot pour gérer de grandes bases de données est un choix pratique; le nombre de phonèmes est petit (de l'ordre de 50) par rapport au nombre de mots (souvent plus de 1000) et des modèles phonétiques peuvent être entraînés sur un corpus plus petit que celui nécessaire pour apprendre des modèles de mots. Ce n'est certes pas l'unité de traitement idéale puisque le phonème est très sensible aux phénomènes de coarticulation. On s'oriente de plus en plus vers des systèmes qui utilisent des combinaisons d'unités : des points d'ancrage phonémiques et des diphones [Adda-Decker 88], des phonèmes en contexte dans le système Byblos BBN [Chow 87], des phonèmes dépendants (triphones) et indépendants du contexte dans le système Sphinx CMU [Lee 90].

2.2 Acquisition du signal de parole

La parole est un phénomène vibratoire qui implique un milieu de propagation. La propagation dans l'air dépend de la forme du conduit vocal et du fonctionnement de la source d'excitation, ainsi que des caractéristiques du milieu de propagation. Ce signal vocal est fortement perturbé par tous les bruits ambiants qui se superposent au signal émis. Le signal de parole est en général échantillonné à 10 ou 16 kHz afin de conserver l'information spectrale jusqu'à 5 ou 8 kHz [Liénard 77], [Calliope 89].

Les bases de données que nous avons utilisées ont été enregistrées dans des conditions de laboratoire, en milieu non bruité. Elles sont constituées par des textes lus.

2.3 Prétraitements

Le but des prétraitements acoustiques pour la reconnaissance de la parole est de réduire la quantité d'information du signal de parole et de faciliter la classification en mettant en évidence des invariants par rapport aux événements acoustiques.

2.3.1 Analyse du signal

La parole peut se définir par son amplitude dans les dimensions de temps et de fréquence : le traitement du signal a pour but d'extraire ces trois paramètres du signal physique. Le signal vocal couvre pratiquement toute l'étendue du spectre audible mais on peut se limiter à une étude sur la bande de fréquences de 50 à 5000 Hz (par exemple la bande passante du téléphone va de 300 à 3400 Hz).

L'analyse spectrale et le codage prédictif linéaire (LPC) sont les deux familles de techniques les plus utilisées en reconnaissance de la parole.

Nous avons utilisé, dans cette étude, une analyse de Fourier discrète à court terme (avec un algorithme de FFT *"Fast Fourier Transform"*) et différentes paramétrisations dont le calcul des cepstres. Des comparaisons entre ces différentes méthodes et d'autres analyses temps-fréquence et paramétrisations sont répertoriées dans le rapport "Analyses temps-fréquence" [D'Alessandro &

Demars 92]. Pour plus de précisions sur les méthodes d'analyse du signal, on peut se référer au livre de "Traitement numérique du signal" [Bellanger 81].

Les informations généralement extraites ne tiennent pas explicitement compte de la prosodie, ni des informations diagnostiques comme l'âge du locuteur, son sexe, ses origines. Par ailleurs, les paramètres retenus ne sont pas particulièrement robustes au bruit, ni invariants par rapport au locuteur. Des recherches sont actuellement menées sur l'amélioration du choix de ces paramètres et l'ajout d'indices diagnostiques [Liénard 92].

L'analyse de Fourier

Les relations entre un signal s(t), avec $-\infty < t < +\infty$, et son spectre harmonique S(ω), avec $-\infty < \omega < +\infty$ sont définies par la tranformée de Fourier :

$$S(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} s(t) e^{-j\omega t} dt$$

La transformée de Fourier discrète

La transformée de Fourier s'applique à des signaux continus, pour les signaux numérisés, on utilise la transformée de Fourier discrète, notée DFT. Le signal s(t) devient s(nT) le signal échantillonné de période T. S(kΩ) est la transformée du signal en fréquences avec $\Omega = \frac{2\pi}{NT}$, N étant le nombre d'échantillons considérés :

$$S(k\Omega) = \sum_{n=0}^{N-1} s(nT) e^{-j\Omega Tnk} \text{ où } k = 0, 1, ...,N$$

Le calcul de la DFT comporte uniquement des additions et des multiplications par les scalaires complexes $e^{-j\Omega Tnk}$ en N^2 opérations, (N fois S(kΩ) et chaque S(kΩ) comporte N termes à k constant). La FFT est un algorithme de calcul de la DFT qui utilise la périodicité des scalaires complexes et permet ainsi d'effectuer un calcul plus rapide en $N\log_2 N$ opérations. La DFT est calculée avec un nombre fini d'échantillons, elle n'est ni bornée temporellement, ni fréquentiellement. Il faut donc définir la durée de la tranche de signal à analyser spectralement.

La transformée de Fourier discrète à court terme

Le signal de parole, en général non stationnaire, a une structure très variée: périodique, apériodique, impulsionnelle ou bruitée. Cependant, si on considère des

tranches de signal suffisamment fines (d'environ 30 ms), le signal est quasi-stationnaire. On parle alors de transformée de Fourier à court terme, notée S_t, w(t) représentant une fenêtre temporelle :

$$S_t(\omega) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} s(t)w(t)e^{-j\omega t}dt = \frac{1}{\sqrt{2\pi}}\int_{-T}^{+T} s(t)w(t)e^{-j\omega t}dt$$

Le fait de borner l'intervalle d'analyse temporelle introduit dans l'analyse fréquentielle des distorsions qu'il faut limiter. La transformée de Fourier d'un produit est le produit de convolution des transformées. La transformée d'une fenêtre de signal a pour conséquence de générer des lobes secondaires qui introduisent des distorsions. Le choix de la fenêtre temporelle permet de limiter ces distorsions. La fenêtre la plus courante est celle de Hamming :

$$W(n) = 0.54 - 0.46\cos(\frac{2\pi n}{N-1}) \text{ pour } n = 0,..,N$$

L'analyse de Fourier donne une représentation à 2 dimensions (si on néglige l'information de phase) : fréquence et amplitude. C'est la répétition d'une analyse de Fourier à court terme suivant un certain pas temporel qui donne une représentation en trois dimensions. A chaque pas de translation de la fenêtre d'analyse sur le signal, un spectre est calculé (l'ordre de grandeur de ce pas est de 10 ms)..

Nous décrivons rapidement les paramètres et traitements que nous avons utilisés pour la base de données DARPA, cf annexe 2. Pour les tests sur la base de données en français, l'analyse est très peu différente, elle est décrite en annexe 1. Une préaccentuation du signal à 6dB par octave est employée pour augmenter l'amplitude des hautes fréquences. Une FFT est appliquée sur un segment de signal de l'ordre de 30ms, pondéré par une fenêtre de Hamming pour éliminer les effets de contributions parasites introduites par les discontinuités du signal aux bornes de la fenêtre d'analyse. Le spectre d'amplitude obtenu est représenté dans la bande de fréquence de 0 à 8kHz pour un échantillonnage à 16kHz. Le spectre est ensuite passé dans un banc de 16 filtres triangulaires non symétriques répartis suivant une échelle de Bark représentant la sensibilité de l'oreille en fréquences. Pour avoir des spectres normalisés en amplitude, une fonction de codage quasi-logarithmique est appliquée sur chaque canal.

Le codage cepstral

Il est défini comme la transformée de Fourier inverse du logarithme de la transformée de Fourier du signal. On considère que le spectre est le résultat de la convolution du spectre d'excitation (vibration des cordes vocales) et de la fonction de transfert du conduit vocal (résonance dans le conduit vocal modifié par les articulateurs). On peut séparer ces deux informations en déconvoluant le signal. Le calcul du cepstre nécessite deux transformées de Fourier. La deuxième transformée de Fourier peut être remplaçée par la Transformée Cosinus Discrète.

2.3.2 Métrique

Les méthodes de classification nécessitent souvent de définir une métrique. Cette distance est une mesure de distorsion entre deux vecteurs. Les différences de distorsion sont suceptibles d'indiquer des différences au niveau perceptif. Plusieurs distances sont actuellement utilisées, qui sont liées à la nature des paramètres extraits du signal. Dans le cas d'une analyse spectrale ou cepstrale, les distances Lr sont les plus utilisées.

Soit x,y deux vecteurs de même dimension,

- la distance de Minkowski d'ordre 1 (L1),

$$d(x,y) = \frac{1}{N}\sum_{k=1}^{N} |x_k - y_k|$$

- la distance de Minkowski d'ordre 2 (erreur quadratique ou distance euclidienne),

$$d(x,y) = \frac{1}{N}\sum_{k=1}^{N} (x_k - y_k)^2$$

2.3.3 Quantification vectorielle

Le problème de la quantification vectorielle (QV) est un problème bien connu en analyse de données. : comment effectuer une classification de données en un nombre discret de catégories représentées par des vecteurs-références en perdant le moins possible d'information. Un tel codage simplifie l'algorithme de reconnaissance, il est utilisé comme prétraitement des données dans de nombreuses applications.

On distingue deux familles de quantifieurs : les méthodes en ligne et les méthodes globales. La méthode incrémentale est une méthode en ligne très simple dont l'avantage est de ne pas nécessiter la présence de toutes les données pendant l'apprentissage. Nous détaillerons plus précisément dans ce paragraphe les méthodes globales que nous avons utilisées dans cette étude ainsi qu'une méthode connexionniste de quantification. Ce sont des algorithmes itératifs, nécessitant la présence de tout le corpus d'apprentissage, qui cherchent à partitionner l'ensemble des vecteurs. Les différents algorithmes décrits plus bas sont des variantes de l'algorithme des nuées dynamiques, cf la thèse de [Bonneau 87]. La méthode connexionniste est décrite plus en détail dans le chapitre 3.

Algorithme des k-moyennes :

Pour cet algorithme [Pollard 82], on choisit un ensemble initial de k vecteurs-références. L'ensemble des vecteurs est classé par la règle du proche voisin (cf §2.4.3.2). Pour chaque sous-ensemble de vecteurs, on calcule le centroïde correspondant aux vecteurs d'apprentissage appartenant à cette classe de vecteurs. Si les variations des centroïdes n'évoluent plus, on arrête la procédure sinon on réitère l'algorithme.

Cet algorithme converge vers un optimum local en un temps plus ou moins long, dépendant du choix des k vecteurs initiaux. La solution obtenue avec ce type d'algorithme n'est donc pas unique. D'autre part, le choix du nombre de vecteurs-références idéal pour décrire la base de données est difficile à trouver.

Algorithme de "*split*" :

L'algorithme de "*split*" [Linch] est une variante de l'algorithme des k-moyennes. Le problème du choix du nombre de vecteurs-références et de l'initialisation des vecteurs est en partie résolu. L'algorithme commence avec un seul vecteur qui correspond au centroïde calculé sur toute la base de données. Ce vecteur est décomposé en deux vecteurs. On ajoute au vecteur initial 2 vecteurs perturbateurs eps et -eps pour former les deux nouveaux vecteurs. Le choix du vecteur perturbateur eps est ici le principal problème, on choisit souvent de ne perturber que la composante qui a le maximum de variance dans la base de données.

Algorithme des boules optimisées :

Le seul paramètre à déterminer est le rayon des classes. L'algorithme des boules optimisées [Flamenbaum] oblige les classes à avoir un rayon inférieur à un rayon R qui est un seuil fixé a priori. Un vecteur de la base de données n'est affecté à une

classe que si sa distance est inférieure au seuil, sinon une nouvelle classe est créée. Les vecteurs-références correspondent au centre de chacune de ces "boules". Cette méthode permet de ne pas choisir a priori un nombre de classes de référence et une initialisation a priori des vecteurs.

Méthode connexionniste des "*Features Map*":

La méthode des "Features Map" ou FM, appelée en français cartes topologiques, a été introduite par [Kohonen 84]. Ces réseaux font une quantification des vecteurs de données en gardant une représentation qui respecte leurs caractéristiques de voisinage, c'est à dire une topologie de l'espace d'entrée. On donne un nombre de vecteurs-références a priori qui sont initialisés de façon aléatoire. Au cours de l'apprentissage, l'espace de représentation se partitionne sans supervision. Structure et méthode d'apprentissage sont décrites en détail dans le chapitre 3.

2.3.4 Etiquetage et segmentation du signal

La plupart des algorithmes d'apprentissage utilisés par les réseaux de neurones nécessitent une segmentation des données d'apprentissage, d'autres n'ont besoin que d'un étiquetage des formes et n'ont pas besoin des bornes précises des segments. On peut se servir d'un spectrogramme, c'est à dire d'une visualisation des séquences de spectres en trois dimensions: temps, fréquence, énergie, pour segmenter et étiqueter manuellement le signal. Il existe des outils permettant d'écouter des tranches de signal et de valider les étiquettes et les bornes choisies. Cette façon de procéder est certainement la plus fiable si on respecte les mêmes normes du début à la fin de l'étiquetage, mais c'est aussi la plus fastidieuse.

Il existe des programmes de détection de frontières phonétiques fondés sur des variations d'énergie qui correspondent aux contrastes voyelles-consonnes de la langue [Andreewsky 87], [Devillers 87]. Cependant, ces programmes utilisent de nombreux seuils et règles heuristiques pour étiqueter les phonèmes et nécessitent un contrôle manuel. La segmentation automatique avec des modèles markoviens est la plus souvent employée. Les modèles de Markov ont besoin de peu de données segmentées pour initialiser leur apprentissage.

2.4 Apprentissage et reconnaissance

On distingue en général deux phases dans un système de reconnaissance des formes :

L'apprentissage où l'on construit un modèle ou une référence d'une forme à partir de plusieurs occurrences de cette forme.

La reconnaissance où on déclare reconnue la forme la plus probable ou la plus proche de celle présentée au sens d'une distance.

Pour l'apprentissage et la reconnaissance, on utilise souvent des méthodes de classification. Nous avons choisi de présenter une vue synthétique d'un certain nombre de méthodes classification en reconnaissance des formes.

Deux facteurs déterminent le succès d'une approche: la structure du classifieur et le critère d'apprentissage. Typiquement, on a une connaissance globale du problème à traiter et un ensemble de données qui sont des exemples des formes que l'on veut classifier. La forme que l'on donnera au classifieur dépend des connaissances a priori du problème. L'apprentissage permet d'extraire une certaine connaissance des exemples présentés afin de pouvoir reconnaître ensuite des données inconnues.

L'ensemble des données d'apprentissage est fini et reflète de manière incomplète tous les cas qui peuvent apparaître. Plus on représente ces données de manière précise, moins on apprend à généraliser à d'autres données. Il est évident que plus la base de données est importante moins l'apprentissage est sensible à ce phénomène. La difficulté de l'apprentissage est donc double : apprendre à classifier tout en gardant une "marge de flou" permettant de généraliser.

Nous essayons dans ce paragraphe de situer quelques méthodes de classification connexionnistes par rapport aux méthodes classiquement utilisées en reconnaissance statistique des formes. Les critères utilisés avec les différentes structures neuronales sont interprétés dans un formalisme probabiliste. Les critères d'apprentissages décrits sont tous supervisés, c'est-à-dire qu'ils utilisent la connaissance des classes des formes apprises.

Le paragraphe suivant décrit plus en détail les modèles markoviens et la reconnaissance par comparaison dynamique. Les modèles connexionnistes étant l'objet principal de cette étude, les algorithmes d'apprentissage et les structures utilisés sont décrits dans le chapitre 3.

2.4.1 Classification des formes et théorie de la décision bayésienne

Il existe de nombreuses façons de définir une méthode de classification. Soit N observations de $X=\{x_1,x_2,x_3...,x_N\}$, C l'ensemble des classes $(C_1, ...C_n)$, n étant le nombre de classes, l'ensemble d'apprentissage est formé de N couples (x_i,C_j).

- L'une d'elles est de considérer un classifieur comme un système qui construit n fonctions discriminantes $f_i(x)$ avec i=1,...,n et sélectionne la classe C_i pour x tel que $f_i(x) > f_j(x)$ pour tout $j \neq i$,

- Un classifieur peut aussi se définir comme un moyen de partitionner l'espace X en n sous-ensembles A_i [1,..,n] auxquels sont associés une classe C_i,

- ou comme un estimateur des densités de probabilités des classes.

La théorie bayésienne est une approche statistique fondamentale du problème de classification des formes. Elle a été appliquée à la reconnaissance des formes par [Chow 70]. Cette approche est fondée sur l'hypothèse que le problème de décision est posé en termes probabilistes.

Soit N observations de $X=\{x_1,x_2,x_3...,x_N\}$ réparties dans un ensemble de classes C=(C1,...Cn), soit P(Ci/x) la probabilité a posteriori, c'est à dire la probabilité de la classe Ci si l'observation est x, p(x/Ci) la probabilité conditionnelle de x pour la classe Ci, P(Ci) la probabilité a priori de la classe Ci ; p(x) correspond à la densité de probabilité de x.

La règle de Bayes s'écrit :

$$P(C_i/x) = \frac{p(x/C_i)\,P(C_i)}{p(x)}$$

avec

$$p(x) = \sum_{i=1}^{n} p(x/C_i)P(C_i)$$

La règle de Bayes montre comment l'observation de la valeur x change la probabilité a priori d'une forme $P(C_i)$ en une probabilité a posteriori $P(C_i/x)$.

La notion de risque fait partie de la terminologie de la théorie de la décision. Le risque correspond à l'erreur a priori. Soit $R(C_i/x)$ le risque conditionel, c'est-à-dire

le risque que C_i ne soit pas la bonne classe ; soit $\lambda(C_i/C_j)$ le coût de mauvaise classification décrit ci-dessous, on peut écrire :

$$R(C_i/x) = \sum_{j=1}^{n} \lambda(C_i/C_j)P(C_j/x)$$

avec le coût de mauvaise classification (toute erreur vaut 1) :

$$\lambda(C_i,C_j) = \begin{cases} 0 \text{ si } i=j \\ 1 \text{ si } i\neq j \end{cases} \text{ avec } 1< i,j < n$$

Le risque R relatif à Ci s'écrit :

$$R = \int_x R(C_i/x)\ p(x)\ dx$$

Pour minimiser ce risque, il suffit de minimiser le risque conditionnel :

$$\begin{aligned} R(C_i/x) &= \sum_{j=1}^{n} \lambda(C_i/C_j)P(C_j/x) \\ &= \sum_{j\neq i} P(C_j/x) \\ &= 1 - P(C_i/x) \end{aligned}$$

Pour minimiser le risque conditionnel, il faut trouver l'indice i qui maximise $P(C_i/x)$. La règle de décision bayésienne donne le nombre minimal d'erreurs de décision. Pour une forme inconnue x, trouver la classe de x, notée C_i correspond à :

$$C_i = \text{argmax}_n(P(C_n/x)$$

c'est à dire aussi $C_i = \text{argmax}_n(P(C_n)*p(x/C_n))$

La règle de décision bayésienne permet de calculer n fonctions discriminantes. La fonction donnant le maximum de discrimination correspond à la probabilité a posteriori maximal. En d'autres termes, les fonctions de probabilités des classes $P(C_i/x)$ décrivent les frontières des classes dans l'espace des données.

Pour les méthodes de classification probabilistes, le critère de décision bayésien permet d'estimer lors de la reconnaissance, les classes des phonèmes qui minimisent le nombre d'erreurs de classification.

Le classifieur bayésien consiste à calculer la probabilité a posteriori de x pour chaque classe Ci. Sur des données dont on connait la distribution, on peut calculer

exactement p(x/Ci) et P(Ci) et obtenir les performances du classifieur bayésien qui correspond à la classification optimale sur ces données.

Dans des applications de reconnaissance des formes, on a rarement la connaissance complète de la structure probabiliste du problème, c'est en fait une approximation de ce classifieur qui est utilisée.

2.4.2 Méthodes de classification classiques et connexionnistes

On a tendance à présenter les méthodes dites "classiques" pour la reconnaissance des formes indépendamment des méthodes connexionnistes. Nous les avons regroupées en méthodes paramétriques et non paramétriques [Duda 73].

Les méthodes non paramétriques cherchent à estimer directement les fonctions $P(x/C_i)$ sans hypothèse sur la forme des densités. Ces méthodes sont en fait des méthodes de quantification vectorielle tenant compte des classes C_i. Les procédures de décision les plus connues sont les kNN pour "*k Nearest Neighboor*" et les fenêtres de Parzen, elles sont décrites dans le paragraphe cf 2.4.3. Les méthodes de quantification vectorielle décrites dans le § 2.3.3 sont généralement utilisées pour réduire la taille de l'espace de données et simplifier la classification.

Les méthodes paramétriques font l'hypothèse que les formes des fonctions de densités sont connues et cherchent à estimer les paramètres de ces densités. Le problème d'estimation de paramètres est bien connu en statistique, deux procédures sont classiquement utilisées: le critère de MLE, "*Maximum Likelihood Estimation*" et le critère bayésien ou MAP pour "*Maximum A Posteriori*". Plusieurs autres critères d'apprentissage ont été proposés, par exemple le MMI pour "*Maximum Mutual Information*" par rapport au MLE, permet une optimisation globale des classes. Ils sont décrits dans le paragraphe §2.4.4.

Les réseaux de neurones de type perceptrons multi-couches utilisés avec le critère de MSE "*Mean Squared Error*" ou d'autres critères similaires ne peuvent être insérés dans ces catégories. En effet, ils ne font pas d'hypothèse directement sur la forme des fonctions de densités mais sur la structure du réseau dont les paramètres permettent d'évaluer implicitement les fonctions de discrimination [Duda et al 73].. Duda classe d'ailleurs le perceptron dans les méthodes à base de fonctions discriminantes. Une interprétation probabiliste de l'apprentissage des réseaux de neurones avec un critère de discrimination tel que le critère de MSE et de "*cross-entropy*", cf §3.6.2, sera donnée.

Les réseaux de neurones avec apprentissage LVQ *"Learning Vector Quantisation"* utilisent une règle de décision de type plus proche voisin pour estimer les classes. Les paramètres d'un réseau LVQ servent à estimer les densités de probabilité.

Nous présenterons ces deux approches sous la catégorie "méthodes paramétriques connexionnistes".

Les tableaux suivants Tab. 2.1, Tab. 2.2 et Tab. 2.3 regroupent quelques exemples de méthodes de classification paramétriques, non paramétriques et connexionnistes. Chacune d'elle est décrite par plusieurs champs: la structure, le critère de décision, le critère d'apprentissage, les algorithmes d'apprentissage et le but de la classification.

Cette description n'a pas pour but d'être exhaustive, nous ne présentons que les modèles les plus utilisés.

Structures	*Critère de décision*	*Critère d'apprentissage*	*Algorithmes d'apprentissage*	*But*
Ensemble des données ou QV	Fenêtres de Parzen	néant	néant	Approximation de P(x/C) d'où on déduit P(C/x
Ensemble des données ou QV	Règle des k plus proches voisins (kNN)	néant	néant	Approximation de P(x/C) d'où on déduit P(C/x

Tab. 2.1 : ***Méthodes de classification non paramétriques***

Structures	*Critère de décision*	*Critère d'apprentissage*	*Algorithmes d'apprentissage*	*But*
HMM	MAP argmax(P(C/x))	MLE: "*Maximum Likelihood Estimation*"	EM "*Expectation Maximization*": "*Forward-Backward*", Baum-Welch, Viterbi	Modelisation des $p(x/\lambda)$
HMM	MAP argmax(P(C/x))	MAP: "*Maximum A Posteriori*"	Version bayésienne, de EM	Modelisation des $p(\lambda)p(x/\lambda)$
HMM	MAP argmax(P(C/x))	MMI: "*Maximum Mutual Information*"	Version dérivée des algorithmes des NNs: Descente de gradient	Modelisation des $P(\lambda/x)$

Tab. 2.2 : ***Méthodes de classification paramétriques.*** Les λ représentent les modèles.

Structures	*Critère de décision*	*Critère d'apprentissage*	*Algorithmes d'apprentissage*	*But*
QV	Règle de type 1NN	Minimisation du taux d'erreur aux frontières	LVQ, LVQ2	Approximation de P(C/x)
MLP TDNN	Soit Y vecteur de sortie de dimension n argmax(Y)	MSE: "*Mean Squared Error*" ou "*Cross-entropy*"	Rétroprogation du gradient d'erreur	Projection sur un autre espace, approximation des P(C/x)

Tab. 2.3 : ***Méthodes de classification connexionnistes.*** *Ce sont des méthodes paramétriques sans hypothèse sur la forme des densités.*

Il existe deux types d'apprentissage : par maximisation d'une vraisemblance ou par minimisation d'une fonction liée à l'erreur de classification . Le premier type d'apprentissage n'utilise pas un critère discriminant. Résumons cette information dans un tableau dont les différents champs seront le critère d'apprentissage et le type d'algorithme: MAX pour les algorithmes EM *"Expectation Maximisation"* ou MIN pour indiquer le concept de minimisation d'erreur.

Critère d'apprentissage	*Algorithmes*
MLE	MAX
MAP	MAX
MSE, cross-entropy	MIN
MMI	MIN
LVQ	MIN

*Tab. 2.4 : **Apprentissage discriminant (MIN) et non discriminant (EM)***

Une autre information peut être ajoutée concernant les méthodes de classification, c'est celle du coût de réalisation. Il est en effet intéressant de classifier ces différentes approches en fonction des hypothèses effectuées, du temps d'apprentissage et des données nécessaires pour optimiser les méthodes. Ce tableau est volontairement simplifié, on considera les trois classes génériques: les méthodes paramétriques (HMMs), les méthodes paramétriques connexionnistes (NNs) et les méthodes non paramétriques (kNNs/Parzen). La classe HMM, cf §2.5, est séparée en deux catégories; HMM(1) correspond à une version de base des HMMs, on utilise une modélisation simple d'une densité unimodale par état, HMM(2) correspond à une modélisation plus complexe où on associe à chaque état une densité représentée par une somme de distributions normales unimodales qui est une approximation d'une densité multimodale. Les hypothèses restrictives du HMM(1) sont diminuées par l'utilisation de somme de distributions normales dans l'approche HMM(2) et le nombre de paramètres à évaluer est beaucoup plus important. L'ordre donné dans le tableau entre N2 et N3, T2 et T3, B2 et B3 peut être discuté. Il est donné a priori et dépend du nombre de paramètres des HMMs, on a fait l'hypothèse dans ce tableau que ce nombre était très grand. Une autre hypothèse a priori a été faite: la taille du réseau est grande et le temps d'apprentissage nécessaire T3 est supérieur à T2, le temps d'apprentissage des HMMs avec un mélange de distributions normales.

Méthodes	*degré (max D4) et type d'hypothèses D1<D2<D3<D4*		*Nombre de paramètres N1<N2<N3*	*Temps d'apprentissage T1<T2<T3<T4*	*Besoin en données B1<B2<B3<B4*
HMM(1)	D4	Densités normales unimodales	N1	T1	B1
HMM(2)	D3	Mélanges de densités	N3	T2	B3
NN	D2	Structure du réseau	N2	T3	B2
kNN/Parzen	D1	1 hypothèse sur k ou sur le volume	0	T4	B4

Tab. 2.5 : ***Coût des méthodes de classification*** *en fonction des hypothèses sur les formes (Di correspond au degré des hypothèses, plus Di est grand plus les hypothèses sont restrictives), en fonction du nombre de paramètres à évaluer (Ni), en fonction du temps d'apprentissage (Ti) et en fonction de la taille de la base de données nécessaires pour optimiser l'approche (Bi).*

Les méthodes non paramétriques sont les plus coûteuses en temps de calcul et en données, ce sont aussi les plus générales. En pratique ces méthodes sont souvent utilisées après avoir simplifié l'espace des données par des méthodes de quantification vectorielle.

2.4.3 Méthodes non paramétriques

Pour la plupart des applications de reconnaissance des formes, l'hypothèse faite sur la forme des fonctions de densité est restrictive. En particulier, les fonctions de densités sont rarement unimodales. Il existe des méthodes dites non paramétriques qui s'affranchissent de ce problème. Elles essayent d'approximer directement les probabilités a posteriori, c'est le cas du classifieur des k plus proches voisins appelé kNN, ou d'estimer les probabilités conditionnelles, c'est le cas des estimateurs kNN et des fenêtres de Parzen. Nous introduisons dans ce paragraphe les principes de base de ces techniques.

2.4.3.1 Estimateur de probabilités : kNN et Fenêtres de Parzen

L'estimateur de probabilités est fondé sur une remarque simple: deux échantillons proches au sens d'une distance ont une grande probabilité d'appartenir à la même classe.

Soit x un exemple dans l'espace des données, on choisit une région R autour de cet exemple. Soient N observations de $X=\{x_1,x_2,x_3...,x_N\}$, soit la probabilité P qu'un exemple x tombe dans une région R, x' les éléments de R, V le volume de la région R, en supposant que p(x) est continu et qu'il ne varie pas beaucoup dans la région R, on peut écrire :

$$P = \int_R p(x')dx' \approx p(x)V \qquad (1)$$

D'autre part, soient n exemples $x_1, ..., x_n,$, soient k éléments parmi ces n tombant dans la région R. E(k) est égale à n fois P, P étant la probabilité qu'un x tombe dans la région R. On a une bonne approximation de P :

$$P \approx \frac{k}{n} \quad (2)$$

d'où d'après (1) et (2) une approximation de la densité x :

$$p(x) \approx \frac{k}{nV} \quad (3)$$

De la même façon, on estime les densités de probabilités conditionnelles $p(x/C_i)$ et les probabilités a posteriori $P(C_i/x)$ par la règle de Bayes.

Supposons que le volume V capture k échantillons dont k_i sont d'étiquette C_i, la probabilité p(x/Ci) se calcule comme :

$$p(x/C_i) \approx \frac{k_i}{nV} \quad (4)$$

D'où une estimation de $P(C_i/x)$:

$$P(C_i/x) \approx \frac{p(x/C_i)}{\sum_{j=1}^{c} p(x/C_j)} = \frac{k_i}{k}$$

L'estimation de la probabilité a posteriori P(Ci/x) est très simple, c'est la fraction du nombre d'échantillons de classe Ci sur le nombre total d'échantillons contenus dans le volume V. Pour minimiser l'erreur, on sélectionne la classe la plus représentée.

Cette méthode est optimale si le volume V est assez petit et le nombre d'exemples suffisant.

Soit n le nombre d'échantillons, $p_n(x/C_i) = \frac{k_n}{nV_n}$ (5)

On peut montrer que $p_n(x/C_i)$ converge vers $p(x/C_i)$, si trois conditions sont respectées :

$\lim_{n->\infty} V_n = 0$ pour que l'approximation (1) soit vraie,

$\lim_{n->\infty} k_n = \infty$ pour que l'approximation (2) soit vraie,

$\lim_{n->\infty} k_n/n = 0$ pour que l'approximation (5) soit vraie,

Deux méthodes sont connues pour estimer ces paramètres, les fenêtres de Parzen et les k plus proches voisins [Duda 73], [Devijver 82]. Ces méthodes sont à la fois puissantes et limitées. Leur puissance réside dans leur généralité. La même procédure est utilisée pour estimer des densités unimodales ou multimodales. Avec suffisamment d'exemples et les conditions citées plus haut, la convergence vers les densités de probabilité inconnues est assurée. Cependant, il faut beaucoup plus d'exemples et de temps pour estimer une densité sans hypothèse sur sa forme. De plus, le nombre d'exemples nécessaires croit de façon exponentielle avec la dimension de l'espace des formes. En pratique, ces méthodes doivent avoir un bon comportement pour être optimales, elles sont aussi peu utilisables directement sur l'espace des formes. Des algorithmes de quantification vectorielle ou de projection, cf §2.3.3, permettent de réduire la complexité de l'espace de données.

Les fenêtres de Parzen.

Soit n échantillons, il y a deux inconnues dans l'équation (4) k_n et V_n. On fixe pour cette méthode le volume V_n et on calcule les k_n échantillons contenus dans V_n.

Le choix, par exemple, de $V_n = \frac{V_1}{\sqrt{n}}$ respecte la condition précédemment décrite qui est que $V_n -> 0$, lorsque $n -> \infty$, il faut ensuite montrer que les autres conditions sont respectées pour que $p_n(x/C_i)$ converge vers $p(x/C_i)$, c'est-à-dire que lorsque $n->\infty$, $k_n -> \infty$ et $k_n/n -> \infty$.

Quel que soit l'ensemble fini de n éléments, cette méthode est très sensible au choix du volume initial V_1. Si V_1 est trop petit, la plupart des volumes seront vides. Si V_1 est trop grand, d'importantes variations dans p(x) seront perdues. De plus V_1 pourra être bien adapté pour représenter une densité et non pour les autres.

Le critère de décision ou classifieur f(x,w) est égal à f(x) car cette méthode est non paramétrique, il s'écrit :

$$f(x) = C_i \text{ tel que } P(C_i/x) = \frac{\text{Max } p(x/C_i)}{\sum_{j=1}^{c} p(x/C_j)} = \frac{\text{Max}(k_i)}{k}$$

" *k Nearest Neighbor*": kNN

Soit n échantillons, il y a deux inconnues dans l'équation (4) k_n et V_n. On fixe pour cette méthode le volume k_n et on en déduit le volume V_n.

Le choix, par exemple, de $k_n = \sqrt{n}$ respecte certaines conditions pour que $p_n(x/C_i)$ converge vers $p(x/C_i)$ lorsque n-> ∞. Pour cette méthode, le volume Vn est calculé en fonction des données et non arbitrairement en fonction du nombre de données comme dans la méthode des fenêtres de Parzen. Si $k_n = \sqrt{n}$ alors $V_n \approx \frac{1}{\sqrt{n}\, p(x)}$, on vérifie donc que lorsque n-> ∞ V_n-> 0.

Le critère de décision f(x) s'écrit :

$$f(x) = C_i \text{ tel que } P(C_i/x) = \frac{\text{Max } p(x/C_i)}{\sum_{j=1}^{c} p(x/C_j)} = \frac{\text{Max}(k_i)}{k}$$

Cette méthode permet d'obtenir de bons résultats pour une distance judicieusement choisie. Mais en pratique, plus le nombre d'échantillons est grand et plus le calcul de la distance est complexe, plus cette technique est coûteuse et peu utilisable.

2.4.3.2 Classifieur kNN

Le classifieur kNN ne doit pas être confondu avec l'estimateur de probabilités kNN décrit précédemment. Avec un classifieur du type k plus proches voisins où k=1, "*Nearest Neighbor*" (1NN), un échantillon inconnu x reçoit la classe dont le représentant est le plus proche de x au sens d'une distance d. Ces représentants peuvent être des vecteurs quantifiés, cf §2.3.3.

Soit C_i la classe de la référence v_i, le critère de décision f(x) est alors simplement:

$$f(x) = C_i \text{ tel que } d(x,v_i) = \min_{c=1,..,V} d(x,v_c)$$

Ce classifieur donne des résultats satisfaisants mais on ne peut justifier de l'optimalité de la règle du plus proche voisin [Duda 73],[Devijver 82].

Les k plus proches voisins pour "*k Nearest Neighbor*" sont une amélioration de la règle du plus proche voisin ou k=1. On affecte à un échantillon la classe majoritairement représentée par les k plus proches voisins au sens de la distance choisie. Cette idée est issue des observations faites précédement sur les estimateurs de probabilité ; Soit un exemple x, si on fixe k et que l'on a un nombre infini d'exemples, les k plus proches voisins convergent vers x. En général, plus k est grand, plus la probabilité d'avoir la bonne classe est grande. Cependant, en pratique avec un nombre fini d'exemples, on veut que les "k plus proches voisins" x' soient réellement proches de x pour s'assurer que $P(C_i/x')$ soit égal à $P(C_i/x)$. Le choix de k est alors un compromis. Ce n'est que lorsque le nombre d'exemples tend vers l'infini que la règle des k plus proches voisins est optimale.

2.4.4 Méthodes paramétriques classiques

Le classifieur bayésien, §2.4.1, est un classifieur optimal si on connaît la probabilité a priori $P(C_i)$ et les densités de probabilités conditionnelles $p(C_i/x)$. Cependant, dans les applications de reconnaissance des formes, on a rarement la connaissance précise de la structure probabiliste du problème. Les approches paramétriques consistent à estimer les probabilités et densités de probabilités à partir des exemples en faisant des hypothèses sur la forme des fonctions de densité. L'estimation des paramètres de densités de probabilité est un problème classique en statistique. Nous présentons deux estimateurs très connus qui sont le critère de "*Maximum Likelihood Estimation*" et le critère bayésien de "*Maximum A Posteriori*".

2.4.4.1 Critère de MLE: "*Maximum Likelihood Estimation*"

Ce critère est sans doute le plus employé en reconnaissance des formes. C'est aussi une des méthodes qui utilisent le plus d'a priori sur la forme des données à classifier. Le nombre de paramètres à estimer dépend des hypothèses faites sur les densités de probabilités.

On estime les **probabilités conditionnelles** $p(x/\lambda)$, λ étant un modèle, avec le **critère de MLE** "***Maximum Likelihood Estimation***".

$$\lambda_i = \underset{\lambda}{\text{argmax}} \, (p(x/\lambda))$$

L'apprentissage consiste à estimer les modèles λ en fonction des données. Seuls les exemples d'apprentissage de classe C_i sont utilisés pour modifier les paramètres du modèle λ_i représentant la classe C_i. Le problème est traité séparément pour chaque classe. Le algorithmes d'apprentissage pour les HMMs sont du type EM "*Expectation Maximization*". Cette classe regroupe les algorithmes bien connus de Baum-Welch, forward-backward...

Les probabilités a priori $P(\lambda)$ peuvent être obtenues par différentes méthodes, la plus simple étant d'utiliser l'occurrence des classes dans la base de données. Le critère de décision bayésien est :

$$f(x,w) = C_i \text{ tel que } P(\lambda_i/x) = \underset{c=1,..,n}{\text{Max}} \, (p(x/\lambda_c))$$

La relation entre la maximisation des probabilités et le minimum d'erreur de classification n'est pas immédiate pourtant il existe des preuves théoriques de convergence des formules de réestimation des paramètres. Une classification optimale, c'est-à-dire un minimum d'erreur de reconnaissance, n'est cependant pas garantie.

2.4.4.2 Critère bayésien

Le **critère bayésien** correspond au **critère d'apprentissage MAP** "*Maximum A Posteriori*" qu'il ne faut pas confondre avec le critère de décision MAP utilisé lors de la reconnaissance. Pour une observation x, on veut trouver le modèle λi pour lequel la probabilité $p(x/\lambda i)P(\lambda i)$ est maximum.

le critère d'apprentissage bayésien MAP est noté :

$$\lambda_{max} = \underset{\lambda}{\text{argmax}} \, p(x/\lambda)p(\lambda)$$

Le but de l'apprentissage est d'évaluer les modèles λ. De la même façon que pour le critère MLE, ils sont estimés indépendamment les uns des autres.

La différence principale avec le critère de MLE concerne les modèles λ . Ils ne sont pas fixés comme dans les approches classiques des HMMs mais considérés comme

des variables aléatoires. Ceci a deux conséquences : la première est de lisser les modèles ce qui permet de mieux généraliser, la deuxième est d'avoir des modèles adaptatifs. Pour plus de détails sur le critère d'apprentissage bayésien utilisé en reconnaissance de la parole, on peut se référer aux travaux de [Gauvain 92]. Les algorithmes utilisés sont en général des variantes "bayésiennes" des algorithmes EM "*Expectation Maximization*": Baum-Welch, forward-backward...

Le critère de décision bayésien s'écrit:

$$f(x,w) = C_i \text{ tel que } P(\lambda_i/x) = \underset{c=1,..,n}{\text{Max}} (p(x/\lambda_c)p(\lambda_c))$$

Cette méthode est théoriquement optimale, elle nécessite cependant l'estimation de nombreux paramètres avec souvent peu de données ce qui engendre le risque d'obtenir de moins bons résultats qu'avec une méthode sous-optimale.

2.4.4.3 Critère de MMI: "*Maximum Mutual Information*"

Le critère MMI "*Maximum Mutual Information*" a été proposé comme une alternative du critère de MLE, permettant un apprentissage disciminant [Brown 87]. Il est en fait généralement utilisé comme un complément du MLE. La différence majeure entre MMI et MLE est que le critère MMI a comme objectif de maximiser la probabilité d'un modèle λ $p(x/\lambda)$ tout en minimisant la probabilité des autres modèles. L'utilisation du MMI après un apprentissage des HMMs avec le MLE [Niles 91], [Normandin 91] permet dans la majorité des cas d'améliorer les performances des HMMs. Il est cependant assez difficile à mettre en oeuvre. Les algorithmes MMI ou de "*corrective training*" qui sont souvent très proches, sont en général utilisé de façon plus ou moins ad hoc.

Ce critère cherche à maximiser la probabilité a posteriori :

$$\lambda_i = \underset{\lambda}{\text{argmax}} (P(\lambda/x) = \frac{p(x/\lambda)P(\lambda)}{\sum_m p(x/\lambda_m)P(\lambda_m)})$$

Le critère de décision bayésien s'écrit:

$$f(x,w) = C_i \text{ tel que } P(\lambda_i/x) = \underset{c=1,..,n}{\text{Max}} (p(x/\lambda_c)p(\lambda_c))$$

Par rapport au MLE, le critère MMI cherche à maximiser $P_\theta(x/\lambda)$ tout en minimisant $P_\theta(x/\lambda')$ pour les autres modèles λ', θ étant un vecteur de paramètres.

Le but est de trouver le vecteur θ qui maximise $I_\theta(\lambda/x)$:

$$I_\theta(\lambda/x) = \log P_\theta(x/\lambda) - \log P_\theta(\lambda)$$

en prenant le gradient de $I_\theta(\lambda/x)$ par rapport au paramètre θ_i, on obtient :

$$\frac{\partial I_\theta(\lambda/x)}{\partial \theta_i} = \frac{\dfrac{\partial P_\theta(x/\lambda)}{\partial \theta_i}}{P_\theta(x/\lambda)} - \frac{\displaystyle\sum_{\lambda'} \frac{P_\theta(x/\lambda')}{\partial \theta_i} P_\theta(\lambda')}{P_\theta(x)}$$

Le premier terme de l'équation correspond au gradient de la fonction de MLE, le deuxième terme correspond au terme ajouté par le critère de MMI. Ce terme sert à modifier les autres modèles λ'.

Avec le critère de décision de MAP, la relation entre le critère d'apprentissage et les erreurs de classification est plus évidente que dans le cas du MLE. Pourtant, on n'a pas de preuve théorique que les formules de réestimation des paramètres du MMI convergent. Un algorithme de descente de gradient est en général utilisé pour estimer les paramètres. On peut justifier empiriquement le critère MMI. Des expériences [Brown 87] [Nadas 83] ont prouvé que pour certains types de problème, le MMI converge vers un classifieur optimal même si les hypothèses de modélisation sont incorrectes alors que le critère de MLE ne converge pas. Ces expériences tendent à montrer que le MMI est plus robuste que le MLE pour des hypothèses de modélisation loin de la réalité. les résultats obtenus sont en général meilleurs qu'avec le MLE seul.

2.4.5 Méthodes paramétriques connexionnistes

2.4.5.1 Interprétation probabilitiste de l'apprentissage de réseaux de neurones utilisant un critère global discriminant

Nous voulons montrer qu'un critère d'apprentissage global discrimant, tel que par exemple le critère de MSE ou la *"cross-entropy"* , peut être interprété en termes probabilistes et revient à estimer directement **des probabilités a posteriori P(C/x).** Cette approximation a déjà été largement utilisée avec les réseaux de neurones MLP ou TDNN [Bourlard & Wellekens 89], [Bridle 89].

Soit N le nombre d'échantillons x, n le nombre de classes, Y un vecteur de sortie du réseau de dimension n dont les valeurs sont comprises dans [0,1]. Le critère de décision f(x,w) peut s'écrire:

$$f(x,w) = C_i \text{ tel que } Y_i = (\underset{c=1,..,n}{\text{Max}} (Y_c))$$

Soit les échantillons (x_t, k_t) avec t= 1,...,N, soit S le vecteur de sortie désirée, si on utilise le critère MSE, le coût C ou l'erreur moyenne s'écrit :

$$C = \frac{1}{N} \sum_{t=1}^{N} \sum_{i=1}^{n} (S_{t,i} - Y_{t,i})^2$$

La sortie du réseau est notée $Y_{t,i} = g(x_t,c)$, cette fonction g dépend bien évidemment aussi des paramètres w du réseau,

la sortie désirée est $S_{t,i} = H(k_t,c)$, avec H la fonction de Heaviside :

$$H(k,c) = \left\{ \begin{array}{l} 1 \text{ si } k = c \\ 0 \text{ si } k \neq c \end{array} \right\}$$

donc le coût se réécrit :

$$C = \frac{1}{N} \sum_{t=1}^{N} \left(\sum_{c} (H(k_t,c) - g(x_t,c))^2 \right)$$

Avec un nombre suffisant d'exemples d'apprentissage N, k étant fini, une fonction aléatoire q(x,k) peut s'écrire :

$$\lim_{N->\infty} \sum_{t=1}^{N} q(x_t,k_t) = \int_x \int_k q(x,k)p(x,k)dxdk = \int_x \sum_{k} q(x,k)p(x,k)dx$$

Le coût C devient :

$$C = \frac{1}{N} \sum_{c} \int_x \sum_{k} (H(k,c) - g(x,c))^2 \; p(x,k) \; dx$$

On cherche à interpréter g(x,c). Soit l'erreur locale e(x,c) pour l'élément x et pour le neurone de sortie c,

$$e(x,c) = \sum_{k} (H(k,c) - g(x,c))^2 \; p(x,k)$$

L'erreur locale e(x,c) est le résultat de l'apprentissage des classes k=1,...,n. La somme sur k peut être exprimée en fonction des sorties désirées, si k=c le neurone de sortie désirée vaut 1 sinon il prend la valeur 0.

p(x,k) est la densité d'exemples de sortie désirée à 1,

et p(x) - p(x,k) est la densité d'exemples de sortie désirée à 0,

on utilise de plus la règle $p(x,c) = p(c,x) = p(x)p(c/x)$.

on obtient:

$$e(x,c) = p(x,c)[1-g(x,c)]^2 + [p(x)-p(x,c)][0-g(x,c)]^2$$

$$e(x,c) = p(x)\,[p(c/x)[1-g(x,c)]^2 + [1-p(c/x)][-g(x,c)]^2]$$

en développant l'expression, on trouve :

$$e(x,c) = p(x)\,[p(c/x) + 2p(c/x)g(x,c) + g(x,c)^2]$$

soit encore :

$$e(x,c) = p(x)\,[[p(c/x)-g(x,c)]^2 + p(c/x)[1-p(c/x)]]$$

Le 2ième terme ne dépend pas de g(x,c), on peut donc le considérer séparément. Pour minimiser l'erreur e(x,c), on doit minimiser le premier terme. La fonction e(x,c) est minimum si $g(x,c) = p(c/x)$.

Lorsque l'on cherche à minimiser l'erreur, les sorties désirées tendent toutes vers les probabilités $P(C_i/x)$. Donc les vecteurs $Y_{i=1,...,n}$ sont des approximations des $P(C_i/x)$,

$$f(x,w) = C_i \text{ tel que } p(C_i/x) = \underset{c=1,..,n}{\text{Max}} (p(C_c/x))$$

Les paramètres w contiennent la structure du réseau et les poids des connexions. On retrouve la règle de décision de Bayes dans l'équation du classifieur neuronal.

Un résultat similaire est obtenu si on utilise le critère de "*cross-entropy*" [Hinton 87], [Fallside 91]. Le coût total s'écrit :

$$C = -\sum_{t=1}^{N}\left(\sum_{i=1}^{n} S_{t,i} * \log\frac{S_{t,i}}{Y_{t,i}} + (1 - S_{t,i}) * \log\frac{(1 - S_{t,i})}{(1 - Y_{t,i})}\right)$$

soit après avoir modifié les notations comme pour le critère MSE :

$$C = -\sum_{t=1}^{N} \left(\sum_{c} H(k_t,c) \log \frac{H(k_t,c)}{g(x_t,c)} + (1 - H(k_t,c)) * \log \frac{(1 - H(k_t,c))}{(1 - g(x_t,c))} \right)$$

de la même façon, on peut exprimer l'erreur e(x,c) pour un neurone de sortie c :

$$e(x,c) = \sum_{k} p(x,k) \, [H(k,c) \log \frac{H(k,c)}{g(x,c)} + [1 - H(k,c)] \log \frac{1 - H(k,c)}{1 - g(x,c)}]$$

Si les sorties sont comprises entre 0 et 1. De la même façon que précédemment, on sépare le terme où k=c du terme où $k \neq c$, on obtient alors :

$$e(x,c) = -p(x) \, [p(c/x) \log g(x,c) + [1-p(c/x)]\log[1-g(x,c)]]$$

La dérivée de la fonction $p(c/x) \log g(x,c) + [1-p(c/x)]\log[1-g(x,c)]$, soit $\frac{p(c/x)}{g(x,c)} - \frac{(1 - p(c/x))}{(1 - g(x,c))}$ s'annule pour $g(x,c) = p(c/x)$.

L'optimum global est donc obtenu si les sorties des réseau sont exactement les probabilités des classes. Cependant, dans la réalité, cette approximation est d'autant plus fiable que le réseau répond à certaines conditions :

- la structure du réseau doit être suffisament flexible pour une telle approximation,

- le réseau de neurones doit avoir un nombre suffisamment grand de paramètres libres,

- le nombre d'exemples d'apprentissage doit être assez grand pour que l'algorithme d'apprentissage soit capable de trouver un optimum global.

- Il est aussi important d'avoir des valeurs de sorties des réseaux entre 0 et 1 dont la somme est égale à 1 pour parler de probabilités.

Il y a deux façons de procéder pour que $\sum_{c} P(C_c/x) = 1$:

- Respecter cette contrainte lors de l'apprentissage. Le critère de MSE est alors utilisé avec des contraintes sur les sorties pour déterminer les $P(C_i/x)$ [Bourlard 90] [Bottou 91] et correspond à une approximation du principe de Maximum A Posteriori (MAP),

- Utiliser le critère de MSE et ensuite, forcer la somme des $P(C_i/x)$ d'être à 1 après apprentissage.

On peut remplacer le critère de discrimination des réseaux de neurones par le critère de maximisation du logarithme des probabilités des classes [Bridle 89], [Brown 87] avec les conditions strictes sur les sorties, c'est-à-dire d'être comprises entre 0 et 1 et d'avoir une somme égale à 1. Ce critère est alors tout à fait comparable au "*Maximum Mutual Information*".

En conclusion, on a interprété les sorties des réseaux en termes probabilistes et retrouvé le critère de décision bayésienne. Mais ce résultat va plus loin, quelque soit la structure du réseau, linéaire ou non linéaire, un critère d'apprentissage discriminant permet d'estimer les probabilités a posteriori. Cependant, les réseaux n'ont pas en général assez de degrés de liberté ou de paramètres pour donner exactement les probabilités des classes. Ils sont seulement capable de donner des approximations des probabilités des classes pour chaque point (x,c).

2.4.5.2 L'algorithme de LVQ: "*Learning Vector Quantization*"

L'algorithme de LVQ ou "Learning Vector Quantization" [Kohonen 84] opère sur des vecteurs déjà quantifiés et utilise une règle de décision du type plus proche voisin. Les vecteurs utilisés sont souvent issus d'un algorithme de quantification vectorielle classique: k-moyennes, boules optimisées... ou d'une carte topologique.

Cet algorithme permet de modifier les frontières afin de minimiser les erreurs de classification. Il existe plusieurs algorithmes LVQ avec différentes règles d'adaptation. Ces méthodes sont décrites en détail dans le chapitre 3 "méthodes connexionnistes".

L'algorithme LVQ utilise la règle de décision f(x,w) de type plus proche voisin. Soit N observations de $X=\{x_1,x_2,x_3...,x_N\}$, V le nombre de vecteurs quantifiés v_i, $C(v_i)$ la classe du vecteur v_i, pour un échantillon x, w représentant les vecteurs quantifiés, on a :

$$f(x,w) = C(v_i) \text{ tel que } d(x,v_i) = \min_{c=1,...,V} d(x,v_c)$$

Il a été démontré que l'algorithme de LVQ est un cas particulier d'une méthode plus générale de descente de gradient [Bottou 91], [McDermott & Katagiri 92]. Le résultat principal est de définir une fonction de coût générale différentiable. la fonction définie pour l'algorithme de LVQ étant une fonction discriminante, on peut alors se ramener à l'interprétation probabiliste des réseaux décrits dans le paragraphe 2.4.5.1.

2.5 Les modèles markoviens

Les modèles markoviens [Jelinek 76], [Rabiner 86], [Lee 88] sont les systèmes les plus performants à l'heure actuelle en reconnaissance de la parole. Ils servent de référence dans le domaine. Pour cette approche paramétrique de reconnaissance des formes, chaque forme (phonèmes, mots) est représentée par un modèle. Les modèles de Markov permettent de traiter dans le même cadre la modélisation stochastique du processus de production de signal et la modélisation de la structure temporelle des séquences de parole. La puissance de ces systèmes est liée à l'utilisation du même formalisme probabiliste et des mêmes algorithmes pour traiter toute la chaine de reconnaissance, des formes acoustiques aux formes graphémiques, en intégrant des connaissances de tous niveaux: phonétiques, lexicales, syntaxiques.

Soit un modèle de Markov C , cf 2.5.1, compte tenu des probabilités de transitions et d'émissions de spectres associées au modèle, on peut calculer la probabilité d'émission d'une séquence de spectres de parole. Cette séquence de spectres est appelée une observation. Si chaque modèle code un symbole, on choisira comme symbole reconnu pour cette observation, le modèle dont la probabilité sera la plus grande.

Le critère d'apprentissage le plus utilisé pour ces modèles est celui du maximum de vraisemblance, "*Maximum Likelihood Estimation*" (MLE). On cherche à déterminer les paramètres des modèles de façon à maximiser la probabilité de générer l'ensemble des mots ou phonèmes du corpus d'apprentissage suivant ce critère. Si x est une observation, les modèles markoviens sont alors des classifieurs qui proposent une estimation des p(x/C), dans le cas de modèles continus ou semi-continus et des P(x/C) dans le cas de modèles discrets.

La mise en oeuvre des modèles markoviens posent 3 problèmes :

- le calcul de la ressemblance entre un modèle et une observation,
- l'estimation des paramètres d'un modèle,
- la suite d'états la plus probable pour une observation et un modèle donné.

Nous aborderons ces points dans les paragraphes 2.5.2 et 2.5.3

2.5.1 Modèles

Un modèle de Markov est un modèle stochastique de production de spectres de parole. Chaque modèle est un automate probabiliste composé d'une suite d'états liés par des probabilités de transition $a_{i,j}$. L'émission de spectre attachée en général à un état peut être définie soit par des probabilités d'émission b_i, c'est le cas des modèles discrets, soit par des densités de probabilité d'emission données par des lois b_i, c'est le cas des modèles continus. Les hypothèses implicites des modèles, pour des raisons de temps de calcul, sont que la probabilité a priori d'un état ne dépend que de l'état précédent (on parle alors de modèle d'ordre 1) et que les observations ne dépendent que de l'état courant.

On utilise le terme de modèles de Markov cachés, Hidden Markov Models ou HMMs pour parler des modèles. A l'encontre des vecteurs acoustiques, les états ne sont pas observés, ils sont dits "cachés". La figure 2.2 est un exemple très courant de topologie de modèle de Markov, le modèle de Bakis. Les probabilités notées $a_{i,j}$ sont les probabilités de transition, les probabilités bi sont les probabilités d'émission d'une observation. Soit W=(W1,...,WT) une séquence d'états d'un modèle, à chaque modèle de Markov, on associe le vecteur de probabilités initiales A0, la matrice des probabilités de transition entre deux états W_t et W_{t+1} avec la contrainte $\sum_{j \in W} a_{i,j} = 1$ et un ensemble de lois d'émission associées aux états Wt du modèle.

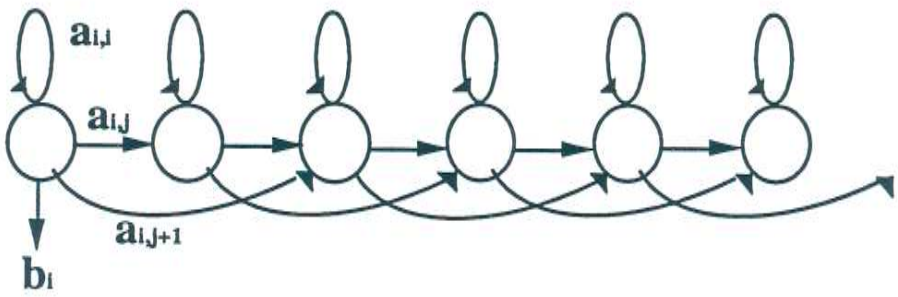

Fig. 2.2 : ***Représentation d'une chaîne de Markov, appelée modèle de Bakis.***

Ce modèle autorise l'élision et l'ajout de plusieurs trames, il peut ainsi coder très précisément l'aspect temporel d'une forme et la variabilité de la parole mais il n'est pas en fait très réaliste. En pratique, on ajoute souvent des heuristiques à cette modélisation comme, par exemple des modèles de durées des phonèmes. Pour le modèle de Bakis, à partir d'un état, on peut rester sur place, avancer d'un état ou de deux états. Ces différents types de transitions permettent de gérer l'aspect temporel des vecteurs acoustiques.

Modèles discrets, semi-continus et continus

Soit un modèle donné, de paramètres $a_{i,j}$ et b_j:

Si le modèle est discret, la loi d'emission b_j peut être donnée exhaustivement, c'est une probabilité,

Dans le cas continu ou semi-continu, cette loi est une loi gaussienne ou laplacienne ou une somme de densités de probabilités d'émission ("mixtures densities"). Dans le cas de modèles semi-continus, les densités de probabilités sont communes à l'ensemble des modèles. Pour les modèles continus, les densités de probabilités dépendent du modèle.

Par exemple, pour une structure de modèles markoviens continus, les paramètres étant par hypothèse des lois normales univariables ou multivariables, on cherchera à estimer les couples $N(\mu,\sigma^2)$ ou $N(\mu,\Sigma)$, μ désignant le vecteur moyen de la loi normale, σ la variance de ce vecteur et Σ la matrice de co-variance.

Les modèles discrets et continus sont détaillés dans le chapitre 8.

2.5.2 Différents algorithmes des HMMs

La mise en oeuvre des HMMs nécessite plusieurs algorithmes :

- l'algorithme d'évaluation: il calcule la probabilité pour un HMM de produire la séquence d'observations donnée,

- l'algorithme d'apprentissage: Etant donné un modèle C et un ensemble d'observations $X=(x_1, ...,x_T)$, il calcule comment modifier les paramètres du modèle pour avoir la meilleure probabilité de générer de telles observations.

- l'algorithme de décodage: il calcule la séquence d'états d'un HMM qui produit une séquence d'observations donnée,

L'évaluation

La vraisemblance L (*L pour likelihood*) d'une observation par rapport à une séquence d'états est le produit des $a_{i,j}$ et des b_i correspondant à cette séquence

d'états.. Soit $X=(x_1, ...,x_T)$ une séquence d'observations, soit $W=(W_1,...,W_T)$ une séquence d'états, soit $i = W_t$ et $j = W_{t-1}$

$$L(X/W) = \prod_{t=1}^{T} a_{ij}\, b_i(x_t)$$

avec $P(X/W) = L(X/W)$

La vraisemblance L sur l'ensemble S des séquences d'états $W(W_1...W_T)$,

avec $i = W_t$ et $j = W_{t-1}$

$$L(X) = \sum_{\{W_1,...W_T\}\in S} L(X/W)$$

$$L(X) = \sum_{\{W_1,...W_T\}\in S} \prod_{t=1}^{T} a_{ij} b_i(x_t)$$

Calcul recursif de la vraisemblance

Il est possible grâce aux hypothèses des modèles de Markov de calculer très simplement $L(X_1,...,X_T/W)$ de façon récursive et de l'écrire:

$$L(X/W) = \sum_{j\in W} L(x_1,...,x_t,W_t=j)\; L(x_{t+1},...,x_T / W_t=j)$$

à l'instant t, pour l'état W_t:

soit $\alpha_t(j) = L(x_1,...,x_t,W_t=j)$

soit $\beta_t(j) = L(x_{t+1},...,x_T / W_t=j)$

on a donc

$$L(X/W) = \sum_{j\in W} \alpha_t(j)\beta_t(j) \text{ avec } t\in[1...T]$$

L'algorithme forward calcule la vraisemblance d'une observation pour un modèle donné, c'est-à-dire $\alpha_T(W_T)$ la probabilité qu'un modèle W génère une séquence d'observations:

$$\alpha_t(j) = \left\{ \begin{array}{c} 0 \text{ si } t=0 \text{ et } j\neq W_1 \\ 1 \text{ si } t=0 \text{ et } j=W_1 \\ \sum_{i\in W} \alpha_{t-1}(i) b_j(x_t) a_{ij} \text{ si } t>0 \end{array} \right\}$$

L'apprentissage

L'algorithme de forward-backward (ou Baum-Welch) calcule les paramètres d'un modèle à partir du critère MLE. Il obtient en plus de $\alpha_t(j)$ qui est la probabilité du modèle W d'être dans l'état j et d'avoir généré la séquence $(X_1 \ldots X_t)$, $\beta_t(j)$, qui est la probabilité de W d'être dans l'état j et de générer la séquence $(X_{t+1} \ldots X_T)$.

$$\beta_t(j) = \left\{ \begin{array}{c} 0 \text{ si } t=T \text{ et } j \neq W_T \\ 1 \text{ si } t=T \text{ et } j=W_T \\ \sum_{k \in W} \beta_{t+1}(k) b_k(x_{t+1}) a_{jk} \text{ si } 0 \leq t \leq T \end{array} \right\}$$

Le décodage

L'algorithme de Viterbi cherche à produire la séquence d'états qui fournit la plus grande probabilité de générer la séquence d'observations, le signe $\sum$ de l'agorithme de forward est remplacé par MAX.

$$L(X) = \max_{\{w_1,\ldots w_T\} \in S} L(X/W)$$

$$\alpha_t(j) = \left\{ \begin{array}{c} 0 \text{ si } t=0 \text{ et } j \neq W_1 \\ 1 \text{ si } t=0 \text{ et } j=W_1 \\ \max_{i \in W} \{\alpha_{t-1}(i) b_j(x_t) a_{i,j}\} \text{ si } t>0 \end{array} \right\}$$

L'algorithme de Viterbi permet une recherche plus rapide que l'algorithme de forward et retrouve explicitement la meilleure séquence d'états.

2.5.3 Apprentissage

Le but de l'apprentissage est d'estimer les paramètres des HMMs pour modéliser au mieux les séquences de parole. On utilise communément **l'algorithme de Baum-Welch** qui est une variante de **l'algorithme forward-backward** ou **l'algorithme de Viterbi** [Rabiner, Juang 86] . L'apprentissage par Viterbi n'utilise que l'information du meilleur chemin pour recalculer les probabilités des modèles. Pour avoir une meilleure estimation de ces paramètres, il est préférable d'utiliser les algorithmes de Baum-Welch ou forward-backward. Cependant, d'un

point de vue pratique, l'algorithme de Viterbi offre des performances équivalentes de reconnaissance avec un temps d'apprentissage beaucoup plus court.

La méthode d'apprentissage permet de faire correspondre les trames d'entrée avec les états des modèles. En fonction de cette segmentation, on modifie les probabilités $a_{i,j}$ et b_j en utilisant la réestimation des paramètres de Viterbi.

L'algorithme de Viterbi comprend :

une phase d'initialisation qui fixe l'ensemble les paramètres $a_{i,j}$ et b_j des modèles à des valeurs raisonnables,

puis de façon itérative, un alignement temporel et une réestimation des paramètres avec le critère de "*Maximum Likelihood*" tant que le critère de convergence de l'apprentissage n'est pas atteint.

Réestimation des $a_{i,j}$ et b_j

Cette réestimation des probabilités est donnée ci-dessous dans le cas de modèles discrets où chaque x_t est associé à un vecteur-code de classe C

Pour chaque état de chaque modèle, on compte le nombre d'émissions de chaque vecteur C_k que l'on divise par le nombre total d'émission de vecteurs dans cet état.

Soit γ_{mj} le nombre total d'émissions de l'état j du modèle m,

soit γ_{mjk} le nombre total d'émissions du vecteur C_k pour l'état j du modèle m,

$$\text{pour le modèle m, } b_j(k) = \frac{\gamma_{mjk}}{\gamma_{mj}}$$

De même, pour les transitions, on recalcule les $a_{i,j}$ en comptant le nombre de transitions de l'état i à l'état j et en divisant ce nombre par le nombre total de transitions issues de l'état i.

Soit ζ_{mi} le nombre de transitions issues de l'état i du modèle m,

soit ζ_{mij} le nombre de transitions de i vers j du modèle m,

$$\text{pour le modèle m, } a_{i,j} = \frac{\zeta_{mij}}{\zeta_{mi}}$$

En pratique, on calcule les ζ et γ pour toutes les phrases d'apprentissage avant de modifier les probabilités des modèles.

Nous présenterons l'algorithme de Viterbi pour les modèles markoviens discrets, continus utilisés et pour les méthodes hybrides développées dans le chapitre 8.

2.6 Reconnaissance par comparaison dynamique

Nous présentons à part le principe de la comparaison dynamique en reconnaissance car les classifieurs présentés dans le paragraphe précédent n'ont pas tous un mécanisme d'alignement temporel leur permettant d'intégrer des séquences de mots ou de phonèmes. C'est le cas, notamment, des modèles probabilistes neuronaux et des méthodes des k plus proches voisins.

Les deux systèmes de comparaison dynamique les plus utilisés sont :

- le "Dynamic Time Warping" DTW ou programmation dynamique "[Wintsjuk 68][Sakoe, Chiba 71],[Gauvain 82] qui est un algorithme de reconnaissance des formes par alignement temporel.

- l'algorithme de Viterbi qui est aussi un alignement temporel sur les chaînes de Markov cachées (HMMs) [Baker 75];, [Jelinek 76], [Jouvet 88]. L'approche par modèles markoviens construit des références acoustiques avec un niveau d'abstraction en plus, la modélisation temporelle des formes acoustiques.

2.6.1 Approche par alignement temporel: DTW

Il s'agit de mettre en correspondance deux formes afin d'obtenir une coïncidence optimale au sens d'un critère à définir en fonction du domaine étudié.

Appliqué à la reconnaissance de la parole, cet algorithme cherche à assurer la correspondance des spectres de deux formes trame à trame. Le DTW permet de dilater ou comprimer les mots dans le temps de façon à les faire coïncider au mieux à un modèle fixe.

Soient deux mots M (mot inconnu) = (m1...mi) et R (mot référence) = (r1...rj), l'ajustement optimal entre ces deux mots est représenté par un chemin croissant dans la matrice [1,I]x[1,J] de la figure 2.4. A chaque point (i,j) du chemin C, mettant

en correspondance la trame i de M à la trame j de R, est associée une distance locale d(mi,rj). Le coût du chemin C, noté g(i,j), est égal aux distances cumulées sur tous les points du chemin. Les transitions possibles entre les points du chemin sont gérées par des équations locales.

Les équations locales, par exemple fig 2.3, limitent l'espace de recherche des trames à mettre en correspondance. Des méthodes de "*pruning*" permettent d'accélérer les calculs de g(i,j) en restreignant encore le nombre de chemins possibles.

On se limite à des transitions des points (i,j) à (i',j') avec $(i' - i) \geq 0$ et $(j' - j) \geq 0$.

Les distorsions temporelles possibles sont décrites par l'équation locale, c'est-à-dire les élisions ou ajouts de trames possibles à la forme testée pour coïncider au mieux avec la forme référence. Le parallèle est immédiat avec les modèles markviens. Cette équation locale code les transitions entre états des modèles markoviens.

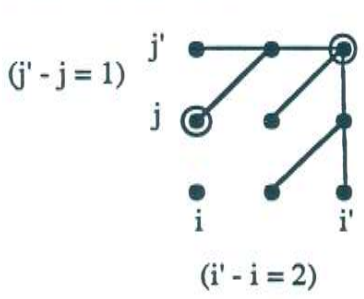

Fig. 2.3 : Equation locale.

Les conditions classiques d'initialisation sont que g(0,0) = 0 et que g(0,j>0) et g(0,i>0) sont à l'infini.

Il existe de nombreuses équations locales correspondant à différents modèles markoviens :

- l'équation locale symétrique [Sakoe, Chiba 71] avec a=1 et b=2 ou a=b=1)

$$g(i,j) = \min \begin{bmatrix} g(i-1,j) + a\ d(m_i,r_j) \\ g(i-1,j-1) + b\ d(m_i,r_j) \\ g(i,j-1) + a\ d(m_i,r_j) \end{bmatrix}$$

- l'équation locale asymétrique

$$g(i,j) = \min \begin{bmatrix} g(i-1,j) + d(m_i,r_j) \\ g(i-1,j-1) + d(m_i,r_j) \\ g(i-1,j-2) + d(m_i,r_j) \end{bmatrix}$$

Les conditions d'initialisation de ces équations diffèrent suivant le but: reconnaissance de parole continue, detection de mots... Pour détecter des mots ("word spotting"), l'initialisation est à zéro pour g(0,0) et pour l'axe des références g(0,j) et à l'infini pour l'axe i de la séquence inconnue g(i>0,0) permettant ainsi de détecter un mot commençant n'importe où dans la séquence.

3	5	8	2
6	9	5	4
12	4	2	6
2	3	9	4
3	6	8	1

29	29	28	24
26	24	20	23
20	15	15	21
8	11	20	24
6	12	20	21

*Tab. 2.6 : **Tableaux des distances locales et des distances cumulées.** Les g(i,j) (tableau de droite) sont calculés à partir du tableau des distances locales (à gauche) en suivant l'équation locale symétrique énoncée plus haut avec a=1 et b=2. Les conditions d'initialisation sont g(0,0) = 0, g(0,j>0) et g(i>0,0) sont à l'infini. Le chemin optimal d'appariement est indiqué en relief dans le tableau de droite.*

Le chemin optimal obtenu par alignement temporel dynamique :

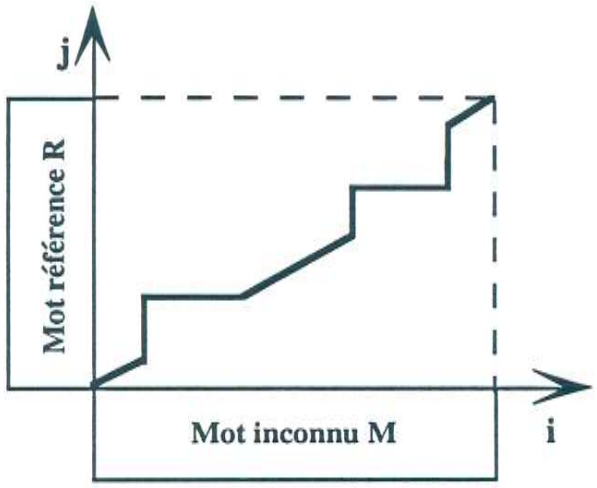

*Fig. 2.4 : **Alignement temporel par programmation dynamique.***

2.6.2 Algorithme de Viterbi

L'algorithme de Viterbi cherche à produire la séquence d'états qui fournit la plus grande probabilité de générer la séquence d'observations, cf 2.4.3.2. L'algorithme de Viterbi est très proche du DTW, les probabilités de transition peuvent être vues comme les poids associés aux équations locales de la programmation dynamique. Il est décrit plus en détail dans le chapitre 8.

2.6.3 Reconnaissance de parole continue

La comparaison dynamique est appliquée très facilement à la parole continue.

Pour le DTW, il suffit de concaténer les références des mots reconnus. Les équations locales sont différentes à la rupture des mots, elles utilisent par exemple un enchainement de mots issu d'un automate deterministe.

Pour les modèles markoviens, de la même façon, il est très facile de construire des super-modèles qui concatènent des modèles de phonèmes en mots puis de mots en phrases en utilisant des probabilités de transitions entre mots issues par exemple d'une grammaire probabiliste. La qualité majeure des HMMs est de pouvoir modéliser des séquences d'objets quelconques: vecteurs acoustiques, phonèmes, mots... Ainsi, on peut construire un système de Markov complet de reconnaissance de parole modélisant le niveau acoustique et le niveau syntaxique et permettant une recherche intégrée [Ney 84] [Ney 90]. Nous décrirons en détail l'architecture globale utilisée dans nos expériences pour la modélisation HMM et hybride au chapitre 8.

2.7 Conclusion

Nous avons présenté les principales difficultés en décodage acoustico-phonétique d'un système de reconnaissance de la parole, les différents traitements à mettre en oeuvre et décrit les méthodes statistiques de classification les plus utilisées.

Cette étude s'attache aussi à montrer l'intérêt des classifieurs neuronaux et la coopération possible entre différents modèles. Il est intéressant de faire quelques remarques sur les classifieurs décrits. Tous ces classifieurs sont optimaux avec une infinité de données même avec des hypothèses restrictives sur la forme des données. Les méthodes non paramétriques posent le minimum d'hypothèses, le choix d'un volume, les méthodes connexionnistes codent les connaissances a priori sur les données dans le nombre de couches et de cellules du réseau, enfin les

méthodes paramétriques font les hypothèses les plus restrictives directement sur la forme des fonctions de distributions des classes. Plus l'approche utilise de connaissances a priori, moins elle nécessite de temps de calcul mais plus elle a de chances de se tromper. D'autre part, moins on a de données, plus il est utile d'avoir des connaissances a priori. L'élaboration d'une méthode de classification est donc très liée aux contraintes de taille du corpus de données et de temps d'apprentissage disponible. On cherchera toujours à enrichir les données dont on dispose pour améliorer le pouvoir de généralisation des classifieurs, l'ajout de nouvelles données est une possibilité, une autre est d'essayer d'utiliser des critères différents pour extraire des connaissances en plus de l'ensemble de données. C'est le but recherché avec les méthodes connexionnistes.

Dans ce chapitre, les méthodes connexionnistes ont été situées par rapport aux classifieurs dits "classiques", elles sont maintenant décrites en détail dans le chapitre suivant. Une étude qualitative entre méthodes neuronales et modèles markoviens est présentée dans le chapitre 4 qui comporte aussi un état de l'art des systèmes hybrides neuronaux et markoviens.

3. Modèles connexionnistes

Les modèles neuromimétiques sont d'essence neurobiologique et ont été conçus pour reproduire les fonctions associatives du cerveau. On appelle ces modèles neuronaux, connexionnistes ou neuromimétiques suivant l'axe de recherche poursuivi. La dénomination la plus connue à l'heure actuelle est certainement celle de réseaux de neurones. Cependant, leurs unités appelées neurones formels, n'ont qu'une très lointaine parenté avec des cellules nerveuses. De plus, les algorithmes comme par exemple la rétropropagation n'ont aucune plausibilité biologique de par leur mode de transmission de l'activité "*forward-backward*". Le parallèle le plus important avec le modèle complexe du système nerveux réside dans le codage distribué des connaissances sur les connexions ou synapses entre les neurones.

3.1 50 ans d'histoire

En 1943, McCulloch et Pitts [McCulloch & Pitts 43] proposent un modèle mathématique très simplifié d'une cellule nerveuse: le neurone formel. Ils décrivent le comportement de ces neurones en structure de réseau. A la même époque, Hebb [Hebb 49] énonce un principe d'adaptation des neurones à leur environnement en modifiant l'efficacité des connexions entre les neurones. Le langage biologique donne à ce principe le nom de plasticité.

Les algorithmes d'apprentissage généralement utilisés dans les réseaux essaient d'extraire des indices statisquement caractéristiques des données. Les informations mémorisées sont interdépendantes et permettent le codage de connaissances non explicites de très bas niveau.

Les premiers réseaux sont prometteurs. Le perceptron [Rosenblatt 57] est une machine adaptative basée sur un classifieur linéaire permettant de calculer la plupart des fonctions logiques. Un rapport de Minsky et Pappert [Minsky & Pappert 68] révèle les limites de ces systèmes; Ils ne peuvent pas séparer des classes non linéairement séparables. En 1970, l'explosion des modèles basés sur la logique et le codage symbolique en intelligence artificielle conduit à l'abandon relatif des méthodes statistiques.

Depuis 10 ans, on assiste à un renouveau des méthodes statistiques connexionnistes. De nouveaux modèles ont été mis au point comme la machine de Boltzmann [Ackley 85] issue de la thermodynamique qui est une importante variante du réseau de Hopfield [Hopfield 82] puisque le fonctionnement des unités est maintenant probabiliste et le perceptron multicouche [Rumelhart 86] qui dépassent les limitations des perceptrons.

L'enrichissement des connaissances en neurobiologie a permis une modélisation de certaines fonctionnalités du système nerveux comme les colonnes corticales [Burnod 89].

D'autres modèles ont vu le jour comme la propagation guidée [Béroule 85] ou les algorithmes génétiques [Goldberg 88], [Booker 89] qui ne s'apparentent pas exactement à des modèles neuromimétiques.

Les années 80 correspondent aussi à un certain bilan d'échec de l'intelligence artificielle classique; tous les problèmes ne se résolvent pas en manipulant des données symboliques et explicites. Les modèles connexionnistes sont surtout performants pour des tâches de bas niveau. Ils ont besoin de peu de connaissances a priori pour classifier des formes. Ces deux systèmes, connexionniste et symbolique, peuvent être complémentaires. En reconnaissance de la parole, par exemple, de nombreuses structures utilisent des modules de règles symboliques pour traiter les problèmes d'analyse syntaxique et sémantique et des modèles statistiques pour classifier les formes dits de bas niveaux comme des phonèmes.

3.2 Rappels sur les réseaux de neurones

3.2.1 Les principes généraux

L'un des principes des modèles connexionnistes est la simplicité des éléments utilisés, comparée à la complexité des opérations réalisées. Le fonctionnement global des réseaux est régi localement par des lois de propagation d'activité. L'apprentissage se manifeste par les modifications des poids des connexions entre les cellules. Ces cellules sont très souvent toutes identiques et répondent suivant la même fonction de stimulation. Les poids associés aux connexions ou synapses reflètent la structure sous-jacente de l'ensemble des exemples présentés au réseau. Ces matrices de poids sont les représentations internes des réseaux. Les connaissances codées ne sont pas explicitement traduisibles en terme de règles.

Les réseaux ont des propriétés caractéristiques; l'apprentissage à partir d'exemples, la robustesse aux données bruitées, l'adaptabilité... Dans les systèmes traditionnels, des inférences partielles sont souvent insuffisantes à déclencher une action. Pour beaucoup d'applications, ces caractéristiques sont plus importantes que les propriétés de déterminisme, fiabilité et reproductibilité propres aux modèles traditionnels de représentation des connaissances.

Il existe de nombreux réseaux de neurones ayant des fonctionnements et des architectures différents. On peut les classer suivant plusieurs critères non exclusifs :

- apprentissage supervisé ou auto-organisation,
- réseaux à structure fixe ou dynamique,
- modèles statistiques ou heuristiques,
- réseaux à forte influence neuro-biologique ou plutôt statistiques,
- réseaux multicouches ou à une couche,
- réseaux linéaires ou non linéaires.

Les différentes méthodes seront décrites en fonction de 3 caractéristiques principales : l'architecture, l'algorithme d'apprentissage et la dynamique de reconnaissance.

3.2.2 Différents objectifs

Différents objectifs peuvent être poursuivis suivant le domaine de recherche ou d'application concerné : biologie, physique ou mathématique.

L'un de ces objectifs est de développer une représentation formelle du fonctionnement du système nerveux. A partir d'expériences physiologiques, une modélisation des traitements est élaborée. Elle permet de synthétiser les données et d'expérimenter des comportements artificiels. C'est plutôt l'optique des modèles d'essence biologique et psychologique. Les ouvrages du groupe PDP pour "*Parallel Distributed Processing*" [Rumelhart 86] montrent par quelques simulations le bien-fondé des techniques connexionnistes, entre autres en faisant un parallèle entre l'aspect distribué de ces techniques et le traitement distribué prenant place dans certaines fonctions du raisonnement humain.

Les réseaux connexionnistes sont aussi largement employés pour des tâches de classification, prédiction ou compression de données.

L'objectif poursuivi dans cette étude a été de construire des modèles performants pour des tâches de classification pour la reconnaissance de formes temporelles. Les modèles connexionnistes sont souvent utilisés comme des classifieurs statistiques. Utilisés comme systèmes de classification, les réseaux tentent d'approximer une solution qui minimise l'erreur globale. Cependant, plus que d'être simplement de nouvelles techniques permettant de classifier des données, ces systèmes offrent une nouvelle représentation des connaissances. Le connexionnisme rassemble un ensemble de techniques dont les points forts sont sans doute l'aspect parallèle et distribué des connaissances, l'aspect dynamique c'est-à-dire la possibilité d'apprentissage et l'aspect d'auto-structuration des connaissances, par exemple dans les couches cachées.

La description des algorithmes connexionnistes s'appuie sur la modélisation des neurones et de leurs interconnexions par des règles d'apprentissage. La modélisation la plus simple est le neurone formel ou automate à seuil [McCulloch & Pitts 43].

Les principales règles d'ajustement des poids pour les méthodes supervisées sont dérivées des méthodes de Hebb et Widrow-Hoff.

la règle d'apprentissage local de Hebb consiste à renforcer les poids en fonction des correlations avec la sortie [Hebb 61]. Cette règle est utilisée par exemple dans l'algorithme de LVQ.

La règle d'apprentissage adaptative de Widrow-Hoff ou règle du Delta consiste à rapprocher progressivement la somme pondérée de la sortie désirée. On cherche à réduire cette distance avec un algorithme de gradient, c'est le cas de l'algorithme de rétropropagation utilisé avec des MLPs.

3.3 Différents types d'unités

On peut distinguer 2 types de neurones suivant le calcul de leur entrée : un produit scalaire ou une fonction distance. On introduira les différents types de réseaux suivant la fonction d'activation (aussi appelée fonction de tranfert).

Nous introduisons ci-dessous les notations qui vont nous servir par la suite pour décrire les réseaux :

Soit L le nombre de couches,

N le nombre de neurones de la couche I connecté à la couche J,

$i \in [1,N]$

le vecteur d'entrée X, l'activité d'un neurone recevant en entrée ce vecteur sera noté X_j,

le vecteur de sortie Y, l'activité d'une cellule de sortie Y_i,

W_{ji} les poids des connexions entre les cellules i de la couche I précédent la couche J et les cellules j de la couche J,

W_{js} les poids des connexions entre la cellule seuil de la couche I et les cellules j de la couche J,

et f une fonction de transfert.

3.3.1 Produit scalaire

Nous décrivons dans ce paragraphe des réseaux à unités basées sur des automates à seuil. L'activité propagée est un produit scalaire. C'est le cas des réseaux de type Perceptron, Adaline et MLP "*Multi-Layer Perceptron*". Les réseaux MLPs seront présentés en détaillant l'architecture, l'apprentissage et la dynamique de reconnaissance dans la suite de ce chapitre.

3.3.1.1 Réseaux à unités basées sur des automates à seuil

L'unité de base des premiers modèles de réseau est un neurone formel [Mc Culloch, Pitts 43] (figure 3.1) qui permet de faire une séparation linéaire entre deux classes par un hyperplan. Cette unité est un automate à seuil qui fait une somme pondérée des activités qui lui parviennent (ou un produit scalaire) et propage cette activité à d'autres unités à travers une fonction de transfert qui est une fonction à seuil. Les activités en entrée d'un réseau sont le plus souvent des valeurs réelles, normalisées entre -1 et 1 ou 0 et 1 pour des raisons de calcul.

Soit I la couche précédent J, l'activité X_j d'un neurone de la couche J est de la forme :

$$X_j = \left(\sum_{i=1}^{N} W_{ji} * Y_i \right) - W_{js}$$

la sortie Y_j du neurone X_j est de la forme:

$$Y_j = f\left(\left(\sum_{i=1}^{N} W_{ji} * Y_i \right) - W_{js} \right)$$

avec f une fonction à seuil cf figure 3.3.

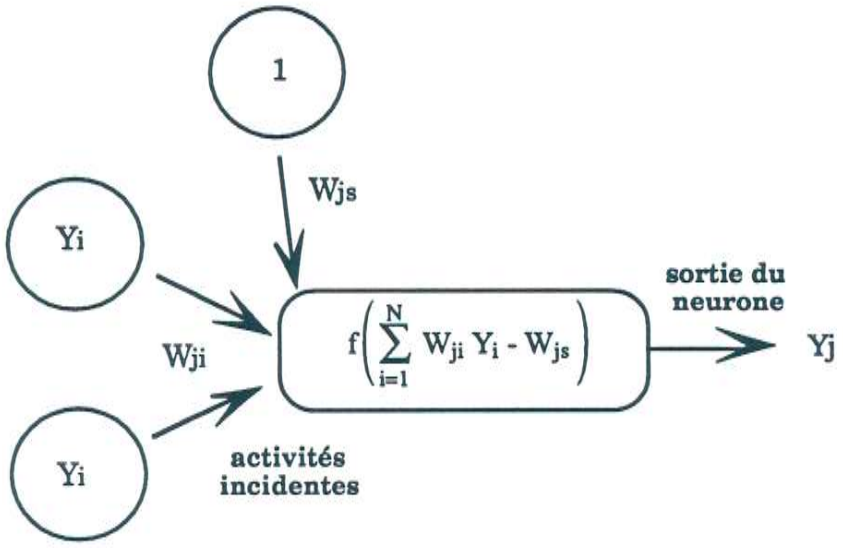

Fig. 3.1 : le neurone formel ou automate à seuil, f est une fonction à seuil..

Un automate à seuil sépare l'espace des entrées en deux demi-espaces. Supposons un automate à deux entrées, la cellule seuil permet de définir des hyperplans séparateurs ne passant pas par 0. S'il est possible de séparer les points, on parlera

de problème linéairement séparable. La figure 3.2 montre un exemple de problème non séparable linéairement par un automate à seuil à deux entrées.

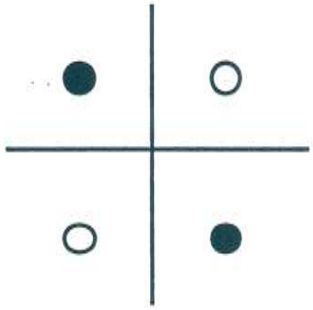

*Fig. 3.2 : **la fonction logique "OU exclusif" (XOR) non linéairement séparable par un automate à seuil à deux entrées.***

3.3.1.2 Fonctions de transfert

Les algorithmes d'apprentissage utilisant les algorithmes d'optimisation de l'analyse numérique nécessitent des fonctions de transfert dérivables.

La figure 3.3 montre 3 exemples de fonctions; à seuil, linéaire par moreceaux et sigmoïde. Pour une fonction de transition linéaire, on parle de réseaux à unités linéaires. Lorsque la fonction de transition choisie est une sigmoïde ou une autre fonction non-linéaire, on parle d'unités non-linéaires.

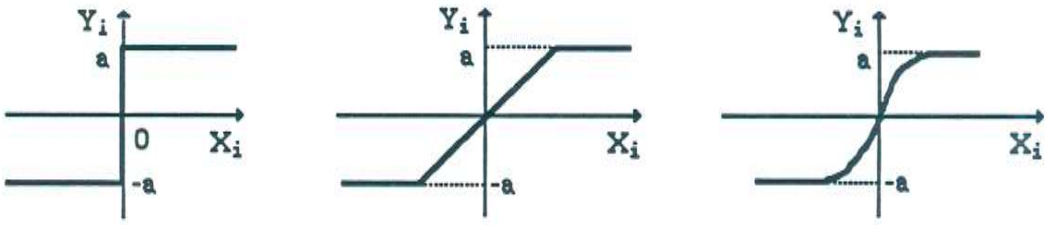

*Fig. 3.3 : **de gauche à droite, fonction à seuil, linéaire par morceaux et sigmoïde.***

Perceptron et Adaline

Le perceptron [Rosenblatt 57] et le système Adaline [Widrow et Hoff 60] sont les deux tendances des années 60. Le perceptron est un réseau basé sur des neurones formels, réalisé pour modéliser la perception visuelle. L'adaline est un système qui est composé d'une série d'entrées modulées par des résistances variables qui sont

ensuite sommées. L'adaline a été crée en vue d'application de traitement du signal. Ces deux systèmes utilisent un produit scalaire pour transferer l'activité.

MLP "*Multi-Layer Perceptron*"

Tout comme le perceptron, les unités de base du perceptron multicouche [Rumelhart 86] sont des neurones formels. La fonction de transfert souvent utilisée dans les MLPs est une fonction sigmoïde qui est non-linéaire. Lorsque le perceptron multicouche est utilisé avec des unités linéaires, la fonction de transfert global du réseau est alors linéaire.

La fonction sigmoide souvent choisie est la fonction symétrique f(x) = tanh(x). Elle permet un apprentissage plus rapide qu'une fonction non symétrique [Haffner 88].

Soit X_j la somme des activités pondérées arrivant sur une cellule, $f(X_j)$ la sortie de cette cellule. La fonction utilisée dans nos expérience avec les MLPs est :

$$f(X_j) = \tanh(\frac{kx}{2}) = \frac{(e^{kX_j} - 1)}{(e^{kX_j} + 1)}$$

3.3.2 Fonction de distance

Nous décrivons dans ce paragraphe des réseaux à unités basées sur un calcul de distance ou de similarité. C'est le cas des réseaux LVQ, des cartes topologiques et des RBFs "*Radial Basis Functions*". Les réseaux LVQ et cartes topologiques seront présentés en détaillant l'architecture, l'apprentissage et la dynamique de reconnaissance dans la suite de ce chapitre.

3.3.2.1 Réseaux à unités basées sur le calcul de distance ou de similarité

Cartes topologiques

Le réseau ne comporte pas de cellules cachées. Soit S la sortie d'un neurone, Xi le vecteur d'entrée, Wi un vecteur de référence dans la carte topologique ou "*features map*" [Kohonen 84] (cf 3.4.2), l'unité élémentaire d'une carte topologique se calcule avec :

$$S = \sum_i (W_i - X_i)^2$$

LVQ "*Learning Vector Quantization*"

Le réseau LVQ ne comporte pas de cellules cachées. Soit S la sortie d'un neurone, Xi le vecteur d'entrée, Wi le vecteur de référence de la carte l'unité élémentaire de l'algorithme de LVQ [Kohonen 88] (ou LVQ2) (cf 3.4.3) se calcule avec :

$$S = \sum_i (W_i - X_i)^2$$

RBFs "*Radial Basis Functions*"

On appelle réseaux à unités gaussiennes ou RBFs pour "*Radial Basis Functions*" [Poggio 89], un réseau dont la fonction de transition est une loi gaussienne. La zone d'activation d'une cellule gaussienne est un ellipsoïde dont le centre et l'étendue sont donnés par les paramètres de la gaussienne (moyenne et matrice de covariance diagonale). Une telle cellule réagit localement sur l'espace d'entrée.

Soit Y_i une sortie, X_j la jème composante du vecteur d'entrée, $\sigma_{i,j}$ l'écart type associé à la jème composante du vecteur d'entrée à N composantes, $m_{i,j}$ la moyenne associée à la jième composante du vecteur d'entrée.

L'équation d'une cellule gaussienne est donnée par l'équation suivante :

$$Y_i = e^{\left(-\sum_{j=0}^{N} \frac{(x_j - m_{i,j})^2}{(\sigma_{i,j})^2}\right)}$$

Les RBFs sont des réseaux à une couche cachée de N cellules gaussiennes et une couche de sortie de cellules linéaires. L'apprentissage par le gradient ou des techniques de l'algèbre linéaire est effectué en utilisant la méthode de pseudo-inverse calculé par la SVD "*Singular Value Décomposition*". Soit J le nombre de gaussiennes, soit i l'étiquette de la classe, x un exemple, w_{ij} le poids associé au centre de gaussienne j et de classe i, (μ_j, σ^2_j) la jième gaussienne de centre μ_j et de variance σ^2_j, h une constante, le critère de décision s'écrit :

$$C_i = \underset{i}{\text{Argmax}} \sum_{j=1}^{J} W_{ij}\, e^{\left(\frac{||x - \mu_j||^2}{2h\sigma^2_j}\right)}$$

Des comparaisons sont présentées dans le chapitre 4 "état de l'art".

3.4 Algorithmes d'apprentissage

3.4.1 Apprentissage supervisé et non supervisé

Nous présentons dans ce paragraphe les algorithmes utilisés dans cette étude. Nous décrivons tout d'abord un modèle neuronal non supervisé "les cartes topologiques" de Kohonen, puis des algorithmes supervisés, l'algorithme de LVQ et l'algorithme de rétropropagation du gradient.

Le mode d'apprentissage non supervisé consiste à ne pas utiliser l'étiquette des formes présentées. Les réseaux, suivant des règles locales de voisinage, s'auto-organisent pour représenter l'espace d'entrée.

Le mode d'apprentissage supervisé consiste à présenter aux systèmes des couples entrée-sortie (caractéristiques de la forme et étiquette de la forme) qui sont associés par les réseaux suivant des règles locales. Les mécanismes d'apprentissage des réseaux sont simples, ils consistent en des opérations sur les neurones cf 3.3 et des modifications des poids des connexions suivant des règles d'apprentissage.

3.4.2 Cartes topologiques

Les CARTES TOPOLOGIQUES ou "features map" (FM) [Kohonen 84] sont des réseaux qui compriment les données en gardant une représentation qui respecte leurs caractéristiques de voisinage, c'est à dire une topologie de l'espace d'entrée. Au cours de l'apprentissage, l'espace de représentation se partitionne sans supervision. Nos premières expériences de classification avec des réseaux connexionnistes ont été menées avec ces modèles.

Architecture

Le réseau ne comporte pas de cellules cachées. C'est un réseau à deux couches dont la couche de sortie est munie d'une structure topologique.

La représentation des formes est spatiale. Ce réseau est une carte de neurones connectés à leur voisinage où chaque neurone est représenté par un vecteur de dimension égale à la dimension des entrées. On définit a priori la forme du voisinage et les dimensions de la carte de représentations. Une droite présente un voisinage trop réduit qui ne permet pas de respecter la topologie des formes. A l'inverse, une structure tri-dimensionnelle ajoute beaucoup de voisinage, donc de

vecteurs non représentatifs des formes d'entrée. Elle est aussi plus coûteuse en temps de calcul. Un plan est la représentation la plus utilisée.

Algorithme d'apprentissage

L'apprentissage est non supervisé, on cherche le plus proche voisin d'une forme sans connaître son étiquette.

Initialisation

Les unités de la carte sont des vecteurs de même dimension que les formes présentées en entrée du réseau. Ils sont initialisés aléatoirement à des valeurs proches de zéro.

Les vecteurs d'entrée sont présentés de façon itérative au réseau. L'algorithme calcule le vecteur de la carte le plus proche du vecteur présenté au sens d'une distance euclidienne et modifie ses coefficients ainsi que les coefficients des vecteurs voisins dans la carte suivant une fonction de pondération. Cette fonction décroît en fonction du degré de voisinage. Au cours de l'apprentissage, le degré de voisinage est réduit et les pondérations diminuent. Le voisinage choisi est en général carré (figure 3.4).

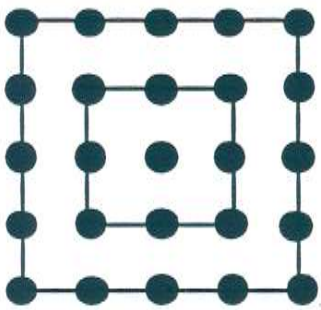

*Fig. 3.4 : **Forme du voisinage**: connexions dans la carte , le centre équivaut au degré 0, cette carte a un degré de voisinage de 2.*

Les notations utilisées sont les suivantes : V_i est un vecteur de la carte, x un vecteur d'entrée, d(x,y) la métrique utilisée, a(t,p) une fonction de pondération où t représente le temps c'est-à-dire le nombre de cycles d'apprentissage et p le degré de proximité du voisinage (0,1,2,3...), m le degré maximum de voisinage et b(t) une fonction décroissante de t, c'est à dire du nombre de cycles d'apprentissage.

Un cycle d'apprentissage équivaut à une présentation d'un exemple de chacune des formes à classer.

Si V_i est le vecteur le plus proche de x, les vecteurs du voisinage sont notés $V_{i,p}$ et $V_i = V_{i,0}$.

Règle d'adaptation des vecteurs de la carte:

pour p=0,m $\quad V_{i,p}(t+1) = V_{i,p}(t) + a(t,p)\, d(x(t),V_{i,p}(t))$

pour p=0,m $\quad a(t+1,p) = b(t)a(t,p)$

Le critère d'arrêt de l'apprentissage est atteint lorsque la carte converge vers un état stable ou pour un nombre fini d'étapes.

Le temps de convergence et l'optimalité de l'algorithme d'apprentissage dépendent de plusieurs facteurs:

- la forme de la carte,
- les dimensions de la carte (le nombre de vecteurs),
- l'ordre de présentation des exemples,
- la métrique,
- la forme du voisinage,
- la taille du voisinage,
- la fonction de modification des coefficients en fonction du temps et du voisinage.

Reconnaissance

Les FMs ne sont pas des classifieurs mais des méthodes de quantification ou de projection des données. Après apprentissage, si on veut utiliser les cartes topologiques pour faire de la classification, on doit connaître les étiquette des formes apprises. Une étiquette est alors associée à chaque vecteur de la carte avec une règle de type plus proche voisin (1NN) cf §2.4.3.2 par un vote majoritaire. Un vecteur reçoit l'étiquette majoritairement reconnue parmi tous les exemples de la base de données qui sont les plus proches de cette référence.

3.4.3 Algorithme de "*Learning Vector Quantization*"

L'algorithme de LVQ ou "*Learning Vector Quantization*" [Kohonen 88] est un classifieur du type "plus proche voisin" qui opère sur des vecteurs déjà quantifiés. L'algorithme d'apprentissage est supervisé, il permet de modifier les frontières des classes afin de minimiser les erreurs de classification. On utilise souvent des vecteurs issus d'un algorithme de quantification vectorielle classique : k-moyennes, algorithme de "*split*"... Nous avons utilisé un algorithme neuronal non supervisé, les cartes topologiques, ou FMs pour "*Features Map*", afin de quantifier les vecteurs, puis un algorithme de type LVQ pour superviser cet apprentissage.

Architecture

On utilise un réseau à deux couches sans couche cachée. 1 groupe de cellules est attribué à chaque classe. On utilise une carte topologique pour initialiser le réseau, si une QV sert à initialiser le réseau LVQ, la topologie des formes n'est pas respectée.

Chaque classe est déterminée par un ensemble de plusieurs références. Plusieurs nuages distincts peuvent représenter la même classe. Les frontières entre ces ensembles sont souvent "non-linéaires" et résultent de la frontière d'influence entre les différents vecteurs de référence. On peut tracer le diagramme de Voronoï (figure 3.5) associé aux références qui montre une frontière linéaire par morceaux entre 2 classes.

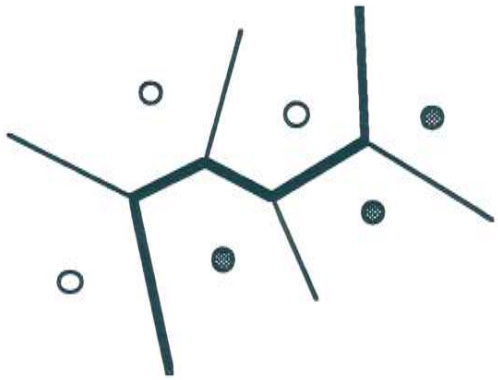

Fig. 3.5 : ***Frontière linéaire par morceaux autour de 2 classes: diagramme de Voronoï.*** *1 classe est représentée par les ronds sombres et l'autre les ronds clairs.*

Il existe plusieures variantes de l'algorithme de LVQ dont la plus connue est certainement l'algorithme de LVQ2 [Kohonen 88].

Algorithmes d'apprentissage

L'algorithme de LVQ:

Soit C_i l'étiquette du vecteur V_i de la couche de sortie, x un vecteur d'entrée de classe C, $\varepsilon(t)$ une fonction de pondération, on détermine la classe de l'exemple tiré au hasard x en prenant la classe C_i du vecteur V_i le plus proche au sens d'une distance euclidienne.

si C_i, la classe de V_i, est différente de la classe C de x, le vecteur V_i est éloigné de x :

$$V_i(t+1) = V_i(t) - \varepsilon(t)\,(x(t) - V_i(t))$$

si C_i, la classe de V_i, est identique à la classe C de x, le vecteur V_i est rapproché de x:

$$V_i(t+1) = V_i(t) + \varepsilon(t)\,(x(t) - V_i(t))$$

La fonction $\varepsilon(t)$ décroît en fonction de t.

L'algorithme de LVQ2:

Soit C_i,C_j les étiquettes des vecteurs V_i et V_j de la couche de sortie, x un vecteur d'entrée de classe C, $\varepsilon(t)$ une fonction de pondération, on détermine la classe de l'exemple tiré au hasard x en prenant la classe du vecteur V_i le plus proche au sens d'une distance euclidienne. V_j est le second vecteur plus proche de x au sens de la même distance. Il existe des définitions différentes pour V_j notamment celle du vecteur de la carte de classe C le plus proche de x.

si C_i, la classe de V_i, est différente de la classe C de x et que C_j, la classe de V_j est la classe de x, les vecteurs V_i et V_j sont modifiés suivant les équations:

Règle d'adaptation des vecteurs

$$V_i(t+1) = V_i(t) - \varepsilon(t)\,(x(t) - V_i(t))^2$$

$$V_j(t+1) = V_j(t) + \varepsilon(t)\,(x(t) - V_j(t))^2$$

dans tous les autres cas, on ne fait rien,

$$\text{sinon } V_{ij}(t+1) = V_i(t)$$

On peut aussi utiliser les mêmes équations avec des conditions plus contraignantes, par exemple:

que $(x(t) - V_j(t))^2 < (1+\delta)(x(t) - V_i(t))^2$

le paramètre δ contrôle la finesse de l'approximation au coût du risque de mauvaise classification [Bottou 91].

La fonction $\varepsilon(t)$ décroît en fonction de t.

Ces deux algorithmes sont très simples et donnent de bons résultats si les vecteurs de référence ont été bien initialisés. Des expériences de classification phonétique en français avec les FM et LVQ2 sont présentées dans le chapitre 5.

Reconnaissance

Une étiquette étant associée à chaque vecteur de la couche de sortie, on sélectionne l'étiquette du vecteur le plus proche de la forme présentée par une règle de type plus proche voisin (1NN) cf §2.4.3.2.

3.4.4 L'algorithme de rétropropagation du gradient et le perceptron multicouche

L'ancêtre du perceptron multicouche MLP, le perceptron [Rosenblatt 57] ne comportait pas de couches cachées et ne pouvait séparer des formes non linéairement séparables comme le problème simple du XOR (ou exclusif). Une étude théorique complète du perceptron: conditions de convergence, complexité des problèmes... est décrite dans [Minsky et Papert 68].

Architecture du perceptron multicouche: MLP

Le perceptron multicouche repousse les limites du perceptron de Rosenblatt. Le MLP est basé sur le calcul de frontières non linéaires séparant les classes dans l'espace des formes.

Sa structure comporte plusieurs couches ce qui permet de séparer des formes non linéairement séparables. Il a été démontré [White 89] que toute fonction continue de $[-1, +1]^n$ dans R, pouvait être approchée uniformément par un réseau possédant une couche cachée d'unités sigmoïdes. La structure multicouche est un filtre de

convolution non linéaire qui à partir d'une projection des activités sur plusieurs couches peut séparer des formes non linéairement séparables.

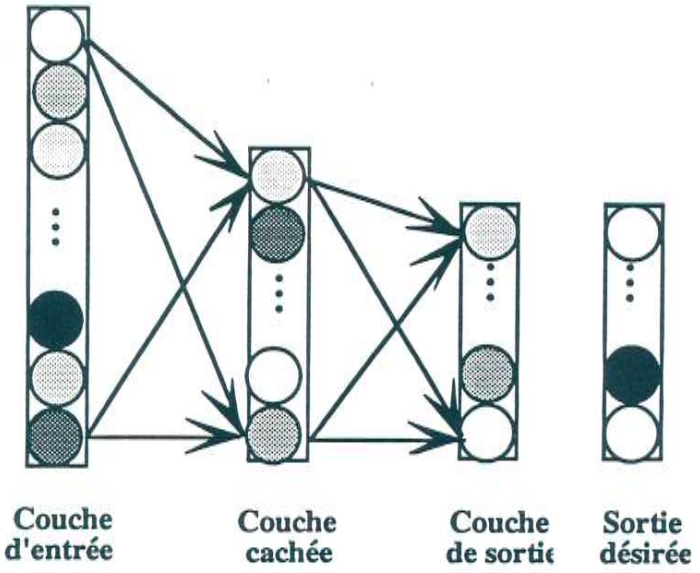

*Fig. 3.6 : **Perceptron multicouche***

Le perceptron multicouche (figure 3.6) est un modèle neuronal à apprentissage supervisé. C'est un système associatif qui permet de mémoriser des couples (entrée - sortie), par exemple **séquence de parole (en entrée) - étiquette linguistique.** L'algorithme d'apprentissage le plus communément utilisé est la rétropropagation du gradient d'erreur.

Ces modèles peuvent être vus comme des modèles statistiques pourvu que le nombre d'échantillons soumis au réseau soit suffisant. A la différence des modèles markoviens, l'hypothèse majeure des réseaux est faite sur la forme du processus de reconnaissance; une cascade de fonctions non linéaires et non sur la forme des données. Il ne s'agit donc pas de modèles parmétriques au sens de [Duda 73], cf chapitre 2.

Le traitement de l'information réalisé par un perceptron multicouche (fig. 3.4) repose sur l'interaction d'activités d'unités élémentaires à l'intérieur d'un réseau. Parmi ces unités, on peut distinguer :

- les unités d'entrée, stimulées par des vecteurs d'activité qui correspondent aux formes à classifier. Les vecteurs d'activité peuvent être binaires ou réels, les composantes sont en général normalisées entre [-1,+1] ou [0,+1].

- les unités cachées, inaccessibles à l'utilisateur, dont l'activité dépend des relations avec les autres cellules. Ces relations se font par l'intermédiaire des connexions dont les valeurs associées sont établies par apprentissage. Ces unités cachées peuvent être sur une ou plusieurs couches.

- les unités de sortie dont chaque configuration d'activité correspond à la reconnaissance d'une forme d'entrée particulière. Les composantes du vecteur de sortie varie entre [-1,+1] ou [0,+1]. Le vecteur de sortie désirée comparé avec le vecteur de sortie contient un coefficient à 1 pour désigner la classe de la forme, les autres coefficients étant tous à -1 ou à 0.

Algorithme d'apprentissage

La rétropropagation du gradient:

L'algorithme de rétropropagation du gradient permettant l'apprentissage d'un réseau avec des unités cachées a été proposé pratiquement en même temps par [Rumelhart et al 86], [Parker 85], [Le Cun 87].

C'est un algorithme d'apprentissage supervisé. Le principe consiste à réduire une distance quadratique entre réponses désirées et calculées au moyen d'une descente de gradient dans l'espace des poids. Il repose sur le calcul de dérivées partielles. Cet algorithme utilise deux mécanismes simples : une opération locale sur les neurones du réseau cf §3.3.1 et une modification des paramètres du réseau afin de minimiser la fonction de coût. Un tel modèle est donc capable d'apprendre à reconnaître des formes déduites les unes des autres par des transformations simples de couche en couche.

Le but de l'apprentissage est de trouver la meilleure configuration de poids pour séparer les classes d'échantillons d'étiquettes différentes. La méthode d'apprentissage et l'architecture du réseau permettent de déterminer automatiquement une partition de l'espace de représentations, ce qui revient à trouver les poids des connexions qui minimisent une fonction de coût: l'erreur quadratique entre les sorties désirées et les sorties obtenues.

L'algorithme d'apprentissage comprend 4 étapes:

- la **propagation de l'activité,**
- le calcul de l'erreur par rapport à une **fonction de coût,**
- la **rétropropagation de l'erreur,**

- la **modification des poids.**

Pour un apprentissage adaptatif, l'erreur est calculée pour un exemple et rétropropagée afin de modifier immédiatement les poids. Pour un apprentissage global, on calcule l'erreur globale sur tout le corpus avant de modifier les poids.

Propagation de l'activité

L'activité des cellules de la couche d'entrée est propagée aux cellules des couches suivantes jusqu'aux cellules de sortie en utilisant la loi de propagation décrite plus haut.

Fonction de coût

La fonction de coût la plus souvent utilisée est la somme des moindres carrés (MSE) pour *"Mean Squared Error"*, c'est à dire la somme des erreurs entre la sortie désirée du réseau et la sortie obtenue après propagation des échantillons.

Soit C la fonction de coût total, C_k l'erreur pour l'échantillon k, X_k un échantillon d'entrée, f la fonction définie par le classifieur, w les paramètres du classifieur avec $f(X_k,w) = Y^s_k$ la sortie du réseau pour l'échantillon X_k, S_k la sortie désirée, NS le nombre de sorties, K le nombre d'échantillons, le coût pour l'exemple k est :

$$C_k = \sum_{n=1}^{NS} (S_{k,n} - f(X_{k,n},w))^2$$

Le coût total est :

$$C = \frac{1}{K} \sum_{k=1}^{K} \sum_{n=1}^{NS} (S_{k,n} - Y^s_{k,n})^2$$

En classification, les unités de sortie auront des valeurs désirées binaires de 1, 0 ou -1, 1. Les neurones du réseau sont reliés entre eux par des connexions auxquelles sont affectés des poids.

D'autres critères d'apprentissage peuvent être intéressants :

le critère du log de l'erreur [Austin 92] est une mesure qui présente certains avantages par rapport au MSE, en autres d'éviter le problème de poids bloqués *"weight locking"*, problème qui survient si on a des erreurs importantes mais de petits gradients avec la sigmoïde.

$$C = -\frac{1}{K} \sum_{k=1}^{K} \sum_{n=1}^{NS} \log(Y^s_{k,n} - (1-S_{k,n}))^2$$

Si les activités d'entrée du réseau ont des valeurs entre 0 et 1 et représentent une distribution de probabilité, un autre critère souvent utilisé est la *"cross-entropy"* [Hinton 87], [Fallside 91] entre la distribution désirée et la distribution de probabilité obtenue. Cette fonction de coût permet d'accélérer l'apprentissage dans certains cas. Le coût total est :

$$C = -\sum_{k=1}^{K} \left(\sum_{n=1}^{NS} S_{k,n} * \log \frac{S_{k,n}}{Y^s_{k,n}} + (1 - S_{k,n}) * \log \frac{(1 - S_{k,n})}{(1 - Y^s_{k,n})} \right)$$

Rétro-propagation de l'erreur

C la fonction de coût dépend des activités $X_{1,...,n}$ et l'activité X_k dépend des poids $W_{1,1},...W_{n,p}$, i est une couche en amont de j.

Soit $C(X_1,...,X_n)$ et $X_k(W_{1,1}, ...,W_{n,p})$, avec $i \in [1,n]$, $j \in [1,p]$

L'algorithme de rétropropagation va modifier les poids en fonction du gradient de l'erreur. Il faut donc calculer pour chaque poids Wji la dérivée $\frac{\partial C}{\partial W_{i,j}}$.

Les dérivées de fonctions composées sont de la forme:

$$\frac{\partial C}{\partial W_{i,j}} = \sum_k \frac{\partial C}{\partial X_k} \frac{\partial X_k}{\partial W_{i,j}}$$

si on développe la somme:

$$\frac{\partial C}{\partial Wi,j} = \frac{\partial C}{\partial X_1} \frac{\partial X_1}{\partial Wi,j} + \frac{\partial C}{\partial X_2} \frac{\partial X_2}{\partial Wi,j} + ...$$

$$\text{avec } \frac{\partial X_k}{\partial Wi,j} = 0 \text{ si } j \neq k$$

car les Wji sont indépendants, d'où :

$$\frac{\partial C}{\partial W_{i,j}} = \frac{\partial C}{\partial X_j} \frac{\partial X_j}{\partial W_{i,j}} = b_j \; a_i$$

$$\text{donc } b_j = \frac{\partial C}{\partial X_j} \text{ et } a_i = \frac{\partial X_j}{\partial W_{i,j}}$$

Calcul du gradient pour la dernière couche:

Soit I la couche précédent la couche J, J la dernière couche du réseau, Y_j^s est la sortie du réseau et Sj la sortie désirée, on calcule le gradient d'erreur pour le poids $W_{i,j}$.

$$b_j = \frac{\partial (Y_j^s - S_j)^2}{\partial X_j} = 2\ (Y_j^s - S_j)\ \frac{\partial Y_j^s}{\partial Xj} = 2(Y_j^s - S_j)\ \ f'(X_j)$$

et $a_i = Y_i$

$$\frac{\partial C}{\partial W_{i,j}} = 2(Y_j^s - S_j)\ \ f'(X_j)\ \ Yi$$

Calcul du gradient pour une couche intermédiaire:

Soit I la couche précédent la couche J, J n'est plus la dernière couche, soit L la couche suivant la couche J, on calcule le gradient d'erreur pour le poids $W_{i,j}$.

$$\frac{\partial C}{\partial W_{i,j}} = \frac{\partial C}{\partial X_j}\ \frac{\partial X_j}{\partial W_{i,j}} = b_j\ \ a_i$$

et $a_i = Y_i$

calcul de b_j :

$$\frac{\partial C}{\partial X_j} = \sum_l \frac{\partial C}{\partial X_l}\ \frac{\partial X_l}{\partial X_j}$$

b_l a déjà été calculé pour la couche suivante.

$$b_l = \frac{\partial C}{\partial X_l}$$

il reste à estimer:

$$\frac{\partial X_l}{\partial X_j} = \sum_m \frac{\partial X_l}{\partial Y_m}\ \frac{\partial Y_m}{\partial X_j} = \frac{\partial X_l}{\partial Y_j}\ \frac{\partial Y_j}{\partial X_j} \text{ car } \frac{\partial Y_m}{\partial X_j} = 0 \text{ si } m \neq j$$

$$\text{donc } \frac{\partial X_l}{\partial X_j} = W_{l,j} \frac{\partial Y_j}{\partial X_j} = W_{l,j}\ f'(X_j)$$

$$\text{d'où } \frac{\partial C}{\partial W_{i,j}} = \left(\sum_l (b_l\ \ Wl,j)\right)\ f'(X_j)\ Y_i$$

en résumé:

$$\frac{\partial C}{\partial W_{i,j}} = b_j\ a_i$$

$$a_i = Y_i$$

pour la couche de sortie s:

$$b_{j \in s} = 2\ f'(X_j)\ \left(Y_j^s - S_j\right)$$

pour une couche n différente de s:

$$b_{j \in n} = f'(X_j) \sum_{l \in n+1} (b_l\ \ Wl,j)$$

Equation de modification des poids:

L'algorithme de rétropropagation est une généralisation de la règle de Widrow-Hoff et s'écrit :

$$\Delta W_{i,j} = \frac{\varepsilon}{2} \frac{\partial C}{\partial W_{i,j}}$$

L'équation de modification des poids à l'instant t+1 peut s'écrire comme une équation différentielle du type Euler, cf §3.6.4.

$$Wi,j(t+1) = Wi,j(t) - \varepsilon\ \frac{\partial C}{\partial Wi,j}$$

$\frac{\partial C}{\partial Wi,j}$: gradient de l'erreur par rapport au poids Wi,j ,

ε: gain ou coefficient de modification du gradient.

Reconnaissance

Soit un exemple x, n le nombre de classes C_i, Y le vecteur de sortie du réseau de dimension n dont les valeurs sont comprises dans [0,1] pour un exemple x. Le critère de décision à la reconnaissance permet de choisir C_i tel que :

$$Y_i = (\underset{c=1,..,n}{\text{Max}} (Y_c))$$

Poids partagés

De nombreuses variantes des MLPs ont été proposées. Une des plus intéressantes pour la parole est l'utilisation de réseaux à poids partagés. On peut contraindre le réseau à avoir les mêmes poids pour plusieurs unités. Dans ce cas, on réduit les dimensions du réseau tout en ayant les mêmes transformations désirées, il en résulte souvent une augmentation du pouvoir de généralisation [Lang 88]. car le niveau d'abstraction est plus élevé. D'un point de vue algorithmique, les poids partagés sont mis à jour de façon identique (figure 3.7). Ces réseaux permettent de prendre en compte l'aspect temporel du signal et de s'affranchir d'un strict alignement temporel et sont donc mieux adaptés à la parole que des MLPs simples [Waibel 87].

Soit un ensemble de poids partagés W_{ij}, la dérivée par rapport à un poids partagé est la somme des dérivées des poids

Soit $W_{i,j} \in S$, soit ∇ le gradient de $W_{i,j}$

$$\nabla W_{ij} = (\sum_{W_{ij} \in S} \frac{\partial C}{\partial W_{ij}})$$

*Fig. 3.7 : **Poids partagés***

Aspects techniques de mise au point de l'algorithme

L'apprentissage dans un réseau est souvent un compromis entre fiabilité et coût. L'algorithme de rétropropagation du gradient d'erreur est non optimal et très lent. Il existe cependant des méthodes pour éviter les mauvais conditionnements et accélérer l'apprentissage.

Lors de l'apprentissage, il est fréquent de ne pas atteindre le minimum global et de s'enferrer dans des minima locaux. La recherche du minimum global n'est pas un problème résolu. Pour une tâche de classification réduite comme la classification de quelques formes, un bon conditionnement de l'apprentissage est nécessaire pour obtenir une convergence rapide. Lorsque la tâche de classification est plus difficile (grand nombre de sorties, base de données importante), ces techniques sont indispensables mais souvent insuffisantes et la décomposition de la tâche en sous-tâches plus faciles à traiter devient nécessaire.

Algorithme adaptatif et global

Dans une méthode globale (ou déterministe), les poids du réseau sont modifiés après avoir calculé l'erreur sur tout le corpus d'apprentissage. L'apprentissage progresse de façon plus stable et plus uniforme que lorsque les poids sont modifiés après avoir calculé l'erreur pour une forme.

Lorsque les poids sont modifiés après chaque présentation d'une forme, par exemple un segment phonétique pour une application de parole, on parle de gradient adaptatif ou stochastique. Une étude théorique de l'apprentissage stochastique est parfaitement décrite par L.Bottou dans sa thèse "*Une Approche théorique de l'Apprentissage connexionniste: Applications à la reconnaissance de la parole*" [Bottou 91]

Le choix d'utiliser un gradient stochastique est motivé par plusieurs arguments:

- les algorithmes stochastiques sont plus rapides, réduction jusqu'à 100 fois du temps de convergence,

- lorsque la base de données est importante, elle contient des formes redondantes,

- les algorithmes stochastiques permettent d'atteindre avec une probabilité plus grande les bons optimums. La recherche est moins déterministe et plus exhaustive. Un autre avantage est l'introduction de bruit dans la modification des

poids qui permet d'échapper à des minima locaux. Il est tentant de faire une analogie avec l'algorithme de recuit simulé.

Dans nos expériences, nous avons choisi d'utiliser un gradient "quasi-stochastique", c'est à dire de modifier les poids du réseau après un cycle d'apprentissage. Un cycle équivaut à une présentation d'un segment de chacun des phonèmes à classifier. Ainsi, l'ordre de présentation des phonèmes dans le réseau n'influe pas sur l'apprentissage et chaque phonème a la même fréquence d'apparition.

L'application de ce gradient "quasi-stochastique" nécessite d'utiliser avec circonspection les paramètres de modifications des poids pour ne pas tomber dans un minimum local. On a un meilleur contrôle de l'apprentissage et une meilleure performance de généralisation lorsque les paramètres de modification des poids sont faibles. Les paramètres communément utilisés sont le gain ε ainsi que d'autres facteurs comme par exemple le moment d'inertie α ou la décroissance exponentielle δ... L'équation de modification des poids devient alors :

$$W_{t+1} = W_t - \varepsilon \nabla C + \alpha (W_t - W_{t-1})$$

ou $$W_{t+1} = (1 - \varepsilon\delta) W_t - \varepsilon \nabla C$$

Les termes α et δ sont utiles pour les algorithmes déterministes mais peu utiles pour un algorithme stochastique. Il est, en effet, très compréhensible de ne pas tenir compte des modifications passées pour un algorithme stochastique. Par exemple le moment d'inertie n'est pas une indication fiable pour aller vers un optimum global lorsqu'on fait une modification des poids après présentation d'un exemple, il peut au contraire empêcher l'algorithme de sortir d'un minimum local. On n'utilisera donc que le gain ε.

Bon conditionnement de l'apprentissage

Initialisation des poids

Les valeurs initiales des poids sont un facteur important dans le choix des paramètres de ce modèle. Les poids ont été initialisés à de petites valeurs aléatoires afin que l'écart-type des sommes pondérées corresponde aux points de courbure maximum de la sigmoïde, c'est à dire que l'activité des cellules se situe dans la partie linéaire de la sigmoïde en début d'apprentissage. Si les poids sont proches de zéro au début de l'apprentissage, la convergence de l'algorithme sera très lent. En

revanche, si les poids sont choisis trop grands en début d'apprentissage, les activités seront à saturation très rapidement sans avoir appris beaucoup d'échantillons.

Les poids sont choisis aléatoirement dans un intervalle déterminé d'après ces critères. Les bornes de cet intervalle sont divisées par la racine du nombre de connexions arrivant sur une cellule, on appelle cette valeur le fan-in. L'activité est donc normalisée, quelque soit la dimension de l'entrée ou des couches cachées, la fonction sigmoïde est la même:

$$\left[\frac{-\text{poids}}{\sqrt{\text{fan-in}}}, \frac{+\text{poids}}{\sqrt{\text{fan-in}}}\right]$$

Ce choix est justifié de façon heuristique pour les expériences développées dans ce domaine [Bottou 91]. Le but est de rendre l'algorithme transparent à l'initialisation choisie. Chaque configuration initiale entraîne une nouvelle configuration des poids. Il est nécessaire d'obtenir une classification stable quelle que soit l'initialisation.

Arrêt de l'apprentissage

Pour estimer les performances du réseau et superviser l'apprentissage, on utilise plusieurs critères: le taux de reconnaissance sur le corpus d'apprentissage, l'évolution de l'erreur quadratique et souvent le taux de généralisation sur un corpus qui ne participe pas à l'apprentissage selon une technique analogue à la validation croisée. La base de donnée est en général divisée en trois parties, deux parties servant à l'apprentissage et une au test :

- une servant à la modification des poids,

- une pour tester l'erreur et le pouvoir de généralisation lors de l'apprentissage,

- et enfin une partie de test qui n'est utilisée que lorsque la phase d'apprentissage est terminée.

Fonction de décroissance du pas du gradient ε et choix d'ε

Le pas de modification des poids ε est difficile à gérer. Il ne doit pas être choisi trop grand de prime abord. Dès que l'erreur quadratique ne diminue plus pendant l'apprentissage, il faut modifier ε afin d'arriver en fin d'apprentissage à une erreur minimale et à une bonne généralisation des données. Des méthodes sont proposées dans la suite de ce paragraphe pour controler automatiquement les modifications du gain.

Contrôle automatique de la stratégie d'apprentissage et de la structure pour éviter les minima locaux

L'utilisation de modèles neuronaux implique l'ajustement de paramètres concernant l'architecture du réseau et la stratégie d'apprentissage, qu'il est difficile de définir automatiquement. Plusieurs méthodes sont possibles pour tenter de rendre plus optimal l'apprentissage du gradient d'erreur dans les réseaux.

Architecture du réseau

A. Waibel propose une recherche automatique des paramètres structurels; nombre de cellules cachées, le nombre de couches... [Waibel 89]. Ses expériences ne montrent pas d'amélioration notable par rapport aux résultats obtenus avec des choix heuristiques. Il est cependant très intéressant d'avoir des outils de ce type pour aborder la classification d'une autre base de données sans connaissance a priori. Ceci est encore plus vrai pour des utilisateurs non expérimentés.

Une méthode heuristique nécessite de connaître l'impact des paramètres sur l'apprentissage. Différents paramètres de la structure sont testés dans le chapitre 6.

L'aspect modulaire des réseaux de neurones est aussi une méthode heuristique pour aider l'algorithme à trouver un minimum global satisfaisant. En effet, le nombre de paramètres des réseaux est une des causes de lenteur et de non-optimalité de cet algorithme. Le temps de calcul augmente de façon exponentielle avec le nombre de paramètres. La recherche d'un minimum satisfaisant dans l'espace de ces paramètres devient plus difficile. Une solution est de construire des modules indépendants qui peuvent coopérer entre eux. Cependant, ces modules sont optimisés séparément et la solution globale est sous-optimale. Nous proposons des

architectures de réseaux modulaires avec différentes solutions pour optimiser globalement le système dans le chapitre 7.

Stratégie d'apprentissage

Une approche théorique d'optimisation de l'algorithme peut être faite de manière mathématique. La méthode d'Euler pour la résolution d'équations différentielles ordinaires correspond à une méthode de gradient, cf l'équation de modification des poids. C'est la plus simple et la plus stable des méthodes qui résolvent ce type d'équations pour atteindre un bon optimum local mais aussi celle qui a la plus grande erreur d'approximation.

La problématique est d'estimer un point $x_{n+1} = x + h$ connaissant x, si on utilise la dérivée première (ordre 1) au point x pour extimer x + h, on fait alors une erreur d'approximation d'ordre 2, $0(h^2)$, à l'ordre 2 l'erreur est d'ordre $O(h^3)$...

La méthode d'Euler (ordre 1) utilise :

$$f(x_{n+1}) = f(x_n) + h\, f'(x_n,y_n) + O(h^2)$$

et la méthode de Runge-Kutta d'ordre 2 :

$$f(x_{n+1}) = f(x_n) + h\, f'(x_n + \frac{1}{2} h, y_n + \frac{1}{2} h\, f'(x_n,y_n)\,) + O(h^3)$$

La figure 3.8 présente la différence de calcul entre la méthode de Runge-Kutta d'ordre 2 et la méthode d'Euler.

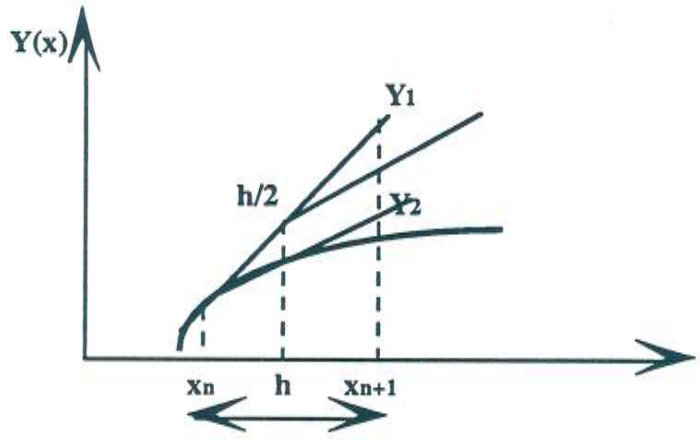

*Fig. 3.8 : **Méthode d'Euler et de Runge-Kutta.** Y_1 est calculé avec la méthode d'Euler et Y_2 avec la méthode de Runge-Kutta d'ordre 2.*

La méthode de Runge-Kutta permet d'estimer x_{n+1} avec plus de précision. Elle est aussi appelée "*midpoint method*". A l'ordre 4, elle est plus performante. D'autres variantes plus sophistiquées ont été proposées par [Abramowitch 64], [Gear 71] à partir de cette méthode de Runge-Kutta. Les méthodes mathématiques de convergence du gradient ne conduisent pas toujours à des algorithmes efficaces en pratique.

Contrôle du gain

Une autre modélisation permettant d'optimiser la recherche d'un minimum satisfaisant est le contrôle du pas du gradient, aussi appelé gain.

Une heuristique simple pour diminuer le gain est une procédure permettant de maintenir le gain maximum tant que l'apprentissage reste stable [Franzini 88]. L'angle θ entre le gradient au temps t et le gradient à temps t-1 contrôle cette heuristique. Soit C le coût d'erreur, $W_{i,j}$ un poids, $\cos\theta$ s'écrit :

$$\cos\theta = \frac{\sum_{i,j} (d_{i,j}(t-1)d_{i,j}(t))}{\sqrt{(\sum_{i,j} d_{i,j}(t-1)^2)(\sum_{i,j} d_{i,j}(t)^2)}} \text{ avec } d_{i,j} = \frac{\partial C}{\partial W_{i,j}}$$

On peut dynamiquement ajuster les gains en utilisant la mesure globale du cosinus de l'angle θ, soit β le facteur d'augmentation du gain:

$$\varepsilon(t) = \varepsilon(t-1)\,\beta^{\frac{\cos\theta + 1}{2}}$$

β vaut typiquement 1.005, il accroît le gain lorsque $\cos\theta$ est proche de 1. Le facteur β est utile si le nombre de données utilisées pour calculer le gradient est suffisamment grand et si l'ensemble de données ne varie pas trop entre le calcul du gradient à l'instant t - 1 et t. Cette méthode d'ajustement des gains est intéressante pour un algorithme non stochastique.

L'utilisation d'une approximation de la matrice Hessienne pour contrôler les gains permet d'utiliser des gains différents pour chaque unité, on parle alors de gain matriciel. La méthode de quasi-Newton [Bottou 91] ne considère que la diagonale du Hessien, elle permet un calcul plus rapide des dérivées secondes, elle a aussi l'avantage de pouvoir être utilisée avec un gradient stochastique. L'équation de modification des poids de la méthode de quasi-Newton est :

$$\Delta w_{ij} = -\varepsilon \frac{\frac{\partial C}{\partial w_{ij}}}{\left|\frac{\partial^2 C}{\partial w^2_{ij}}\right| + \mu}$$

Le paramètre μ permet d'éviter les poids infinis lorsque la dérivée seconde est proche de zéro, C est la fonction de coût à minimiser, W_{ij} un poids et ε le gain.

Ce contrôle n'a pas été appliqué dans nos expériences. Il oblige à calculer les dérivées secondes en plus des dérivées premières, ce qui représente une perte de temps et ne garantit pas une meilleure généralisation [Bottou 91]. En revanche, il ne nécessite plus de surveillance de l'évolution de l'algorithme et chaque poids à un gain particulier qui peut aller en augmentant ou diminuant au cours de l'apprentissage.

Pour éviter des calculs trop onéreux, on peut cependant utiliser une approximation des dérivées secondes :

$$\Delta W_{i,j}(t) = \mathrm{Min}\left\{ \mu, -\varepsilon \frac{\frac{\partial C(t)}{\partial W_{i,j}}}{\frac{\partial C(t-1)}{\partial W_{i,j}} - \frac{\partial C(t)}{\partial W_{i,j}}} \Delta W_{i,j}(t-1) \right\}$$

le paramètre μ sert à éviter les $\Delta W_{i,j}(t)$ infinis lorsque $\frac{\partial C(t-1)}{\partial W_{i,j}} - \frac{\partial C(t)}{\partial W_{i,j}}$ est proche de 0.

Une alternative pour contrôler automatiquement les gains est de contrôler l'évolution de l'erreur globale au cours de l'apprentissage. Elle ne donne pas d'informations précises sur les poids mais permet de diminuer les gains automatiquement, cette stratégie est décrite dans le chapitre 6 avec de nombreux tests de stabilité.

3.5 Codage de séquences: MLP avec contraintes sur les connexions

Le codage de séquences est indispensable à certaines applications. La parole, par exemple, nécessite de part son aspect dynamique une gestion des séquences de

vecteurs. L'énumération des réseaux codant un aspect temporel n'est pas exhaustive, nous ne parlerons que de réseaux qui ont été appliqués à la parole.

3.5.1 Les réseaux récurrents à couche: RNN

Les systèmes neuronaux récurrents sont des dérivés du perceptron multicouche. L'idée générale de ces réseaux est de coder des séquences de vecteurs. Ils accumulent des informations sur les formes d'entrée en combinant leurs représentations internes avec les données qui arrivent à chaque instant. Il existe des réseaux récurrents avec des boucles de récurrence sur différentes couches : le MLP avec rétroaction de contexte (figure 3.9) [Elman 88], le MLP avec rétroaction des unités de plan (figure 3.10) [Jordan 86].

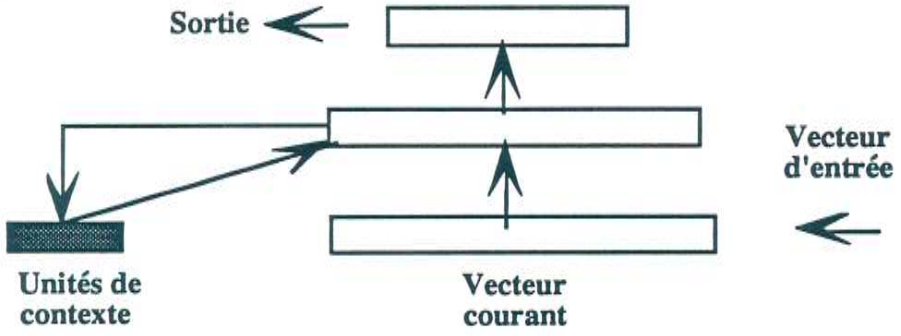

Fig. 3.9 : le MLP avec rétroaction des unités cachés de type Elman.

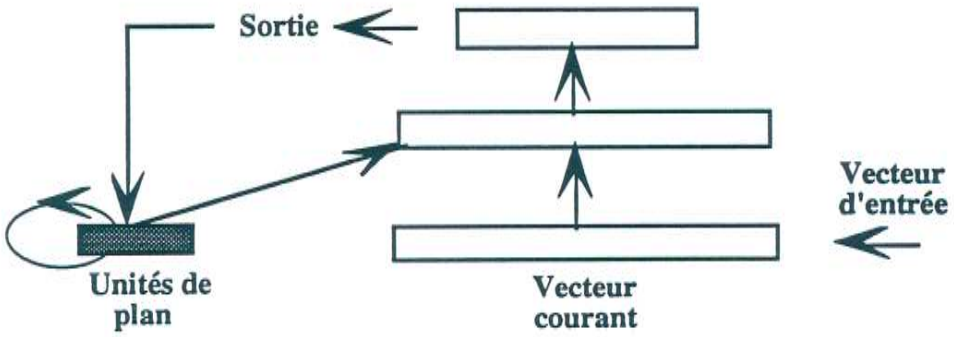

Fig. 3.10 : Le MLP avec rétroaction des unités de sortie de type Jordan.

Lorsque les séquences ne sont pas trop longues, on peut utiliser l'algorithme de retro-propagation du gradient standard en dépliant le réseau dans le temps [Hinton al 86]. Pour des contextes plus longs, différents algorithmes d'apprentissage ont été proposés [Watrous 87] [Kuhn 90].

Le TDNN [Waibel 87] [Waibel 88] est un "réseau récurrent déplié", il comporte une récurrence temporelle explicite sur chacune de ses couches et un algorithme plus simple à mettre en oeuvre qui consite à utiliser la rétropropagation du gradient pour des poids partagés.

3.5.2 Architecture TDNN

A. Waibel [Waibel 87] propose un réseau MLP avec des matrices de poids partagées pour mieux prendre en compte l'aspect temporel de la parole. Le TDNN "Time Delay Neural Network" (figure 3.11) code un contexte sur chaque couche. L'architecture générale comporte une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées.

Les réseaux proposés par Waibel pour classifier des segments phonétiques comprennent une couche d'entrée, deux couches cachées et une couche de sortie soit des réseaux utilisant trois matrices de connexions. On parle alors d'architecture de réseaux à trois couches. La matrice de connexions entre l'avant-dernière couche et la dernière couche est d'un type différent des autres; on calcule les scores de la dernière couche en sommant l'activité de toutes les cellules de l'avant-dernière couche correspondant aux mêmes phonèmes sur l'axe des temps, le score le plus élevé correspond au phonème reconnu. La matrice de connexions entre ces deux couches est totale. Les tailles des fenêtres d'analyse sont respectivement 3 pour la première couche et 5 pour la deuxième. La fenêtre d'intégration temporelle est fixée à 15 trames dont le débit est de 10ms.

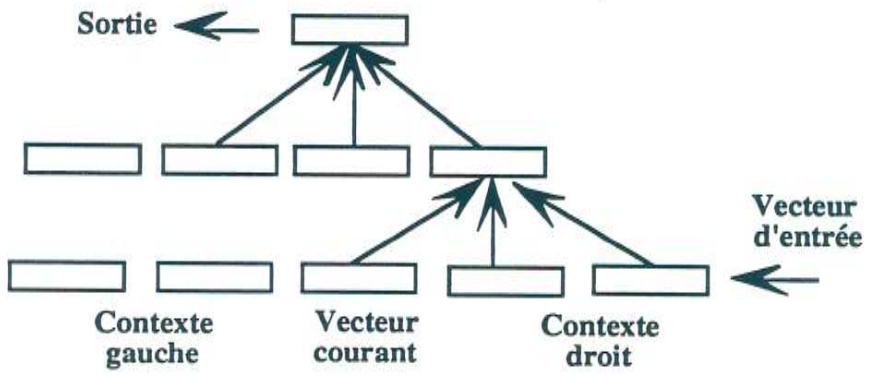

Fig. 3.11 : TDNN

L'architecture choisie dans nos applications comporte une couche cachée, ce qui semble suffisant pour séparer des nuages de données complexes tels que des classes

de phonèmes, cf. chapitre 6. La matrice de connexions décrite ci-dessus entre les deux dernières couches n'est pas créée, l'architecture ne comporte alors que deux couches. La dernière couche d'un tel réseau est une suite de vecteurs que l'on peut associer à un treillis centisecondes d'étiquettes phonétiques. On appelle ces étiquettes phonétiques, des "quasi-phonèmes". Une séquence de quasi-phonèmes représente un phonème.

TDNN et MLP à contexte

Les réseaux à contraintes sur les connexions permettent de coder de façon compacte les poids de connexions et offrent un potentiel d'abstraction et de généralisation plus grand. Les poids sont partagés par plusieurs cellules cf §3.5.4. Le TDNN code un contexte; les potentiels d'action qui parviennent à un neurone proviennent de plusieurs trames aux instants t, t+1, t+2 .. selon la taille du contexte pris en compte. A la différence d'un perceptron multicouche, les connexions entre les couches d'un TDNN sont partielles, les matrices de poids sont reproduites à l'identique le long de l'axe temporel afin d'apprendre les caractéristiques acoustiques et leurs relations temporelles indépendamment de la position dans le temps. Ce mécanisme permet en principe une invariance par translation sur le signal de parole.

Les MLPs à décalage temporel ou à contexte (figure 3.12) sont des réseaux très simples qui codent directement toute la séquence d'entrée (un segment phonétique ou un mot) en contexte et n'utilisent pas le mécanisme de poids partagés Les MLP simples ou à décalage sont des réseaux entièrement connectés. Coder plusieurs trames en entrée ajoute de nombreuses connexions dans l'architecture du réseau.

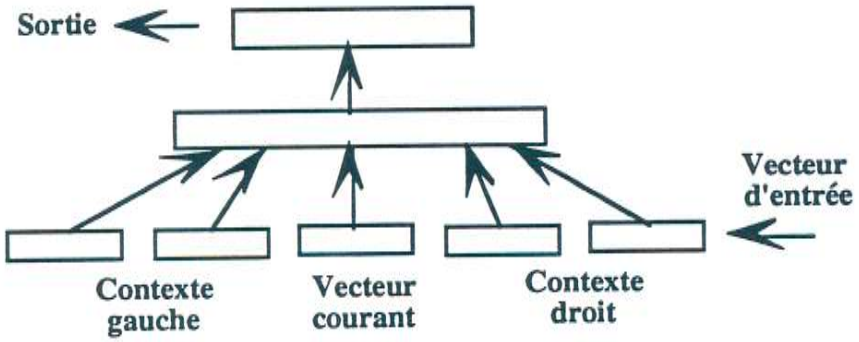

Fig. 3.12 : MLP à décalage ou à contexte

Les modèles utilisant un contexte temporel sont plus performants qu'un simple perceptron multicouche. Les TDNNs, les MLPs à contexte gèrent une intégration temporelle correspondant à un nombre de trames fixe. Elle ne permet cependant pas

de prendre en compte les distorsions temporelles et de résoudre des problèmes de parole continue.

TDNN et RNN

Les réseaux récurrents sont mieux adaptés aux problèmes de gestion temporelle que les TDNNs mais nécessitent malgré tout l'utilisation d'algorithme d'alignement temporel pour intégrer les séquences de sortie des réseaux. Des résultats intéressants avec des systèmes hybrides RNNs/Viterbi ont été récemment obtenus en reconnaissance de la parole [Robinson 92] (cf §4.4.1) [Franzini et Waibel 91] (cf §4.4.2). D'un point de vue théorique, Bridle [Bridle 89] a mis en évidence l'équivalence d'un certain type de réseau récurrent appelé "*alphanet*" avec un système à base de modèles markoviens. Malgré les différences de structures, il montre que le même algorithme peut être utilisé (cf §4.3.2). Ces réseaux sont théoriquement capables de gérer des distorsions temporelles. En pratique, les RNNs posent cependant quelques problèmes. Bengio [Bengio 91] soulève le problème de l'instabilité de l'apprentissage et le risque élévé de s'enferrer dans des minimum locaux. Franzini et Waibel [Franzini, Waibel 91] comparent les performances obtenues avec des RNNs et des TDNNs et montrent que les RNNs offrent des taux de reconnaissance un peu plus élevés avec cependant des temps de calcul très importants.

Les TDNNs sont en revanche plus facile à mettre en oeuvre et moins coûteux en temps de calcul que les RNNs. Ils permettent aussi une certaine gestion temporelle et ont été appliqués avec succès à de nombreuses tâches. A l'heure actuelle, les réseaux récurrents manquent encore d'algorithmes d'apprentissage efficaces pour pouvoir être utilisés avec des temps de calcul raisonnables mais sont certainement une voie de recherche très prometteuse.

3.5.3 Réseaux prédictifs

Ces réseaux ne sont plus entraînés pour générer une classification phonétique mais pour faire de la prédiction du signal [Iso 90], [Levin 90]. Un modèle neuronal prédictif utilise une séquence de MLP comme prédicteur non-linéaire de chaque classe. Un exemple de réseaux prédictifs est donné en figure 3.13. Chaque classe est modélisée par un réseau, celui-ci apprend la correlation temporelle entre les trames des segments d'une classe. Ces réseaux ne resolvent pas le problème de l'intégration temporelle, un algorithme d'alignement temporel de type DTW ou Viterbi est en général utilisé pour gérer les séquences de phonèmes.

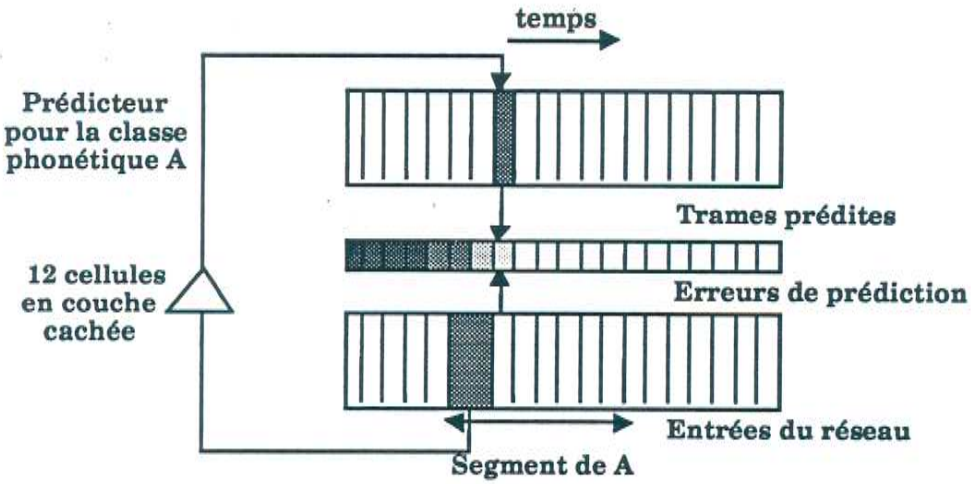

*Fig. 3.13 : **Réseau prédictif pour la classe A.** Dans cet exemple, on a 2 trames de contexte en entrée du réseau.*

3.6 Conclusion

Le caractère stochastique et séquentiel de la production de la parole rend sa reconnaissance automatique particulièrement difficile. Grâce à leur dynamique implicite, les réseaux de neurones récurrents, les TDNNs, les réseaux prédictifs sont capables d'intégrer des informations temporelles et dans certaines limites de traiter les déformations temporelles. Cependant, l'apprentissage et la reconnaissance de parole continue sont actuellement impossibles en pratique en utilisant seulement des réseaux de neurones. Pour l'apprentissage, les réseaux nécessitent des données segmentées, on utilise souvent une segmentation issue des modèles markoviens. Pour la reconnaissance, les réseaux de neurones ne comportent pas de mécanisme efficace permettant d'intégrer les séquences de sorties lorsque ces sorties ne sont pas des mots isolés. La reconnaissance de la parole continue notamment ne peut être résolue seulement avec des réseaux d'où la nécessité de leur intégrer un algorithme d'alignement temporel. L'intérêt des systèmes hybrides est justement de faire coopérer des approches complémentaires permettant de pallier ces problèmes.

Des problèmes de classification phonétique sont explorés avec des réseaux MLP, TDNN et LVQ2 dans le chapitre 5. Des procédés d'alignement temporel sont utilisés sur les sorties des réseaux, comme l'algorithme de programmation dynamique [Driancourt 91], ou l'algorithme de Viterbi issu des HMMs [Bourlard 91], afin d'intégrer les séquences de phonèmes lors de la reconnaissance. Un état de l'art des systèmes hybrides est décrit dans le chapitre 4.

4. Etat de l'art: les systèmes hybrides en reconnaissance de la parole

Les meilleurs systèmes actuels de reconnaissance de la parole sont les modèles markoviens cachés. Ils n'ont cessé d'être améliorés depuis une quinzaine d'années. Depuis 1986, date de présentation du perceptron multicouche [Rumelhart 86] et de l'algorithme de rétropropagation du gradient d'erreur, les modèles neuronaux ont suscité un regain d'intérêt auprès des chercheurs. Ces méthodes offrent de nombreux avantages que n'ont pas les HMMs: pouvoir de discrimination, peu d'hypothèses sur les données... et ont été mis directement en compétition avec les modèles markoviens. Pour des tâches réduites, les réseaux de neurones se comportent très bien et rivalisent avec les HMMs. Pour des tâches plus complexes de parole continue, ils sont très mal adaptés. A l'heure actuelle, les limites des réseaux sont relativement bien connues et les recherches se sont portées vers des coopérations des deux techniques markoviennes et neuronales afin de tirer bénéfice de leurs avantages respectifs.

Nous présenterons tout d'abord dans ce chapitre une synthèse sur les modèles markoviens et les réseaux à rétroprogation du gradient d'erreur pour la reconnaissance de la parole, cf §4.1 et 4.2. Les avantages et inconvénients de chaque méthode sont résumés ainsi que les axes de recherche actuels. Les deux inconvénients majeurs des réseaux sont étudiés dans les chapitres suivants. La classification statique et l'aspect gestion temporelle font l'objet du chapitre 5, l'aspect temps et qualité d'apprentissage de grandes bases de données est décrit dans le chapitre 6. Les chapitres 7 et 8 de cette étude proposent des solutions à ces

problèmes. Le chapitre 7 offre une solution à l'apprentissage de grandes bases de données : la modularité. Les méthodes hybrides décrites dans le chapitre 8 sont une solution pour intégrer un mécanisme d'alignement temporel aux réseaux.

Les recherches sur les systèmes hybrides suivent plusieurs tendances. La première tendance est de savoir si les modèles neuronaux peuvent remplacer les systèmes actuels comme les HMMs et de comprendre les différences théoriques entre ces deux techniques. Nous aborderons ce point dans le paragraphe 4.3. La deuxième tendance est de développer des structures hybrides neuronales et markoviennes. Nous présenterons un état de l'art des recherches actuelles sur les systèmes hybrides dans le paragraphe 4.4.

4.1 Modèles markoviens

Le rapport de [Rabiner, Huang 86] fait référence dans le domaine, il décrit les aspects théoriques et pratiques des HMMs de façon très claire. Les améliorations des performances des systèmes de reconnaissance de la parole, tant au niveau de la taille des vocabulaires que des applications multi-locuteurs, ont été obtenues en utilisant des modèles markoviens de plus en plus sophistiqués. Citons quelques systèmes de référence en reconnaissance de parole continue multilocuteur avec des modèles markoviens cachés, le système Byblos de "*BBN Systems and Technologies*" [Kubala 88] et Sphinx de CMU "*Carnegie Mellon University*" [Lee 88]... Une vingtaine de laboratoires sont entrées dans la course aux meilleures performances. Ils ont en majorité américains dont BBN, CMU, Bell-Labs, Dragon systems, SRI,... mais on y trouve aussi quelques européens dont Philips (Allemagne), Cambridge (GB), LIMSI (France)... Les systèmes sont testés sur des corpus communs qui permettent de comparer les différents résultats. Les plus importantes évaluations ont été menées sur la tâche "*DARPA: Defense Advanced Research Projects Agency - Ressource Management*" qui regroupe le corpus TI-DIGITS, TIMIT, DARPA RM "*speaker dependent*", DARPA RM "*speaker independant*" et plus récemment sur ATIS, corpus de parole plus spontanée qui correspond à l'interrogation d'une base de données de réservation aérienne.

Nous résumons dans ce paragraphe les qualités et défauts des HMMs et présentons les axes de recherche actuelles.

4.1.1 Qualités et défauts

Ce tableau résume les principales qualités et défauts des modèles markoviens cachés.

+ Algorithme performant d'alignement temporel,
+ Intégration de tous les niveaux: décodage acoustique, accès au lexique, modèle de langage,
+ Modèles représentant des unités différentes: phonème, diphone, triphone, mot,
+ Apprentissage de données non segmentées préalablement,
+ Possibilité d'ajouter des classes,
+ Informations probabilistes en sortie,
- Manque de capacité de discrimination entre les modèles,
- Hypothèse restrictive sur la forme des distributions de classes,
- Hypothèse restrictive d'indépendance des trames.

Aspects positifs

Il existe de multiples aspects positifs à l'utilisation des modèles markoviens pour la parole, notamment : la capacité de prendre en compte l'aspect séquentiel de la parole, de pouvoir exprimer une phrase en séquence de HMMs, de ne pas nécessiter de segmentation. Les deux principales qualités qui font défauts à l'heure actuelle aux réseaux de neurones sont :

- une intégration de tous les niveaux: décodage acoustique, accès au lexique, modèle de langage,

- et un algorithme d'alignement temporel très performant.

Limites

Malgré leur succès, ces modèles présentent quelques faiblesses qu'il est complexe d'améliorer en utilisant le même formalisme mathématique. Les algorithmes d'apprentissage les plus employés utilisent un critère non discriminant fondé sur le *"Maximum Likelihood Estimation"* (MLE). Les modèles générés sont optimisés

indépendamment les uns des autres et ne contiennent pas d'informations discriminantes.

La définition même des modèles markoviens présente certaines limites L'hypothèse majeure est faite sur la forme des densités de probabilités. La topologie des modèles est aussi choisie a priori dans les modèles d'ordre 1 qui sont ceux utilisé pour la parole, la probabilité d'un état ne dépend que de l'état précédent et la dépendance entre deux observations à deux instants donnés est exprimée par la dépendance entre les états correspondants. Elles ont pour conséquence de rendre les modèles peu robustes si les données sont assez variables.

4.1.2 Axes de recherche actuels

Nous nous sommes intéressés aux axes de recherche visant à pallier les inconvénients des HMMs que nous présentons comme des "avantages" des réseaux de neurones, c'est-à-dire principalement le pouvoir de discrimination et le caractère moins restrictif des hypothèses sur la formes des densités ainsi que la gestion d'information contextuelle.

Les capacités d'algorithmes d'apprentissage utilisant des critères d'optimisation globale comme le *"Maximum Mutual Information"* [Brown 87] ou le *"corrective training"* [Niles 92] permettant d'apprendre des informations discrimantes sont explorées depuis quelques années. On les utilise souvent après une phase d'apprentissage avec le critère de MLE. Ces procédures sont en général peu aisées à mettre en oeuvre et n'apportent pas toujours les résultats escomptés [Chow 90],[Niles 92].

L'hypothèse sur la forme des densités est souvent très loin de la réalité des formes. Ce défaut est amoindri lorsqu'on utilise des mélanges de densités de probabilités ou des densités multimodales [Rabiner 86]. Ces modélisations permettent de pondérer les hypothèses restrictives sur les formes des densités. Cependant, elles nécessitent d'estimer plus de paramètres et donc d'avoir de plus grandes bases de données. Les modèles continus peuvent être représentés par plusieurs gaussiennes par modèle, on parle alors de *"mixture densities"*. Ces gaussiennes sont estimées automatiquement avec un algorithme de *"split"*.

On obtient les mêmes performances avec des HMMs codant des unités contextuelles de type diphones ou triphones et des HMMs utilisant suffisamment de *"mixture densities"* [Ney 90]. Les mélanges de densités permettent de représenter la variabilité contextuelle.

Le critère d'apprentissage bayésien "*Maximum A Posteriori*" [Gauvain 92] apporte de meilleures performances que le critère de MLE mais nécessite l'estimation de beaucoup plus de paramètres. C'est une méthode de lissage des paramètres des modèles markoviens qui permet aussi de diminuer la restriction des hypothèses sur les fonctions de densités et d'obtenir une meilleure résistance à la variabilité des données.

Des modèles segmentaux sont testés afin de prendre en compte l'aspect contextuel des trames [Russel 89], [Gauvain 92]. Pour réduire le nombre de coefficients sans réduire l'information, des prétraitements intégrant le contexte: dérivées premières et dérivées secondes ou coefficients issus de LDA "*Linear Discriminant Analysis*" sont aussi utilisés [Haeb-Umbach 92].

4.2 Réseaux de neurones

Le rapport de [Lippmann 89] sert de référence dans le domaine, il décrit les différents types de réseaux et montrent leurs avantages et inconvénients. Depuis quelques années, des modèles neuronaux, en particulier les réseaux utilisant la rétropropagation du gradient d'erreur, RNN, TDNN et MLP, ont été appliqués à des problèmes de reconnaissance de la parole avec un certain succès. Les réseaux RBFs ont permis d'obtenir des résultats comparables à ces réseaux avec des temps d'apprentissage souvent plus rapides. L'algorithme de LVQ2 offre aussi des résultats intéressants. Les réseaux n'ont pas les mêmes critères d'apprentissage, les mêmes structures, ou les mêmes qualités et défauts que les modèles markoviens. Les deux approches apparaissent plutôt complémentaires.

4.2.1 Qualités et défauts

Ce tableau résume les principales qualités et défauts des réseaux de neurones utilisant l'algorithme de rétropropagation du gradient d'erreur.

+ nature discriminante inhérente aux critères d'apprentissage, *+ pas d'hypothèse sur la forme des densités,* *+ prise en compte du contexte des trames,* *+ robustesse aux données altérées,* *+ capacité d'adaptation à de nouvelles données,* *+ possibilités de structures non linéaires,* *+ haut degré de parallélisme des architectures.* *- pas de mécanisme d'alignement temporel,*
- peu d'algorithmes d'apprentissage, *- apprentissage très lent,* *- nécessité de connaissances a priori pour construire une structure adéquate à la tâche de classification,* *- nécessité de données segmentées pour l'apprentissage* *- les formes d'une même classe sont souvent représentées par une seule sortie (c'est le cas du MLP, TDNN...)*

Aspects positifs

Les réseaux ont des qualités différentes des modèles de reconnaissance des formes statistiques paramétriques classiques; des qualités de discrimination dues aux critères utilisés à l'apprentissage, des qualités de robustesse à la variabilité des données, d'adaptabilité, de traitement direct des données réelles... De plus, leur architecture massivement parallèle permet d'envisager de nombreuses applications en temps réel sur des machines appropriées. La structure des réseaux (nombre de couches, nombre de cellules) correspond à l'hypothèse faite a priori sur les données, elle est beacoup moins restrictive que dans le cas des HMMs et permet une meilleure résistance aux données "bruitées". Mais les réseaux nécessitent plus de temps d'apprentissage que les HMMs.

Les capacités d'un MLP non linéaire sont aussi la compression de données telle que l'analyse en composantes principales Blanchet 89], [Bourlard 87] et dans le cas linéaire, l'analyse discriminante [Gallinari 88].

Pour des classifications de formes statiques, les réseaux ont montré des performances supérieures à celles des meilleurs modèles classiques

[Huang, Lippmann 87], [Nakagawa 90], [Gallinari 91]. Le principal intérêt des réseaux est leur robustesse au bruit et leur aspect discriminant. Nous citons quelques résultats intéressants obtenus avec différents réseaux en parole.

Pour des classifications de mots isolés, sur les bases de données en américain comme le E-set ou les Ti-digits, de très bonnes performances ont été obtenues avec des TDNNs [Waibel 88], [Haffner 89], [Bottou 89]. Des résultats intéressants ont aussi été obtenus avec des réseaux récurrents en reconnaissance phonétique [Kuhn 90] [Bridle 90]. Avec des données bruitées sur des tests relativement réduits, les réseaux de neurones, TDNN et LVQ2 obtiennent de meilleurs résultats que des méthodes classiques [De Bollivier 92].

Les réseaux RBFs, réseaux à unités gaussiennes, cf §3.1.1, permettent d'obtenir des performances égales à celles des réseaux à unités sigmoïdes pour moins de temps d'apprentissage [Huang 89], [Niles 90], [Boiteau 92]. Les performances des RBFs sont cependant très dépendantes d'une bonne initialisation, obtenue par exemple avec un algorithme de K-moyennes ou de LVQ2.

Limites

Leurs principales limites commencent à être aussi bien répertoriées :

- Les réseaux ont en général des temps d'apprentissage très longs et un algorithme assez sensible à un bon conditionnement des paramètres du réseau: structure, gain de modification des poids... Il faut souvent une connaissance d'expert pour définir les structures des réseaux et pour gérer les paramètres d'apprentissage.

- La nécessité de données segmentées pour la reconnaissance de la parole est aussi un point négatif des réseaux par rapport aux HMMs.

- Les systèmes de reconnaissance de la parole nécessitent un alignement temporel des formes afin d'éliminer dans une certaine mesure les problèmes de distorsions temporelles. Les réseaux de type TDNN ou réseaux récurrents n'offrent pas de mécanisme assez robuste au problème temporel. Ces réseaux font une classification "temps/fréquence" des formes en utilisant une représentation "spatiale" des formes.

4.2.2 Axes de recherche actuels

Le problème de l'alignement temporel est sans doute le plus délicat. Les réseaux hybrides sont une solution à ce problème. Il existe cependant des modèles neuronaux permettant un codage "temporel et spatial" comme par exemple la propagation guidée [Béroule 85], [Escande 92] dont les principes sont intéressants mais qui n'ont pas encore été éprouvés sur des bases de données suffisamment grandes pour permettre une évaluation de leurs capacités.

La dynamique temporelle est prise en compte par certains réseaux autement que par un contexte fixe. Citons [Rander, Unnikrishnan 92] qui proposent une gestion dynamique des délais temporels par des gaussiennes. La structure HCNN pour "*Hidden Control Neural Network*" [Levin 90] est une méthode qui combine une prédiction neuronale non linéaire des réseaux classiques avec une modélisation temporelle. Ce modèle gère la variabilité temporelle en ajoutant dans la structure du réseau un contrôle des entrées qui modifie l'organisation du réseau. Cette structure a un avantage qui est de réduire le nombre de paramètres libres dans le système, et théoriquement d'améliorer la généralisation. Un algorithme de Viterbi reconnaît ensuite les séquences de sortie du réseau. Elle a été testée avec succès sur des mots isolés [Levin 90]. En parole continue sur un vocabulaire de 400 mots [Tebelski, Waibel 91], elle doit être utilisée avec beaucoup de prudence et a du mal à gérer beaucoup de formes.

Il est aussi très intéressant de pouvoir définir les structures des réseaux sans connaissance a priori pour aborder simplement la classification de nouvelles bases de données. Citons [Bodenhausen,Waibel 89] qui propose un contrôle automatique des paramètres structurels d'un réseau comme le nombre de cellules cachées, le nombre de couches...

Les améliorations du temps d'apprentissage sont souvent réalisées au détriment de la qualité de cet apprentissage. Afin de pouvoir gérer des bases de données de taille importante sans aller au détriment de la qualité de l'apprentissage, on utilise des architectures modulaires. Citons [Jacobs 91] qui permet de construire automatiquement une architecture modulaire cf chapitre 7.

Il existe aussi quelques recherches sur la segmentation automatique avec des modèles neuronaux. Des expériences avec des réseaux prédictifs permettant une segmentation non supervisée sur la base de données TIMIT ont permis d'obtenir d'assez bons résultats [Doutriaux 89]. La segmentation de la parole à partir de prédiction est fondée sur la supposition que celle-ci est possible à l'intérieur d'un

segment et impossible aux frontières. On détecte les frontières des segments car l'erreur est plus importante et l'activité des cellules instable.

Les unités gaussiennes permettent des représentations locales tandis que les unités sigmoïdes sont globales. Les unités sigmoïdes définissent des hyperplans qui partagent l'espace des données, elles sont donc utiles pour des représentations globales. Les unités locales des RBFs permettent en autres tout comme pour l'algorithme de LVQ de représenter les classes par plusieurs densités ou plusieurs références ou nuages de références. L'utilisation conjointe des deux sortes d'unités est une voie de recherche intéressante [Bengio 91]. Des modèles hybrides TDNN-LVQ-DTW utilisant des unités globales et locales permettent d'obtenir de très bonnes performances sur des mots isolés [Driancourt, Gallinari 92]. Dans ce système, L'algorithme de LVQ permet de s'affranchir de la restriction d'une sortie par classe du réseau global.

4.3 Relations entre HMM et NN

4.3.1 Comparaison entre HMM et NN

De nombreuses comparaisons ont été faites sur des mots isolés entre les HMMs et les NNs. Il est difficile d'avoir une opinion tranchée en faveur de l'une ou de l'autre des méthodes. Les résultats obtenus dépendent de la base de donnée et des HMMs utilisés. Les HMMs testés peuvent être très différents: modèles discrets, continus ou encore avec des mélanges de densités continues... La base de données peut être bien segmentée ou non, plus ou moins bruitée, de taille suffisante ou non pour entraîner des HMMs avec des mélanges de densités...

Nous avons pris comme référence l'article de synthèse issu de la coopération de plusieurs laboratoires, Cambridge, RSRE et LRI dans le projet européen BRAIN [Bedworth 89] et comparant des classifieurs neuronaux et classiques sur la base de données en américain du "*E-set*". Cette base de données contient des paires-minimales : B, C, D, E, G, P, T, V, Z. Elle comporte 104 locuteurs séparés pour cette expérience en deux corpus, de test et d'apprentissage d'environ 400 mots chacun. La comparaison entre des HMMs avec des modèles discrets (23% d'erreur pour 50 locuteurs testés) et des NNs (MLP (17%), TDNN (20%), RBF (17%)) est en faveur des réseaux. Cependant les HMMs utilisant des modèles continus avec une densité par état sont plus performants (seulement 14% d'erreur) que les réseaux testés pour cette base de données. Sur la même base de données mais avec seulement 16

locuteurs testés, nous présentons des résultats obtenus par le MIT avec des systèmes hybrides HMM-NN, cf 4.4.3.

4.3.2 Equivalences théoriques

On évoquera deux résultats importants des recherches théoriques sur les NNs et HMMs. Tout d'abord, le fait de considérer les sorties des réseaux comme des probabilités a posteriori qui peuvent être utilisées dans le formalisme HMM et ensuite l'équivalence possible entre les algorithmes d'apprentissage des HMMs et des NNs.

Bourlard et Wellekens [Bourlard 89] ont été les premiers à montrer que les MLPs pouvaient générer en sortie des probabilités locales discriminantes qui peuvent remplacer les probabilités d'emission des HMMs, cf §2.4.5.1. Il suffit alors de calculer les probabilités de transition dans le HMM pour modéliser l'aspect temporel.

Les travaux de Bridle [Bridle 90] montrent l'équivalence entre les modèles markoviens et les réseaux de neurones via un réseau récurrent particulier, l'alphanet. Bridle décrit l'algorithme de "*forward-backward*" des HMMs comme un cas spécifique de rétropropagation du gradient d'erreur dans un réseau récurrent. Des équivalences théoriques entre les deux algorithmes sont aussi proposées par [Bottou 91] et [Young 90]. La démonstration suivante est issue d'un article de synthèse de [Fallside 92].

Pour un modèle HMM, la séquence d'états $(X_1, ...X_T)$ est controlée par les probabilités de transitions $a_{i,j} = P(X_t=j/X_{t-1}=i)$. et l'état courant est déterminé par les propriétés statistiques $b_j(y_t) = P(Y_t=y_t/X_t=j)$ pour une séquence $y_{1T}= y_1y_2...y_T$ et une sortie du modèle Y_t.

On peut calculer la vraisemblance de toutes les observations (cf chapitre 2 pour une description détaillée des modèles markoviens) $P(Y_{1T}= y_{1T}/\text{modèle})$.

D'après l'algorithme forward :

$$\alpha_{jt} = P(Y_{1t}= y_{1t} \text{ et } X_t=j/\text{modèle})$$

$$\alpha_{ij} = b_j(y_t)\sum_i a_{i,j}\alpha_{it-1} \quad (1)$$

Si on a un modèle par mot, soit w_i un mot, la probabilité a posteriori d'obtenir ce mot d'après la règle de Bayes est :

$$P(W=w_i/Y_{1T}) = \frac{P(Y_{1T}/W=w_i)P(W=w_i)}{\sum_i P(Y_{1T}/W=w_i)P(W=w_i)}$$

Soit N le dernier état du modèle, si les modèles sont équiprobables alors on peut écrire la probabilité a posteriori du mot comme étant :

$$P(W=w_i/Y_{1T}) = \frac{P(Y_{1T}/W=w_i)}{\sum_i P(Y_{1T}/W=w_i)} = \frac{\alpha_{N_{wi}T}}{\sum_i \alpha_{N_{wi}T}}$$

Pour l'alphanet, on retrouve dans la propagation de l'activité à travers le réseau le calcul des α donné par l'équation (1). Les poids du réseau sont les probabilités de transitions a_{ij} et les bj sont calculés à partir des observations, les α_{ij} sont les états internes ou cellules cachées, calculés par la récurrence cf fig. 4.1.

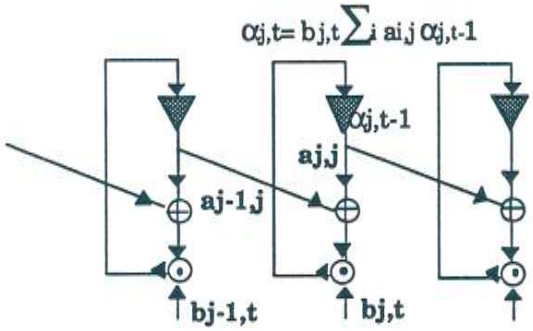

*Fig. 4.1 : **Mécanisme de récursion dans l'alphanet.***

Si on considère maintenant la rétropropagation du gradient d'erreur, on calcule l'erreur globale du réseau. Soit la sortie obtenue $Q_w = P(W=w_i/Y_{1T})$ et la sortie désirée T_w, qui prend la valeur 1 pour la bonne classe, 0 sinon, l'erreur totale est :

$$E_{tot} = \sum_i E(Q_{w_i}, T_{w_i})$$

Bridle a montré comment calculer le gradient d'erreur E_{tot} par rapport à a_{ij} et E_{tot} par rapport à m_j, m_j étant la moyenne de la densité de probabilité modélisée par exemple par une gaussienne.

L'équation de rétropropagation équivaut alors à :

$$\frac{\partial \alpha_{NwT}}{\partial \alpha_{i,t-1}} = \sum_j a_{i,j} b_j(y_t) \frac{\partial \alpha_{NwT}}{\partial \alpha_{j,t}}$$

ce qui est la même expression que la passe de "*backward*" de l'algorithme "*forward-backward*" de Baum-Welch :

$$\beta_{i,t-1} = \sum_j a_{i,j} b_j(y_t) \beta_{j,t} \quad (2)$$

Bridle a ainsi montré que malgré les différences de structures entre NN et HMM, ces méthodes utilisent le même algorithme. Les équations (1) et (2) sont les passes de propagation et de rétropropagation pour les deux modèles.

De la même façon, il a montré l'équivalence entre un alphanet utilisant le critère de "*relative entropy*" et les HMMs utilisant le critère de MMI et le critère de "*corrective training*".

4.4 Systèmes hybrides

Il existe de nombreux systèmes hybrides, nous nous limitons à l'étude des modèles hybrides HMM et NNs du type : MLP, TDNN et RNN. Le but de ces systèmes est d'obtenir de meilleurs performances que chacun des systèmes pris indépendamment. Différentes approches ont été développées : incorporer aux HMMs des connaissances discriminantes, intégrer un algorithme d'alignement temporel aux réseaux ou utiliser les connaissances extraites par deux critères différents en même temps.

On distingue quatre principales approches dont certaines peuvent être combinées :

- le modèle neuronal sert de "pré-traitement" aux modèles markoviens,
- les deux modèles sont optimisés ensemble à l'apprentissage,
- le modèle neuronal est un "post-traitement" des modèles markoviens,
- les deux modèles coopèrent lors de la reconnaissance.

Nous présentons quelques résultats afin d'avoir une vision de l'état de l'art dans le domaine. Cette énumération n'est pas exhaustive car le domaine est en rapide évolution. Nous avons aussi situé les approches développées dans cette étude dans cette catégorisation. Elles seront décrites en détail dans le chapitre 8.

4.4.1 NNs comme *pré-traitement* des HMMs

Les réseaux de neurones génèrent des approximations des probabilités a posteriori qui sont utilisés ensuite dans un formalisme markovien. le but poursuivi est d'intégrer un mécanisme d'alignement temporel dans les réseaux. L'apprentissage est séquentiel car les deux modèles sont optimisés séparément.

Deux façons de procéder sont possibles : utiliser directement l'algorithme de Viterbi sur les sorties des réseaux qui sont modélisées dans un formalisme HMM ou faire un apprentissage HMM avec un critère de MLE après l'utilisation d'un premier classifieur neuronal.

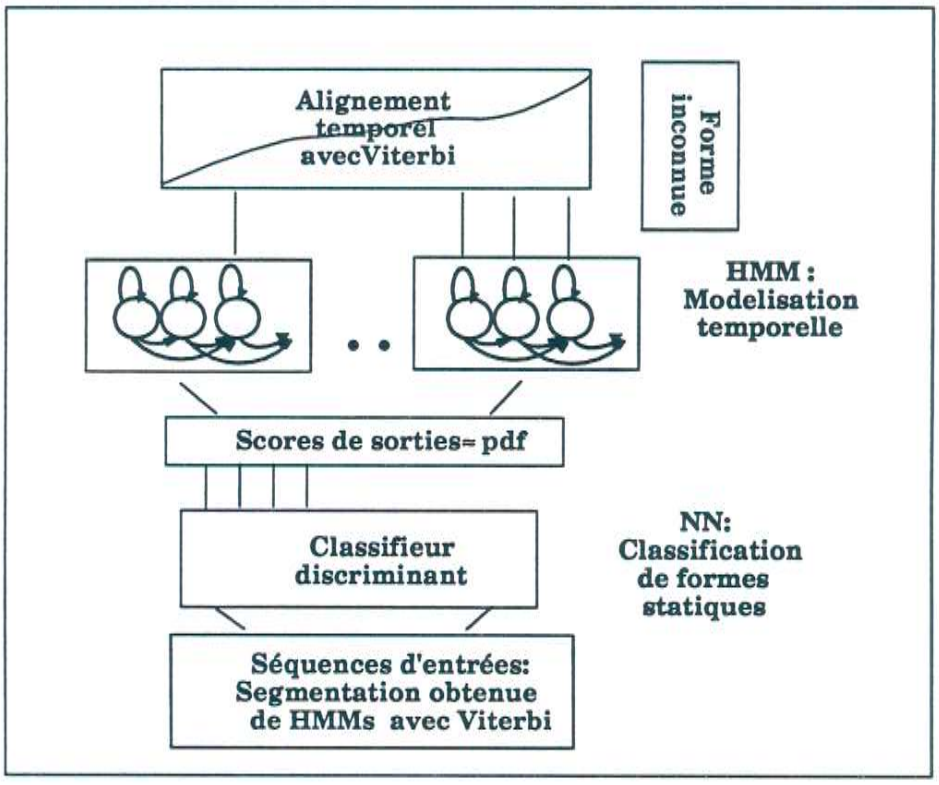

*Fig. 4.2 : **Exemple de pré-traitement neuronal d'un HMM.** L'abréviation "pdf" signifie densité de probabilité.*

Nous décrivons dans la suite de ce paragraphe quelques uns des travaux ayant permis d'obtenir des résultats significatifs.

Sorties du réseau considérées comme approximations de probabilités (pdf) d'un HMM

H. Bourlard *[Lernout et Hauspie-Belgique]*, **N.Morgan** *[SRI-USA]*

Les recherches menées par H. Boulard et Wellekens sur les réseaux de neurones ont débuté en 1987 chez Philips [Bourlard 87] et se poursuivent maintenant chez *"Lernout et Hauspie Speech Products"* qui commercialise entre autres leur système hybride "MLP/Viterbi".

Bourlard utilise un MLP de 9 trames de contexte puis un alignement temporel avec l'algorithme de Viterbi sans apprentissage HMM. Les sorties du réseau sont considérées comme des approximations des probabilités a posteriori P(C/x), x étant une observation, C une classe. La contrainte sur les scores de sorties est que la somme des scores soit égale à 1 et qu'ils se situent entre 0 et 1. Les probabilités a posteriori sont divisées par la probabilité a priori, la fréquence de chaque classe, pour être utilisées dans le formalisme HMM, cette information est en général codée dans le HMM par le modèle de langage et la structure du lexique. Notre approche est assez proche de celle de Bourlard (chapitre 8).

De nombreux tests ont été menés sur les bases de données de DARPA RM1, dépendante [Morgan 90] et indépendante du locuteur [Bourlard 91]. Nous citerons les derniers résultats obtenus à ICASSP 92 [Renals 92], [Bourlard 92]. Le réseau construit a une entrée de 26 * 9 trames et une couche cachée de 512 cellules. Le réseau a une structure globale et n'a pu être conçu ainsi que grâce à l'utilisation de machines spécialisées en traitement parallèle pour résoudre le problème du temps de calcul. L'apprentissage avec 5 RAP *"Ring Array Processors"* prend 24 heures pour 69 sorties, soit environ 150000 poids à estimer. Le réseau est intégré dans le système *"Decipher"* de SRI qui permet d'utiliser un formalisme markovien et de comparer de façon objective les méthodes hybrides aux HMMs.

Les résultats sur la base de données DARPA RM, indépendante du locuteur, sont donnés dans ce tableau. Cette base de données a un vocabulaire de 998 mots, en utilisant une grammaire bi-gram la perplexité devient 60. Les données d'apprentissage consistent en phrases prononcées par 109 locuteurs, 2830 phrases par des hommes et 1160 par des femmes. Les tests ont été menés sur 600 phrases correspondant aux tests Darpa de Février et d'Octobre 89.

Système	*Paramètres*	*% erreur Perplexité*	
		998	60
HMM-CI	125 762	44.7	14
MLP-69-CI	155 717	36.1	12.8
HMM-3428CD	5 541 844	21.9	4.9
MLP-69-3428CD	5 697 726	20.2	4.3

Tab 4.1 : ***Performances des HMMs et d'un modèle hybride MLP/Viterbi*** *pour des modèles indépendants du contexte (CI) et pour des modèles dépendants du contexte (CD).*

Un réseau de type MLP permettant d'obtenir directement 3428 sorties est impossible à construire même en utilisant des machines spécialisées comme les systèmes RAPs. Les HMMs avec autant de paramètres, par exemple des modèles dépendants du contexte, des mélanges de densités... sont plus aisés à construire. Une architecture modulaire de réseaux a été conçue. Les réseaux MLPs ont été utilisés avec un contexte, ils sont notés CDNN "*Context Dependent Neural Network*" [Bourlard 91]. Les réseaux ont été entraînés séparément pour des contextes différents puis intégrés dans une seule architecture qui génère les 3428 sorties correspondant aux phonèmes en contexte.

Deux résultats importants ont été obtenus dans cette étude; L'estimation des probabilités a posteriori par un réseau connexionniste permet d'améliorer les performances d'un HMM utilisant le critère de MLE et des modèles indépendants du contexte. Les performances des NNs sont proches de celles des HMMs dépendants du contexte.

F. Fallside, A. Robinson *(CUED-GB)*

Le système hybride, appelé REPN pour "*Recurrent Error Propagation Network*" utilise un réseau de neurones récurrent et un algorithme d'alignement temporel de type Viterbi, un algorithme à une passe [Ney 84], lors de la reconnaissance [Robinson, Fallside 91]. Cette approche est comparable au MLP/Viterbi de Bourlard. Une sortie du réseau donne la probabilité qu'une trame fasse partie d'un segment phonétique indépendamment du contexte. L'apprentissage du réseau récurrent dure environ 3 jours avec 10^{13} opérations sur des réels. Le système utilisé pour entraîner le réseau est une machine à architecture parallèle contenant 64 "transputers" T800

L'algorithme d'alignement temporel est utilisé sur les sorties du réseau récurrent global pour trouver la séquence de phonèmes la plus probable.

La base de données DARPA TIMIT [Garofolo 88] comprend 420 locuteurs ayant chacun prononcé 8 phrases. 317 locuteurs ont servi pour l'apprentissage et 103 pour le test. Le taux de reconnaissance phonétique pour les 61 symboles de TIMIT est de 70% (63,5% avec les insertions). Les résultats de K.F. Lee [Lee 89] et de Levinson [Levinson 89] obtenus dans les mêmes conditions sont comparables.

La base de données DARPA RM1 *"speaker dependent"* a aussi été testée avec ce système [Robinson 92]. Le réseau récurrent élaboré contient 38 456 poids. Comparé à un système de Markov utilisant des modèles de triphones ou à une architecture neuronale MLP utilisée par Bourlard, ce système RNN est beaucoup moins coûteux en occupation mémoire. Les taux de reconnaissance obtenus sont très encourageants pour ce type de méthode, 23% d'erreurs sans grammaire (perplexité 975) et 5.7% avec grammaire (perplexité 60). Les résultats ne sont cependant pas directement comparables avec d'autres systèmes, l'apprentissage ayant été mené sur 500 phrases pour les 12 locuteurs et les tests sur les 100 dernières phrases du corpus généralement utilisé pour l'apprentissage.

L. Devillers *(LIMSI-France)*, Ch. Dugast *(Philips-Allemangne)*

Notre expérience avec cette approche est décrite en détail dans le chapitre 8 de cette étude. Le système construit est formé d'un réseau modulaire TDNN et d'un algorithme de Viterbi. La base de données testée est aussi DARPA RM1 *"speaker dependent"*. Les résultats sont cependant difficilement comparables avec d'autres systèmes car les tests ont été menés jusqu'à présent sur trois locuteurs. Les meilleures performances obtenues avec cette intégration avec l'architecture la plus simple sont de 25% (perplexité 991) et de 4.7% (perplexité 63) sur les 100 phrases de test pour un réseau modulaire d'environ 40 000 poids. De meilleurs résultats ont été obtenus avec des architectures plus complexes, cf chapitre 8. Les expériences ont toutes été menées avec une machine monoprocesseur.

4.4.2 Optimisation des deux modèles ensemble lors de l'apprentissage

Les réseaux de neurones génèrent des approximations des probabilités a posteriori qui sont utilisés ensuite dans un formalisme markovien. le but poursuivi est d'intégrer un mécanisme d'alignement temporel dans les réseaux. L'apprentissage ici est coopératif. Après un apprentissage séquentiel, les réseaux sont optimisés avec un alignement temporel de type Viterbi. Ils participent à un réapprentissage ou "fine-tuning". On peut utiliser l'information issue de Viterbi de deux manières : l'erreur obtenue après alignement temporel est rétropropagée dans le réseau afin de modifier les poids des connexions, ou la segmentation issue de Viterbi sert à modifier les données d'entrée du réseau.

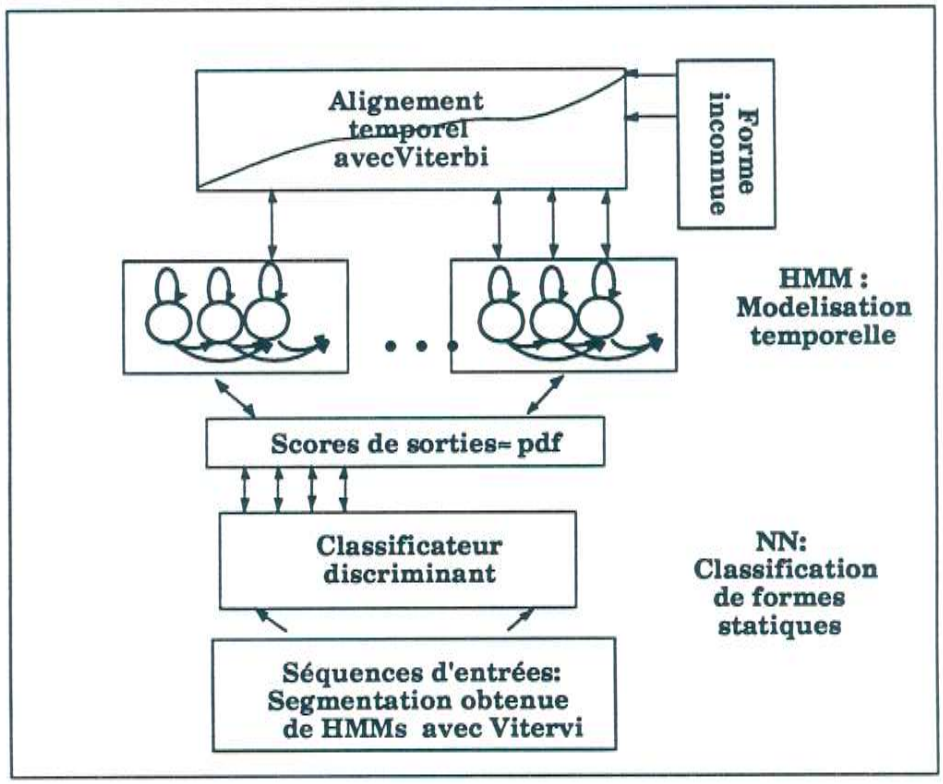

*Fig. 4.3 : **Exemple de coopération optimale entre HMMs et NN.***

Nous décrivons pour cette approche quelques uns des travaux ayant permis d'obtenir des résultats significatifs.

A. Waibel *(CMU-USA, ATR-Japon, Université de Karlsruhe-Allemagne)*,
M. Franzini *(TID-Espagne)*

Le CVT "*Connexionnist Viterbi Training*" [Franzini, Waibel 91] est constitué d'un réseau et d'un HMM. Les deux composants ne sont pas indépendants; L'apprentissage de l'un dépend de l'autre. Les deux réseaux sont entraînés séparément puis optimisés ensemble. Les sorties des réseaux sont les probabilités associées aux transitions entre états des modèles. Durant l'apprentissage, l'alignement temporel est forcé puisqu'on connaît la séquence correcte à reconnaître, les résultats d'alignement temporel sont réutilisés pour entraîner à nouveau le réseau avec de nouvelles données et servent aussi à modifier les poids des connexions.

Un réseau à 2 couches cachées est utilisé, le nombre de cellules dans les couches cachées est de 34, le contexte est de 7 trames en entrée pour des vecteurs de 13 composantes. La taille du réseau n'est donc pas très grande Le choix d'un réseau récurrent (à t-10) n'a pas permis d'améliorer suffisamment les performances pour jusitifier du temps de calcul nécessaire. Le système est entrainé sur une machine Convex-C (30 MIPS).

La base des TI-Digits "*Texas Instruments Connected Digits Recognition Task*" correspond à des chiffres enchaînés, le nombre de mots différents est de 12. Cette base contient 10 000 exemples de phrases de longueur variant entre 1 et 7 mots. Le tableau suivant donne les meilleurs scores obtenus avec le CVT avec en supplément les stratégies suivantes : des modèles de mots plutôt que de phonèmes, plusieurs modèles par mots et un module de "*corrective training*".

% erreur TI DIGITS	*Système*	*Test %erreur/mot*	*Test %erreur/phr*
Gauvain (Bell labs)	CHMM (MAP)	0.4	1.3
Waibel-Franzini (CMU)	CVT	0.9	2.0

Tab 4.2 : ***Comparaison entre les meilleures performances des CHMMs utilisant le critère de MAP et du modèle hybride CVT.***

J. Bridle *(RSRE-GB)*

Les travaux de J. Bridle [Bridle 90] ont permis de faire progresser les théories mathématiques sur les réseaux de neurones et les modèles markoviens, cf §4.3. Les "*alphanets*" ont cependant plus de ressemblance avec des modèles markoviens qu'avec des réseaux. "*Alpha*" fait référence à l'algorithme d'apprentissage "*forward*", "*net*" à un réseau. C'est en fait une architecture de réseau récurrent avec une interprétation HMM. La dernière couche du réseau récurrent est transformée pour donner de réelles probabilités en sortie et le critère usuel de MSE est remplacé par le critère de MMI utilisé dans les HMMs. L'algorithme d'apprentissage est celui de Baum-Welch modifié pour utiliser le critère de MMI. Ce système hybride n'a été testé que sur des mots isolés.

L. Bottou, X. Driancourt, P. Gallinari *(LRI-France)*

Les travaux de L. Bottou [Bottou 91] ont permis de formaliser les deux approches, réseaux de neurones et modèles markoviens de façon similaire. Notons aussi tout l'intérêt porté aux travaux de X. Driancourt et P. Gallinari sur des réseaux hybrides TDNN-LVQ-DTW [Driancourt 92]. Nous ne détaillerons pas ces systèmes dans cette partie consacrée plus particulièrement aux méthodes hybrides NN/HMM mais nous ferons référence aux idées développées dans ces travaux dans le chapitre 8.

P. Haffner *(CNET-France)*

Le système MS-TDNN pour "*Multi State Time Delay Neural Network*" proposé par P. Haffner permet d'intégrer directement le mécanisme d'alignement temporel dans un réseau dont les sorties sont des mots. Les mots sont modélisés temporellement dans la dernière couche cachée des TDNNs, si on utilise 5 états pour représenter un mot, la dimension de cette couche est de 5 (nombre de cellules) multiplié par le nombre de trames obtenues après propagation de la séquence d'entrée. Chaque sortie intègre la séquence des vecteurs de la dernière couche cachée avec une procédure d'alignement temporel. La somme d'activations maximum obtenue pour le meilleur chemin de chaque mot détermine le mot reconnu. Les frontières des phonèmes dans les mots étaient nécessaires pour initialiser les modèles temporels dans les premières expériences menées (MS-TDNN avec bornes). Sans cette connaissance, le MS-TDNN donne des résultats un peu moins bons (MS-TDNN sans bornes).

La base de données est une base téléphonique de 10 chiffres prononcés par 750 locuteurs. L'apprentissage de 3500 chiffres (375 locuteurs) dure 1 jour sur une station IBM RS/6000.

% erreur Chiffres isolés	*Système*	*Test %erreur/mot*
CNET	MS-TDNN *sans bornes*	1.6
CNET	MS-TDNN *avec bornes*	1.0
CNET	CHMM	0.7

Tab 4.3 : ***Comparaison entre les meilleures performances des CHMMs et le MS-TDNN avec ou sans connaissance des bornes des phonèmes dans les mots pour initialiser le système.***

Y. Bengio *(Mac Gill-Canada)*

Y. Bengio propse des comparaisons entre plusieurs approches hybrides utilisant un réseau comme prétraitement des HMMs [Bengio 91] notamment entre un système hybride NN-DTW et NN-HMM et entre un réseau NN-HMM avec une intégration optimale et sans intégration optimale. Les tests ont été menés sur une tâche réduite de classification de 8 phonèmes (les plosives) issues de la base TIMIT. Pour cette expérience, de meilleurs résultats sont obtenus avec un système hybride NN-HMM que NN-DTW et l'intégration optimale est plus performante. La différence entre une intégration optimale et non optimale n'est pas très grande mais est tout de même significative, de 87% (pour la version non optimale) à 90% de reconnaissance.

J. Tebelskis *(CMU-USA)*, Waibel *(CMU-USA)*

Le réseau utilisé dans le modèle hybride est un modèle neuronal prédictif, appelé LPNN pour "*Linked Predictive neural networks*". Une description des réseaux prédictifs est donnée au paragraphe 3.5.3. Chaque classe phonétique est modélisée par un réseau, celui-ci apprend la correlation temporelle entre les trames des segments d'une classe. Des algorithmes de programmation dynamique DTW ou Viterbi sont utilisés pour organiser ces réseaux, les liants en séquences correspondant aux phonèmes des mots. L'apprentissage des mots de la base de

donnée comprend trois phases. La première correspond à une propagation de l'activité à travers les réseaux et au calcul d'une matrice d'erreur entre les trames prédites en sortie et les trames d'entrée. La seconde est l'estimation du meilleur chemin dans cette matrice entre le signal (trames d'entrée) et les modèles (trames de sortie). Enfin la dernière phase consiste à rétropropager l'erreur sur les poids des réseaux. De bonnes performances ont été obtenues sur des mots isolés [Iso 90], [Levin 90]. Tebelskis et Waibel ont appliqué cette structure à la parole continue [Tebelskis, Waibel 90][Tebelskis, Waibel 91]. Chaque réseau prédictif code un état d'un modèle HMM. Pour être plus précis, un modèle HMM peut être décrit par 3 états représentés par 3 réseaux codant le début, le milieu et la fin d'un phonème. 120 LPNNs sont alors nécessaires pour modéliser les 40 phonèmes de la base de donnée. L'apprentissage est le même que pour les mots isolés. Mais pour la parole continue, le "*One Stage algorithm*" [Ney 84] est utilisé pour optimiser en même temps la segmentation et la reconnaissance des mots d'une phrase. Les résultats obtenus sur une base de donnée de 204 phrases en anglais sont meilleurs pour ce système hybride LPNN-HMM que pour un système HMM utilisant une gaussienne par état. Les réseaux prédictifs sont très interessants pour modéliser la parole qui est par essence dynamique. Cependant, il faut souligner leur manque de pouvoir discriminant. Les réseaux sont entraînés pour prédire séparemment une classe phonétique, ils n'ont jamais appris à rejeter les autres classes.

4.4.3 NNs comme *post-traitement* des HMMs

Les réseaux sont utilisés comme un second classifieur après un apprentissage avec un critère non discriminant comme le MLE. Cette approche vise à améliorer les performances des HMMs avec une connaissance discriminante.

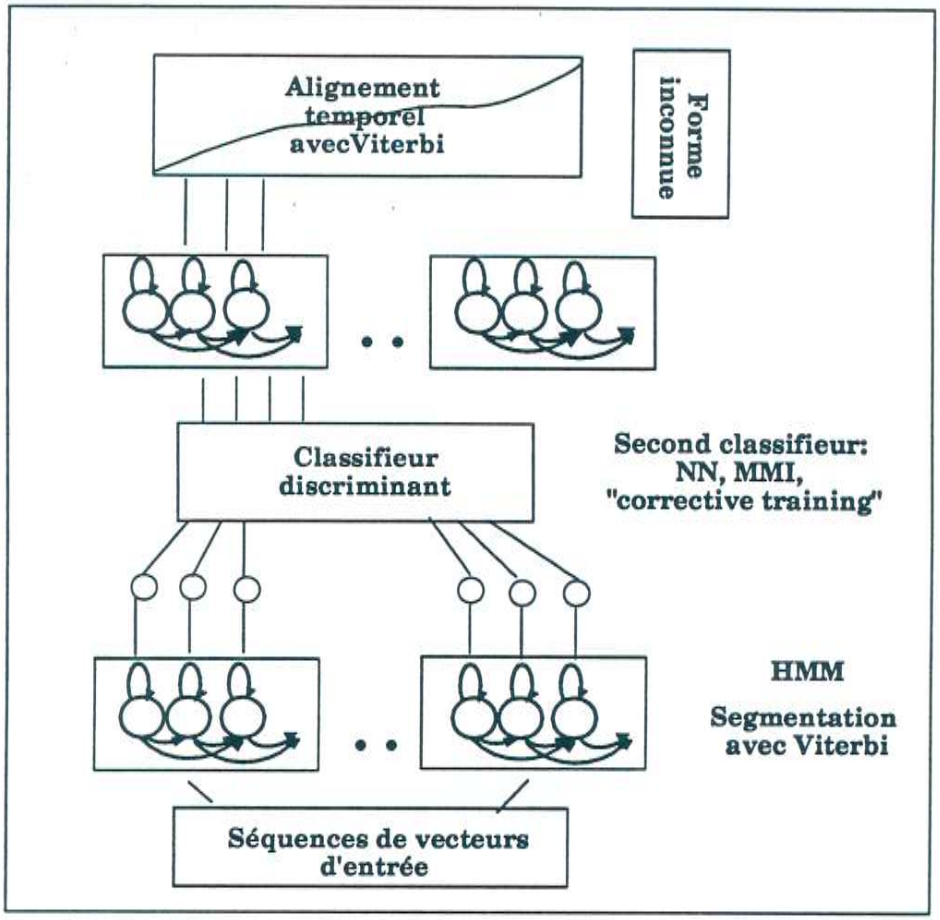

Fig. 4.4 : NN comme post-traitement des HMMs.

Nous décrivons pour cette approche quelques uns des travaux ayant permis d'obtenir des résultats significatifs.

W. Huang, R. Lippmann *(MIT-USA)*

L'intérêt de l'étude [Huang, Lippmann 89] est de comparer plusieurs techniques. Le classifieur de référence, c'est-à-dire le HMM, utilisé comme premier classifieur dans le système hybride, est modélisé avec 1 gaussienne par état et est entraîné avec un algorithme EM. Le second classifieur peut être un MLP, un HMM modélisant des mélanges de densités, des RBFs ou un classifieur de type KNN avec k=1. La base de données utilisées contient des paires-minimales, il s'agit du E-set qui comporte les 9 lettres, BCDEGPTVZ. Le test contient 16 locuteurs. On espère sur une base comme celle là que les méthodes discriminantes donnent les meilleurs résultats.

HMM	+MLP	+3 Gaussiennes par état	+RBF 70 centres	+KNN k=1
11.3	13.4	20.6	12.8	36

Tab 4.4 : ***Comparaison entre des systèmes hybrides et non hybrides sur le E-set.***

Les meilleurs résultats sont obtenus avec le HMM standard sans second classifieur, les RBFs et le MLP. Les autres méthodes n'ont pas suffisamment de données pour converger. Dans ce cas, l'ajout d'un second classifieur n'apporte pas de meilleurs résultats qu'avec un HMM utilisant une densité par état.

Malgré l'utilisation de la même base de données pour évaluer les performances des HMMs et NNs dans le §4.3.1, les résultats ne sont pas complètement comparables avec cette expérience car le corpus de test est différent. La seule conclusion commune est de dire que les modèles markoviens continus sont meilleurs dans ces deux expériences que les NNs et que les HMM-NNs.

Avec une base de données de petite taille, les résultats sont difficiles à exploiter. C'est souvent un défaut des réseaux, d'être testé sur des bases de données insuffisantes. Une des raisons en est le temps d'apprentissage nécessaire pour entraîner les réseaux. Non seulement un apprentissage peut être très long pour une grande base de données mais souvent plusieurs essais sont nécessaires pour obtenir une structure appropriée à la tâche de classification.

S. Austin, R. Swartz *(BBN-USA)*

S. Austin [Austin 92] présente une méthode de "*rescoring*" des probabilités HMMs par un réseau de neurones. Afin d'éviter des apprentissages trop longs, cette nouvelle évaluation est faite sur les N meilleures phrases sélectionnées après un apprentissage HMMs déjà très performant. Le but est d'utiliser une connaissance discriminante simplement sur les mots ambigus sélectionnés par la méthode des "*N best*".

Les sorties des réseaux représentent des phonèmes. Le réseau est dit "segmental", quelle que soit la taille des segments, ils sont ramenés à une taille fixe permettant d'obtenir un score. La durée des segments est modélisée artificiellement et rajoutée pour pondérer les sortie des réseaux.

La base de données testées est celle de DARPA RM, indépendante du locuteur. L'apprentissage est fait sur les 109 locuteurs + les tests de février 89. La reconnaissance est menée sur les tests d'Octobre 89. Le tableau regroupe les scores obtenus sur les 20 phrases avec les CHMMs seuls, avec les réseaux de neurones et avec l'approche mixte.

DARPA RM S.I	*système*	*%erreur/mot*
BBN	CHMM	3.5
BBN	NN segmental	9.0
BBN	CHMM-NN	2.8

*Tab 4.5 : **Comparaison entre le "rescoring" avec le NN seul et une approche mixte CHMM-NN par rapport aux scores obtenus avec les CHMMs.***

Cette méthode de "*rescoring*" apporte de nettes améliorations des scores.

Chow [Chow 90] présente des résultats avec un système équivalent (HMM + N Best) mais avec un "*rescoring*" effectué par une modification des paramètres avec un apprentissage utilisant un critère de MMI. Les performances obtenues ne sont pas convaincantes. Les taux de reconnaissance sont pratiquement les mêmes avec et sans le critère MMI, avec un avantage non significatif pour la méthode de "*rescoring*".

T. Niles *(Xerox-USA)*

L'approche de Niles [Niles 92] utilise les qualités de discrimination des NNs pour les phonèmes particulièrement difficiles à reconnaître comme /r/, /t/. C'est une méthode de "*phoneme spotting*" pour faire du "*corrective training*". Les tests ont été menés sur la base Timit pour reconnaître ces trois phonèmes avec une amélioration de 30% par rapport aux HMMs utilisés.

4.4.4 Coopération des deux modèles lors de la reconnaissance

La coopération des probabilités se fait au niveau de la reconnaissance. L'idée était de mesurer l'apport des connaissance extraites par les réseaux.

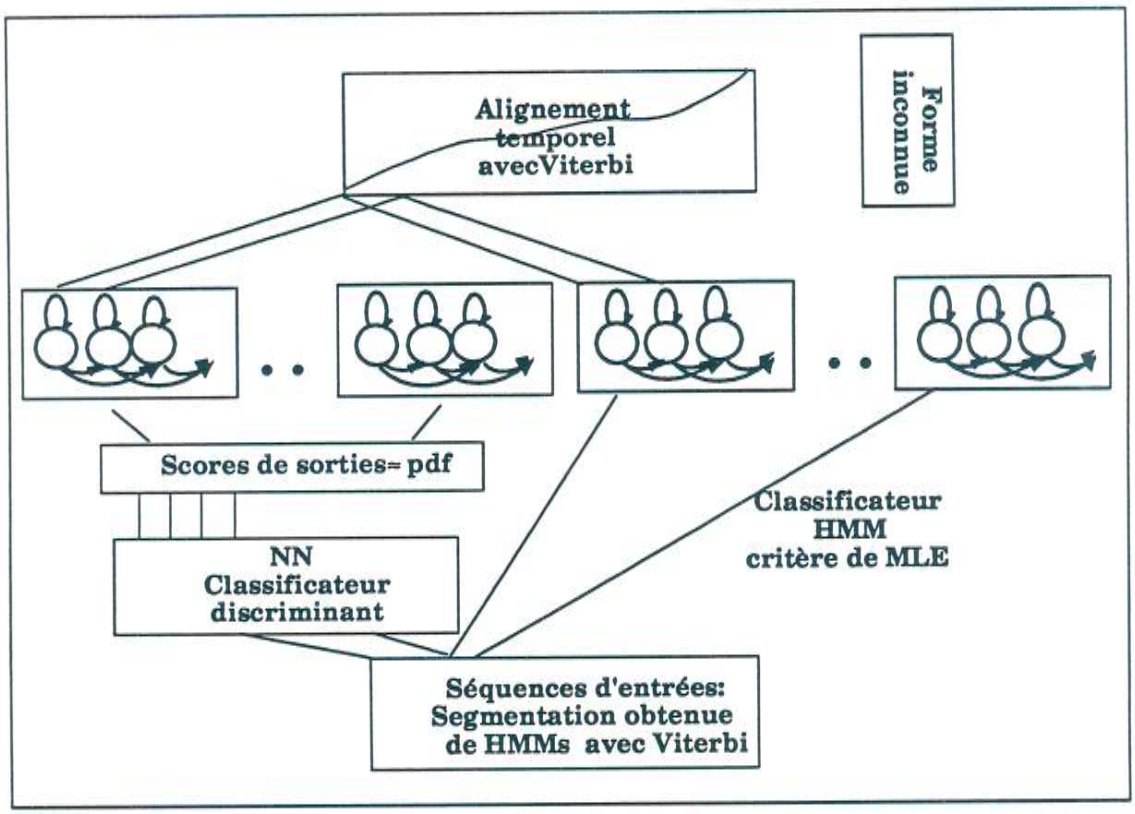

Fig. 4.5 : ***Coopération des probabilités des HMMs et du NN lors de la reconnaissance.***

L. Devillers *(LIMSI-France)*, **Ch. Dugast** *(Philips-Allemangne)*

Nous avons présenté cette voie de coopération entre les deux systèmes au congrès ICASSP 92 [Dugast, Devillers 92]. Les systèmes mis en parallèle sont un système hybride TDNN-HMM décrit dans le paragraphe 8.4 et un CHMM. Nous détaillerons les résultats que nous avons obtenus sur la base de 1000 mots DARPA RM1 dépendante du locuteur (pour le locuteur JWS0) dans le chapitre 8.

S. Renals *(ICSI-USA)*, **N. Morgan** *(SRI-USA)*

Les probabilités sont issues des systèmes déjà décrits dans le §4.4.1. Les performances obtenues portent sur la même base de données DARPA RM, indépendante du locuteur.

La règle d'intégration des scores est utilisée au moment de la reconnaissance avec l'algorithme de Viterbi. C'est tout simplement l'addition des logarithmes des probabilités pondérés par des coefficients λ_i. Les scores issus du MLP sont approximativement des probabilités a posteriori. Lorsqu'on divise ces probabilités par les probabilités a priori, on retrouve les densités conditionelles habituellement modélisées par les HMMs.

$$\log(P(x/q_j)) = \lambda_1 \log(\frac{P_{mlp}(q_j/x)}{P(q_j)}) + \lambda_2 \log(P_{hmm}(x/q_j))$$

Les λ_i sont empiriquement optimisés avec comme seule contrainte que leur somme soit égale à 1.

Système	*Paramètres*	*% erreur Perplexité*	
		998	60
HMM-CI	125 762	44.7	14
MLP-69-CI	155 717	36.1	12.8
HMM + MLP	281 479	29.5	7.9

Tab 4.6 : ***Comparaison entre les performances des HMMs, des MLPs et de la structure mixte.***

Le codage discriminant des réseaux apporte des connaissances complémentaires de celles codées par les HMMs. Nous expliquerons en détail sur nos résultats l'apport des réseaux.

4.5 Conclusion

Un grand nombre de travaux sont menés à l'heure actuelle sur les systèmes hybrides. Les laboratoires cités préalablement pour avoir des systèmes markoviens très performants se sont en général intéressés aux modèles neuronaux et explorent maintenant des coopérations entre les deux techniques. Nous avons cité une partie des travaux les plus connus à l'heure actuelle parmi eux, ceux de BBN, de SRI, du MIT... pour les américains, ceux de Lernout et Hauspie, de Cambridge, de RSRE, du CNET, du LRI pour les européens et ceux de ATR pour les japonais. Cette liste n'est évidemment pas exhaustive et de nombreuses approches n'ont pas été décrites dans cette étude. Les résultats obtenus jusqu'à présent sont prometteurs. Il est à remarquer aussi que les systèmes hybrides commencent à être concurrents

maintenant des meilleurs HMMs dans la course aux performances sur les bases DARPA. La voie la plus prometteuse et la moins couteuse semble être la coopération des probabilités entre les deux techniques lors de la reconnaissance. Le système de Morgan et Bourlard utilisant seulement les probabilités issues du réseau pour modéliser les densités de probabilité et l'algorithme de Viterbi pour faire un alignement temporel montre que les réseaux de neurones sont capables d'obtenir de meilleurs performances que les HMMs. Cependant le coût d'apprentissage de tels systèmes est très onéreux et nécessite des machines multiprocesseurs spécialisées. L'avenir des réseaux dépend surement beaucoup de la puissance des machines utilisées et du traitement parallèle. Ces méthodes sont malgré tout récentes et nécessitent encore de nombreux travaux.

Cette conclusion termine la partie présentation des *"Méthodes de classification en reconnaissance de la parole"*. Les méthodes classiques ainsi que les modèles connexionnistes développés dans ce travail ont été décrits dans cette partie. Nous avons essayé de présenter les différentes techniques et de montrer les similitudes entre les méthodes. Les défauts et qualités des modèles markoviens et des réseaux de neurones de type MLP sont plus particulièrement détaillés dans ce chapitre ainsi qu'un état de l'art des coopérations entre ces deux méthodes.

Les parties suivantes ont pour objet de décrire nos travaux. La première partie regroupe des expériences de classification phonétique avec différents réseaux (MLP, LVQ, TDNN) ainsi qu'avec des architectures modulaires de réseaux TDNN. La deuxième partie décrit nos expériences avec des systèmes hybrides TDNN-HMM.

Partie 2

Classification phonétique et Conception d'architectures modulaires

Cette partie porte sur des expériences de classification phonétique avec différentes architectures de réseaux et algorithmes d'apprentissage. ***Les chapitres 5 et 6*** *regroupent une série d'expériences visant à tester les capacités des réseaux, le problème de l'adéquation de la tâche à l'architecture, les relations entre prétraitements et performance en classification, l'optimisation des algorithmes d'apprentissage. Ces travaux ont été très utiles au déroulement de cette thèse et ont permis de développer des architectures de réseaux et des algorithmes d'apprentissage performants. Nous relatons les expériences et les conclusions sur les stratégies à adopter pour faire de la classification en parole.* ***Le chapitre 7*** *porte sur la notion de modularité dans les réseaux. La décomposition de la tâche en sous-tâches apporte une réponse au problème d'apprentissage de grandes bases de données.*