ORSAY

n° d'ordre

UNIVERSITE DE PARIS SUD

CENTRE D'ORSAY

THESE

présentée

pour obtenir

LE TITRE DE DOCTEUR EN SCIENCES

par

Léon BOTTOU

Sujet

Une Approche théorique de l'Apprentissage Connexioniste; Applications à la reconnaissance de la Parole.

Soutenue le 6 fevrier 1991 devant la Commission d'examen composée de

M. Dominique GOUYOU-BEAUCHAMPS, Président

M. Robert AZENCOTT

M. Hervé BOURLARD

M. Michel COSNARD

Mme Françoise FOGELMAN-SOULIE

M. Jean-Sylvain LIENARD

M. Yann LE CUN

Je remercie Monsieur le Professeur D. GOUYOU-BEAUCHAMPS, qui m'a fait l'honneur de présider ce jury.

Monsieur le Professeur M. COSNARD et Monsieur H. BOURLARD ont accepté d'évaluer mon travail de thèse, et m'ont adressé des critiques instructives. Je tiens à les en remercier vivement.

Je remercie également Monsieur le Professeur R. AZENCOTT, qui a immédiatement accepté de participer à ce jury.

Ce travail a été dirigé par Madame le Professeur F. FOGELMAN, qui a animé avec dynamisme notre équipe. Qu'elle trouve ici l'expression de mes remerciements.

C'est à l'enseignement et aux conseils avisés de Monsieur J.S. LIENARD que je dois mon interêt pour les problèmes de traitement de la parole. Je tiens à le remercier, ainsi que les membres du LIMSI avec lesquels j'ai pu collaborer, P. BLANCHET et L. DEVILLERS.

C'est à la passion et à la persévérance de Y. LE CUN et de J. BOURRELY que je dois d'avoir étudié les algorithmes d'apprentissage connexionnistes. Je ne saurais assez les en remercier.

Enfin, je remercie vivement tous les membres de notre équipe. Les conseils pertinents de P. GALLINARI et S. THIRIA m'ont été précieux. Les commentaires constructifs de C. & J. BOURRELY, M. de BOLLIVIER, X. DRIANCOURT, C. MEJIA, Y. BENNANI et de tous les thésards, que je ne peux citer tous ici, m'ont été infiniment utiles.

• Sommaire

• **Sommaire** 3

• **Résumé** 9

Introduction 13

1.1 Contexte. 13

1.2 Thème. 16

1.2.1 Reconnaissance de la parole. 16

1.2.2 Révisions. 17

1.3 Plan de la thèse. 18

1.3.1 Partie 1: Connexionnisme et apprentissage. 19

1.3.2 Partie 2: Application à quelques problèmes de reconnaissance de la parole, et critique. 19

1.3.3 Partie 3: Algorithmes connexionnistes et modèles de Markov. 20

1.4 Chronologie. 20

Techniques connexionnistes. 21

2.1 Qu'est ce que le connexionnisme ? 21

2.1.1 Un courant des sciences humaines. 21

2.1.2 Un ensemble de techniques partageant quelques points communs. 22

2.2 Petit lexique du connexionnisme. 23

2.2.1 Unités linéaires à seuil. 23

2.2.2 Autres types d'unités élémentaires. 28

2.2.3 Réseaux et connexions. 28

2.2.4 Synthèse. 31

2.3 Quelques modèles classiques 31
2.3.1 Perceptrons et Adalines. 32
2.3.2 Mémoires associatives linéaires. 34
2.3.3 Réseaux de Hopfield. 36
2.3.4 Perceptrons multi-couches. 37
2.3.5 Learning Vector Quantization (LVQ). 42
2.4 La problématique de la généralisation 43
2.5 Une problématique unifiée 45

Apprentissage stochastique. 47

3.1 Apprentissage et optimisation stochastique. 47
3.1.1 Optimisation stochastique. 48
3.1.2 Forme générale de l'optimisation stochastique. 49
3.1.3 Hypothèses asymptotiques et généralisation. 52
3.2 Exemples. 53
3.2.1 Régressions et algorithmes connexionnistes. 53
3.2.2 Minimisation de l'erreur de quantification. 55
3.3 Etude de la convergence. 56
3.3.1 Un cas simple, incluant le cas convexe. 57
3.3.2 Cas général. 65
3.3.3 Conclusion 71
3.4 Application au problème de reconnaissance des formes. 72
3.4.1 Le risque moyen. 73
3.4.2 Minimisation du risque moyen. 74
3.4.3 Estimation des vraisemblances conditionnelles. 78
3.4.4 Estimation des probabilités a posteriori. 80
3.4.5 Conclusion. 85

Apprendre avec un nombre limité d'exemples. 87

4.1 Ressources limitées. 87
4.1.1 Apprentissage avec répétitions. 88
4.2 Difficultés du problème. 89
4.2.1 Restauration d'une densité inconnue. 89
4.2.2 Théorème de Glivenko-Cantelli. 89
4.2.3 Interprétation intuitive. 91
4.3 Un problème de convergence uniforme. 92
4.3.1 Limite en probabilité de la fonctionnelle empirique. 92
4.3.2 Convergence uniforme en probabilité. 92

4.4 Théorèmes de Vapnik et Chervonenkis. 93
4.4.1 Conditions suffisantes de convergence uniforme des fréquences des événements vers leur probabilité. 93
4.4.2 Convergence uniforme des moyennes empiriques vers les espérances mathématiques. 96
4.4.3 Optimisation du coût empirique. 98
4.5 Cas de la reconnaissance des formes. 99
4.5.1 Minimisation d'une approximation du risque moyen. 99
4.5.2 Evaluation des probabilités a posteriori. 100
4.5.3 Evaluation des vraisemblances. 102
4.5.4 Ordres de grandeur 103
4.6 Procédures d'apprentissage. 104
4.6.1 Mesure de la qualité d'un système. 105
4.6.2 Minimisation structurelle. 106

Aspects techniques. 111
5.1 Algorithmes stochastiques. 112
5.2 Aspects numériques de l'optimisation. 113
5.2.1 Bon et mauvais conditionnement. 114
5.2.2 Heuristiques pour le perceptron multi-couches. 117
5.2.3 Utilisations du hessien pour contrôler les gains. 119
5.3 Outils de simulation. 124
5.3.1 Intérêt. 124
5.3.2 Exemples. 126
5.4 Conclusion. 128

Reconnaissance de la parole. 131
6.1 Le signal de parole. 131
6.1.1 Physique du signal de parole. 131
6.1.2 Caractéristiques spectrales. 132
6.1.3 Du signal au message. 133
6.1.4 Contraintes. 134
6.2 Chaînes de traitement. 134
6.3 Acquisition et prétraitements. 135
6.3.1 Codage spectral, banc de filtres. 136
6.3.2 Codage prédictif linéaire. 136
6.3.3 Codage homomorphique (cepstres). 137
6.3.4 Quantification vectorielle. 138

6.4 La reconnaissance par alignement temporel. 138
6.4.1 Métriques. 139
6.4.2 Programmation dynamique. 140
6.4.3 Parole continue. 141
6.4.4 Indépendance par rapport au locuteur. 141
6.5 Modèles statistiques: HMM. 142
6.5.1 Une approche paramétrique. 142
6.5.2 Source de Markov. 143
6.5.3 Détermination des paramètres. 145
6.5.4 Convergence. 149
6.6 HMMs pour la reconnaissance de la parole. 150
6.6.1 Règle Bayesienne. 151
6.6.2 Cas de la parole continue. 152
6.6.3 Remarques. 156

Méthodes connexionnistes pour la reconnaissance de la parole. 161
7.1 Architectures pour la reconnaissance de la parole. 161
7.1.1 Une classification atypique. 162
7.1.2 Reconnaissance statique. 162
7.1.2 Reconnaissance dynamique. 163
7.2 Réseaux à délais. 166
7.2.1 Structure et apprentissage des réseaux à délais. 166
7.2.2 Réseaux récurrents et réseaux à délais. 167
7.2.3 Exemples de réseaux à délais. 169
7.3 Conclusion. 174

Algorithmes modulaires d'apprentissage. 175
8.1 Systèmes modulaires. 175
8.1.1 Motivation. 175
8.1.2 Ecriture modulaire du coût local J(x,w). 177
8.2 Apprentissage de systèmes modulaires. 179
8.2.2 Calcul des dérivées du coût local. 179
8.2.3 Liens avec les approches usuelles. 182
8.3 Illustration. 185
8.3.1 Exemples d'algorithmes modulaires. 186
8.3.2 Implémentation d'algorithmes modulaires. 189

8.4 Conclusion. 190

Systèmes connexionnistes et modèles de Markov. 193

9.1 Descente de gradient dans les modèles de Markov. 194

9.1.1 Gradient de la vraisemblance d'une observation. 195

9.1.2 Résolution des contraintes. 197

9.2 Modèles de Markov discriminants. 201

9.2.1 Capacité discriminante. 202

9.2.2 Evaluation des probabilités a posteriori dans un modèle de Markov caché. 202

9.2.3 Mise en œuvre. 208

Reconnaissance de séquences et modèles de Markov discriminants. 213

10.1 Segmenter, reconnaître. 213

10.1.1 Extraire des séquences de symboles. 213

10.1.2 Programmation dynamique. 214

10.2 Modèles de Markov discriminants pour reconnaître des séquences. 214

10.2.1 Motivation et contraintes. 214

10.2.2 Un modèle Markovien discriminant. 215

10.2.3 Topologies des modèles discriminants. 218

10.2.4 Algorithme de Viterbi, et dérivés. 221

- **Epilogue 225**

- **Références bibliographiques 227**

• Résumé

Ce mémoire de thèse est consacré à l'étude théorique des modèles d'apprentissage connexionniste, et à leur application pour la reconnaissance automatique de la parole. Il est divisé en trois parties.

* * *

La première rappelle les principaux algorithmes connexionnistes, et établit leurs liens profonds avec les méthodes statistiques d'une part, avec les algorithmes stochastiques utilisés en traitement du signal d'autre part.

Le connexionnisme regroupe plusieurs algorithmes d'apprentissage en apparence très différents. La plupart cependant consistent à minimiser un coût de la forme

$$C(\mathbf{w}) = E(J(\mathbf{x},\mathbf{w})) = \int J(\mathbf{x},\mathbf{w})\ p(\mathbf{x}) d\mathbf{x} \qquad (1)$$

au moyen d'une approximation stochastique de la descente de gradient. Dans la formule (1), $J(\mathbf{x},\mathbf{w})$ désigne le coût associé au traitement de l'exemple $\mathbf{x}$ par le système de paramètre $\mathbf{w}$. La densité de probabilité $p(\mathbf{x})$ des exemples $\mathbf{x}$ représente les caractéristiques inconnues que l'on souhaite apprendre par l'exemple; elle est donc inconnue. On peut cependant minimiser $C(\mathbf{w})$ au moyen d'un algorithme itératif de gradient stochastique, consistant à tirer un exemple $\mathbf{x}_t$ au hasard, et à recalculer le paramètre $\mathbf{w}$ de la façon suivante:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \varepsilon_t \frac{\partial}{\partial \mathbf{w}} J(\mathbf{x}_t,\mathbf{w}_t) \quad \text{avec } \varepsilon_t>0, \quad \sum_t \varepsilon_t = +\infty, \ \sum_t \varepsilon_t^2 < +\infty \qquad (2)$$

Les propriétés théoriques des coûts de la forme (1) sont bien connues en statistiques; celles des algorithmes de la forme (2) ont été explorées en traitement du signal. Elles permettent:

- De démontrer de façon générale la convergence de l'algorithme de descente stochastique de gradient au prix de quelques hypothèses sur J. Cela prouve simultanément la convergence d'un grand nombre d'algorithmes connexionnistes, dont le perceptron, l'adaline, les perceptrons multi-couches, LVQ2…

- D'identifier les équivalences entre certaines fonctions de coût et l'estimation statistique de probabilités definies sur les exemples $\mathbf{x}$, au moyen des principes des moindres carrés et du maximum de vraisemblance.

- D'appliquer les théorèmes de convergence uniforme de Vapnik & Chervonenkis, qui permettent d'étudier les propriétés non asymptotiques de ces algorithmes. Cela permet d'étudier la généralisation, c'est à dire la performance sur des exemples autres que ceux utilisés pour l'apprentissage. On peut alors établir des procédures de validation et de selection, afin d'obtenir une meilleure généralisation.

* * *

La seconde partie traite des problèmes pratiques de mise en œuvre, d'une façon générale, et plus particulièrement dans le cas de la reconnaissance de la parole.

Les algorithmes d'optimisation décrits dans la partie 1 sont fréquemment non linéaires. Ils posent donc des problème numériques ardus, qu'il convient de traiter efficacement. Quelques méthodes rigoureuses et empiriques permettent en effet d'améliorer de quelques ordres de grandeur la vitesse de convergence. Plusieurs implémentations ont été réalisées durant la thèse, et constituent de volumineux logiciels.

Après avoir présenté les techniques actuelles de reconnaissance de la parole, l'expérimentation d'une technique connexionniste, les réseaux à délais, est relatée. Un système combinant un réseau à délai et une programmation dynamique a également été utilisé.

Ce type de systèmes hybrides posent un problème que l'on rencontre dans un grand nombre d'applications: la coopération de plusieurs algorithmes d'apprentissage. Il est possible d'étudier ce problème dans le cadre général de l'analyse de la première partie. On obtient alors un formalisme permettant de concevoir des réseaux modulaires.

Plus encore que la coopération d'algorithmes, c'est l'aspect structurel des systèmes d'apprentissage qui est abordé ici: les algorithmes usuels apparaîssent également comme l'enchaînement de quelques modules élémentaires. Ce jeu de construction facilite la conception et l'implémentation d'algorithmes complexes.

* * *

Les algorithmes connexionnistes standards intègrent difficilement les aspects temporels de la parole. En particulier, il n'est pas possible de traiter élégamment des signaux de longueurs différentes. L'approche récente de Bourlard & Wellekens consiste à combiner une technique classique, les modèles de Markov cachés, et un algorithme connexionniste, le perceptron multi-couches.

On retrouve là encore une problème de coopération d'algorithmes d'apprentissage. Un système de reconnaissance de séquences fondé sur cette approche est présenté. Son étude éclaire certains problèmes spécifiques posés par ce type d'algorithme.

* * *

En conclusion, une théorie unifiée des méthodes d'apprentissage connexionnistes est présentée dans ce mémoire de thèse, illustrée par le problème riche de la reconnaissance automatique de la parole. Cette théorie est fondée sur les liens étroits qui unissent algorithmes connexionnistes, statistiques, et approximations stochastiques.

1 Introduction

1.1 Contexte.

Depuis le milieu des années 80, les réseaux connexionnistes ont connu un fantastique regain, soutenu il est vrai par l'augmentation régulière des financements européens et nationaux apportés à ces recherches.

i Aperçu historique.

On s'accorde parfois à trouver l'origine de ce courant les travaux de McCulloch et Pitts [1] qui montraient qu'assembler des modèles très simples de neurones permet de réaliser des opérations plus complexes. On recherchait alors des architectures possibles pour les calculateurs numériques; les cybernéticiens constatèrent qu'il est plus difficile d'assembler des neurones de McCulloch et Pitts, que d'écrire un programme pour une machine de von Neumann.

---

1 **McCulloch W., Pitts W. L.R.:** *A logical calculus for the ideas immanent in nervous activity* - Bull. Math. Biophysics, 5, pp 115-133, (1943)

Apparaissent ensuite les premières machines adaptatives, c'est à dire capables d'ajuster leurs propres réglages à l'aide d'exemples. Le perceptron [1] et l'adaline [2] sont deux machines à l'époque extrêmement séduisantes. La légende veut qu'elle sombrent dans l'oubli après la parution de l'ouvrage de Minsky et Papert [3], qui exhibait au grand jour leurs déficiences.

C'est au contraire une période de grand succès, sous d'autres noms. Des dérivés de l'adaline, sont aujourd'hui fréquemment utilisé en traitement du signal [4]. Les systèmes adaptatifs ont subi dans ce contexte des traitements théoriques raffinés (voir [5], par exemple) qui ont relégué au second plan leur lointaine parenté avec les neurones formels de McCulloch et Pitts.

Une grande partie de la théorie statistique des systèmes adaptatifs est l'œuvre de l'école russe: Dans les années 70, elle a été étendue à tous les domaines d'applications revendiqués aujourd'hui par les réseaux connexionnistes. Les ouvrages de Tsypkin [6] [7], pourtant cités par Kohonen, restent malheureusement peu connus des connexionnistes. Ils constituent une référence impressionnante sur ce sujet. Vapnik [8] énonce à la même époque un théorème de convergence uniforme qui aborde de façon rigoureuse les problèmes de généralisation.

---

1 **Rosenblatt F.:** *The Perceptron: a perceiving and recognizing automaton* - Project PARA, Cornell Aeronautical Lab. Report 85-460-1. (january 1957)

2 **Widrow B., Hoff M.E.:** *Adaptive switching circuits* - IRE WESCON Conv. record, part 4 1960, pp 96-104 (1960)

3 **Minsky M., Papert S.:** *Perceptrons* - MIT Press (1968)

4 **Widrow B., Stearn S.D.:** *Adaptive signal processing*- Prentice Hall (1985)

5 **Ljung L., Söderström T.,** *Theory and Practice of Recursive Identification* - MIT Press (1983)

6 **Tsypkin Ya.:** *Adaptation and Learning in Automatic systems* - Mathematics in science and engineering, vol 73, Academic Press, (1971)

7 **Tsypkin Ya.:** *Foundations of the Theory of Learning Systems* - Mathematics in science and engineering, vol 101, Academic Press, (1973)

8 **Vapnik V.N., Chervonenkis A.Ya.:** *On the uniform convergence of relative frequencies of events to their probabilities* - Theory of Probability and its Applications, vol 16, n°2, pp 264-280 (1971)

Les algorithmes adaptatifs ont également survécu dans les travaux sur les mémoires associatives de Kohonen [1,2] en Finlande et d'Amari [3] au Japon. Associés à une meilleure connaissance des domaines d'applicabilité des approches symboliques, ces travaux constituent les germes du regain des années 80.

Il fallut pour cela attendre le modèle de Hopfield [4]. Très proche des travaux d'Amari, ce modèle n'échappe pourtant pas aux limitations théoriques du perceptron. Peu après, deux nouveaux algorithmes franchissaient ces limites: La machine de Boltzmann [5] et le perceptron multi-couches [6]. Ce dernier, donna lieu très vite à des réalisations [7] qui ont assuré sa promotion.

ii Développements des perceptrons multi-couches en France.

L'algorithme d'apprentissage des perceptrons multi-couches, appelé rétro-propagation du gradient, avait (et a conservé) une réputation de lenteur désespérante. La modestie des crédits de recherche français à cette époque fit merveille. Le Cun [8] qui travaillait sur un algorithme similaire (HLM), établit alors une version stochastique de la rétro-propagation du gradient capable de tourner sur un micro-ordinateur d'alors.

De multiples améliorations de détail ont été apportées depuis: On constate de nos jours un facteur supérieur à 1000 entre les temps d'apprentissage de la procédure de LeCun, et les rétro-propagations trop simples parfois décrites dans la littérature. Ces accélérations permettaient d'entraîner des réseaux de taille assez importante pour envisager d'aborder des problèmes réels.

---

1 **Kohonen T.:** *Content addressable memories* - Springer series in information sciences, vol 2, Springer Verlag (1981)

2 **Kohonen T.:** *Self organisation and associative memories* - Springer series in information sciences, vol 8, Springer Verlag (1984)

3 **Amari S.I.:** *Learning patterns and pattern sequences by self-organizing net of threshold elements* - IEEE Trans. computer, vol C-21, n°11, (november 1972)

4 **Hopfield J.J.:** *Neural networks and physical systems with emergent collective computational abilities* - P.N.A.S. USA, Vol 79, pp 2554-2558, (1982)

5 **Ackley D.H., Hinton G.E., Sejnowski T.J.:** *A learning algorithm for Boltzmann Machines* - Cognitive Science, 9, pp 147-169 (1985)

6 **Rumelhart D., Hinton G.E., Williams R.:** *Learning internal representations by error propagation* - in Parallel Distributed Processing, vol1: exploring the microstructure of cognition, D.Rumelhart, J.McClelland eds., MIT Press, (1986)

7 **Sejnowski T.J., Rosenberg, C.R.:** *NETtalk, a parallel network that learns to read aloud* - Tech Report 86-01, Dept of EE-CS, John Hopkins University, Baltimore MD - (1986)

8 **Le Cun Y.:** *Modèles Connexionnistes de l'Apprentissage*- Thèse de doctorat de l'Université de Paris 6, (1987)

1.2 Thème.

Début 1988, il apparaissait clairement au sein de la petite équipe animée par Françoise Fogelman que tester ces techniques sur des problèmes réels devenait un enjeu déterminant. Cette attitude était bien partagée dans le monde. Le connexionnisme s'exposait là à des révisions et mutations d'ampleur, dont il n'est pas encore sorti.

Cette thèse s'appuie sur un ensemble d'expériences connexionnistes en reconnaissance de la parole, réalisées dans le cadre des projets européens "Brain" et "Pygmalion", et en collaboration avec plusieurs laboratoires spécialisés, dont le LIMSI (Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur) [1] et le RSRE (Royal Signal and Radar Establishment, Malvern, UK) [2].

Maints enseignements ont bien sûr été retirés d'autres applications étudiées dans le laboratoire, et qui ne constituent pas le sujet de cette thèse.

1.2.1 Reconnaissance de la parole.

Parmi les applications auxquelles ont été appliquées des techniques connexionnistes, la reconnaissance de la parole tient une place de choix. Plusieurs facteurs y contribuent:

- La complexité du problème. L'information véhiculée par un signal de parole de haute qualité représente environ 300Kbits/s. L'information contenue dans la séquence de phonèmes correspondante est inférieure à 50 bits/s. Cette information est donc intimement mêlées temporellement et spectralement à des informations parasites, comme du bruit de fond, la voix du locuteur, ses trémolos, ses angines…

- L'existence d'une large communauté scientifique travaillant sur ce sujet. Ces chercheurs ont posé des problématiques, c'est à dire déterminé des ensembles de contraintes qui définissent un problème abordable. Un système peut être limité à un seul locuteur, à un bruit de fond réduit, à un petit vocabulaire. Pour ces problèmes, des solutions performantes ont été développées, et servent de points de repère.

---

1 **Bottou L., Fogelman Soulié F., Blanchet P., Liénard J.S.:** *Experiments with Time Delay Networks and Dynamic Time Warping for speaker independent isolated digits recognition*- Procs of EuroSpeech 89, vol II, pp 537-540 (1989)

2 **Bedworth M.D., Bottou L., Bridle J.S., Fallside F., Flynn L., Fogelman F., Ponting K.M., Prager R.W.:** *Comparison of neural and conventioal classifiers on a speech recognition problem.* In IEE 1st International Conference on Artificial neural networks, London, (1989)

Améliorer ces solutions consiste à réduire les contraintes associées à la reconnaissance: Par exemple, obtenir des systèmes résistant à un bruit de fond, au changement de locuteur, ou même à une modification de vocabulaire.

En fait, les systèmes de reconnaissance de la parole commettent rarement plus de 5% d'erreurs. Ils ne se distinguent que par les contraintes qu'il requièrent pour atteindre ce taux de performance. Les qualités de résistance aux bruit et d'adaptabilité des méthodes connexionnistes en font donc des candidats intéressant pour réduire certaines contraintes.

1.2.2 Révisions.

L'expérience des problèmes réels entraîne avec elle une série de révisions parfois cruelles. Plusieurs points émergent alors:

i La nécessité d'un fondement théorique.

On a souvent remarqué la remarquable résistance des modèles connexionnistes. Perturber leur fonctionnement ou leurs réglages ne détériore que lentement leurs performances.

A fortiori, les méthodes connexionnistes résistent fort bien à des algorithmes suboptimaux. Un algorithme connexionniste approximatif donnera tout de même des résultats, alors qu'un algorithme classique approximatif se traduit en général par un non fonctionnement du système.

Ce qui est un grand avantage pendant l'exploitation d'un système est une catastrophe pendant son développement: On n'a jamais la certitude que l'algorithme est programmé correctement, ou même qu'il est exploité à bon escient.

Ce point devient crucial si on désire comparer ces algorithmes à des approches classiques utilisées à bon escient, dont on peut tester la qualité de programmation.

Cette constatation est une condamnation de l'empirisme. Seule une théorie fondamentale des algorithmes adaptatifs peut apporter des garanties d'optimalité, des tests de bon fonctionnement, et garantir leur bonne utilisation.

ii La nécessité de coopérer avec des algorithmes classiques.

Une chaîne complète de reconnaissance de la parole, par exemple, est un ensemble complexe composé de modules de traitement du signal, de segmentation, de reconnaissance des formes, de traitement symbolique du langage…

Y insérer un dispositif connexionniste requiert de celui ci des techniques d'interface. Il faut au minimum pouvoir décrire ses entrées et ses sorties avec des notions adaptées aux modules placés en amont et en aval.

Fréquemment, plusieurs de ces modules connaissent des phases d'apprentissages. Deux stratégies sont possibles. On peut les *entraîner séparément*, avec des exemples pour lesquels on peut fournir les résultats intermédiaires idéaux que doivent générer les divers modules de notre chaîne de traitement. Disposer de tels exemples est rare; on peut désirer *entraîner coopérativement* tous ces modules, à collaborer entre eux.

La encore, peuvent se poser des question d'optimalité: Quelle influence ont ces stratégies d'apprentissage ou d'adaptation sur le comportement global du système?

1.3 Plan de la thèse.

Deux approches pouvaient être envisagées pour rédiger cette thèse. La première consistait simplement à suivre l'ordre chronologique des expérimentations et de l'apparition de questions d'ordre théoriques. Le matériau existait, il suffisait de le lier pour relater une fastidieuse série de dispositifs expérimentaux, de résultats parfois encourageants, souvent décevants, et de développements sur des questions soulevées par quelques détails.

Il était plus attirant de recréer un arrangement thématique, plus logique pour le lecteur. Cela revient à commencer par la fin: les applications ont soulevé des questions théoriques et des recherches bibliographiques qui aboutirent parfois à apporter des explications.

S'appuyer sur la théorie statistique des algorithmes adaptatifs permet de voir les expériences sous une lumière nouvelle, de comprendre les intuitions d'autrefois, d'identifier les erreurs qu'elles contiennent.

Cette thèse peut être divisée en trois parties:

- Une première partie présente les méthodes connexionnistes en établissant des liens entre l'approche usuelle et une vision globale des algorithmes adaptatifs.

- Une seconde partie s'attache à montrer les aspects pratiques de ces algorithmes, et leur utilisation pour le problème de reconnaissance de la parole. Celui ci requiert des systèmes d'apprentissage complexes, interprétés comme la coopération de plusieurs systèmes d'apprentissage.

- Dans une troisième partie, on étudie les problèmes spécifiques posés par la coopération d'algorithmes connexionnistes et de modèles de Markov.

Afin de faciliter la lecture, les sections les plus importantes sont précédées d'une notice exposant les principaux résultats, et la démarche développée.

1.3.1 Partie 1: Connexionnisme et apprentissage.

Le chapitre 2 est consacré à la présentation usuelle des algorithmes connexionnistes. Cette présentation culmine avec la description des modèles les plus utilisés, associés à autant d'algorithmes d'apprentissage distincts.

Le chapitre 3 présente un modèle général d'algorithme d'apprentissage et d'adaptation. Il contient une étude de la convergence asymptotique, qui fait appel aux méthodes classiques de martingales pour l'analyse des approximations stochastiques en traitement adaptatif du signal.

Le chapitre 4 abandonne l'étude asymptotique en supposant que l'on ne dispose que d'un nombre fini d'exemples tirés au hasard. Les théorèmes de convergence uniforme de Vapnik et Chervonenkis permettent en effet de tirer quelques conclusions.

Ces deux chapitres tentent de présenter de façon synthétique les problèmes d'apprentissage et de généralisation, afin de montrer comment ces deux problèmes sont intimement liés.

1.3.2 Partie 2: Application à quelques problèmes de reconnaissance de la parole, et critique.

Le chapitre 5 constitue un préliminaire essentiel: il fait le point des problèmes numériques associés aux algorithmes d'apprentissage et d'adaptation, et présente les solutions pratiques qui ont été adoptées. Parmi les sujets couverts, signalons l'utilisation des approximations stochastiques, le conditionnement des problèmes d'optimisation, et le développement des outils de simulation.

Le chapitre 6 présente divers aspects du problème de reconnaissance de la parole, ainsi que les solutions "classiques", incluant les prétraitements, et les dispositifs de reconnaissance. Une attention particulière est apportée à la théorie et l'utilisation des modèles de Markov cachés.

Dans le chapitre 7, sont présentés quelques approches connexionnistes. Celles ci traitent rarement de façon heureuse l'aspect séquentiel du signal. Le cas des TDNN (Time-Delay Neural Networks) y est largement développé, y compris un algorithme hybride mêlant un TDNN et une programmation dynamique.

Le chapitre 8, fruit d'une collaboration avec Michel de Bollivier, Patrick Gallinari et Sylvie Thiria, généralise cet algorithme hybride, en étudiant l'apprentissage coopératif des systèmes composés de plusieurs modules adaptatifs.

1.3.3 Partie 3: Algorithmes connexionnistes et modèles de Markov.

Le chapitre 9 rattache les modèles de Markov cachés à la famille des algorithmes d'adaptation et d'apprentissage identifiés au chapitre 3. Il présente ensuite les modèles de Markov discriminants préconisés par Bourlard & Wellekens.

Ces derniers combinent un perceptron multi-couches et un dispositif Markovien. Le chapitre 10 étudie en détail l'un de ces modèles, et souligne les difficultés que leur utilisation peut poser.

1.4 Chronologie.

Comme le plan de cette thèse ne suit absolument pas la chronologie, il m'a semble utile de donner au lecteur une idée grossière des étapes qui ont marquées le déroulement de ce travail.

La période 1986-1987 a été consacrée aux aspects pratiques et numériques, tels qu'ils sont décrits dans le chapitre 5. A cet effet, Le Cun et moi-même écrivîmes la première version du simulateur SN [1] au printemps 1987, qui systématisait et affinait les procédures de rétro-propagation stochastiques de Le Cun.

En 1988 et 1989, ont eu lieu la plus grande partie des expériences en reconnaissance de la parole relatées au chapitre 7, dans le cadre des projets "Brain" et "Pygmalion".

Cependant, le développement de systèmes réalistes de reconnaissance de la parole continue s'accommode mal des réseaux de l'époque. Petit à petit apparut la nécessité de nouveaux développements théoriques.

Ceux ci prennent forme en 1990. Les chapitres 3 et 4 ont en fait été rédigés en dernier lieu. Auparavant, une réflexion sur la reconnaissance de séquence produisait les remarques sur les Modèles de Markov Cachés des chapitre 6, 9 et 10. Ceux ci ont été rapprochés dans le chapitre 8 avec les travaux de Michel de Bollivier, Patrick Gallinari et Sylvie Thiria.

---

1 **Bottou L., Le Cun Y.:** *SN: Un simulateur pour réseaux connexionnistes* - Neuro Nîmes 88, pp 371-382, EC2 ed. (1988)

2 Techniques connexionnistes.

2.1 Qu'est ce que le connexionnisme ?

L'intérêt pour ce domaine mal défini est tel, ces temps-ci, que le label "connexionniste" est plus un moyen d'obtenir des subsides qu'un critère scientifique. Se risquer à définir le connexionnisme, c'est assurément exclure quelque prétendant, fort honorable au demeurant, et donc s'attirer son ressentiment.

On se limitera ici à donner quelques éléments…

2.1.1 Un courant des sciences humaines.

Avant les années 80, le terme connexionnisme n'était pas spécifiquement utilisé pour qualifier quelques algorithmes, mais un courant d'idées en psychologie.

Celle ci a longtemps culminé avec la théorie du *behaviorisme*, ou théorie des comportements. Un comportement humain, selon Watson (1878-1958), n'est que l'ensemble des réactions (musculaires, glandulaires…) de l'organisme au stimuli qu'il perçoit. Le strict positivisme de cette théorie, qui se refuse à dépasser le *fait expérimental*, lui apporta un succès que l'on mesure aux aménagements que les faits lui ont imposés.

L'aménagement le plus fondamental sera certainement la *Gestalttheorie*, ou théorie de la forme: Le behaviorisme morcellait le comportement en une somme de circuits stimulus-réaction. Von Ehrenfels (1859-1932) réintroduit la notion de structure: une mélodie n'est pas qu'un ensemble de notes; La

structure globale de la mélodie introduit des réactions propres. Les gestaltistes entament alors une théorie des structures privilégiées dans la perception (formes fortes et formes faibles).

Mais ce structuralisme statique ne rend pas compte de l'aspect *dynamique* de la perception. Piaget (né en 1896) étudie chez l'enfant la genèse de l'intelligence et de la perception. A ses yeux, elles mettent en œuvre des structures souples, auto-régulées à la manière des mécanisme biologiques.

Cette recherche d'un modèle à la fois *structurel* et *dynamique* donne naissance au *connexionnisme*, qui repose sur une vision distribuée des processus mentaux. Ceux ci n'ont plus lieu dans des modules spécifiques, mais apparaissent dans l'interaction globale de ces modules, et ceci jusque dans les processus neurobiologiques les plus élémentaires [1].

2.1.2 Un ensemble de techniques partageant quelques points communs.

Une nouvelle étape commence avec la création du "PDP Group" (pour Parallel Distributed Processing), groupe informel de recherche composé de connexionnistes, dont l'objectif était de montrer par quelques simulations le bien-fondé de leurs arguments. Leur deux premiers ouvrages [2] exposent l'un, quelques algorithmes censés s'approcher du traitement distribué prenant place dans le raisonnements humain, l'autre, les implications psychologiques que ces algorithmes représentent.

Le fait est que ces algorithmes s'avèrent intéressants en eux-mêmes. Le qualificatif "connexionniste" leur est resté, et a déteint sur quelques algorithmes apparentés issus des travaux de Rosenblatt, Widrow, Kohonen…

Cette origine oblique joue encore un rôle important dans le contenu scientifique et technologique de la facette algorithmique du connexionnisme. Celle ci est composée d'une juxtaposition de techniques dont les liens relèvent en fait de l'autre facette du connexionnisme:

- L'aspect parallèle et distribué rassemble ces techniques au travers de l'habitude de décrire les algorithmes au moyen de formalisations variées des neurones et de leurs interactions.

- L'aspect dynamique se traduit par la recherche d' algorithmes d'auto-régulation ou d'apprentissage.

---

1 **Changeux J.P.:** *L'homme neuronal*, Fayard, (1983)

2 **Rumelhart D., Mc Clelland J. (eds):** *Parallel Distributed Processing, vol1&2*, D.Rumelhart, J.McClelland eds., MIT Press, (1986)

- L'aspect structurel transparaît dans la préoccupation constante d'étudier l'auto-organisation de ces systèmes, et de rechercher une structure dans les représentations internes qu'ils génèrent.

Aucune unification algorithmique de ces techniques ne peut les englober toutes sans en inclure quelques autre qui ne relèvent pas au sens strict de la démarche connexionniste. Les prochains chapitres sont délibérément orientés vers l'aspect dynamique et les notions d'apprentissage et d'adaptation. Une foule de techniques statistiques usuelles, distribuées ou non, s'introduiront alors naturellement.

2.2 Petit lexique du connexionnisme.

2.2.1 Unités linéaires à seuil.

La description des algorithmes connexionnistes s'appuie presque toujours sur une modélisation plus ou moins fidèle des neurones et de leurs interconnexions. La modélisation la plus simple est celle de *l'automate à seuil*, introduit par McCulloch et Pitts [1] en 1943.

2.2.1.1 Définition.

Un neurone formel y est modélisé comme un automate possédant plusieurs entrées booléennes (X_i), et une sortie S booléenne également. Sa fonction consiste à effectuer une somme des entrées pondérée par des coefficients synaptiques ou *poids* w_i. Si cette somme est supérieure à un seuil θ, la sortie vaut +1 (vrai), sinon elle vaut 0 (faux).

$$S = \mathbb{1}_{\Re_+}\left(\sum_i w_i X_i - \theta \right) \qquad (2.1)$$

où $\mathbb{1}_{\Re_+}$ est la fonction caractéristique de $\Re_+$, dite *fonction de Heaviside*. On peut écrire cela vectoriellement, en notant $\mathbf{w}^T$ le transposé du vecteur $\mathbf{w}$.

$$S = \mathbb{1}_{\Re_+}\left(\mathbf{w}^T \mathbf{X} - \theta \right)$$

Cette définition se prête à plusieurs généralisations:

- Il est inutile de se limiter à des entrées booléennes. La formule (2.1) s'applique également à des entrées réelles, ce que l'on fera systématiquement dans la suite.

---

1 **McCulloch W., Pitts W. L.R.:** *A logical calculus for the ideas immanent in nervous activity* - Bull. Math. Biophysics, 5, pp 115-133, (1943)

- Le seuil θ peut être traité comme un poids, pourvu que l'on considère un vecteur d'entrée augmenté d'un élément -1. Cela simplifie tous les calculs, sans diminuer la généralité des conclusions. L'équation (2.1) peut s'écrire alors

$$S = \mathbb{1}_{\Re_+}\left(\sum_i w_i X_i \right) = \mathbb{1}_{\Re_+} \left(\mathbf{w}^T \mathbf{X} \right) \qquad (2.2)$$

- Rien ne s'oppose à choisir des sorties ailleurs que dans l'ensemble $\{0,1\}$. Un cas classique, dit symétrique, consiste à remplacer la fonction de Heaviside par une fonction signe, et donc d'avoir des sorties dans $\{-1,1\}$.

2.2.1.2 Séparabilité linéaire.

Un automate à seuil sépare l'espace de ses entrées en deux demi-espaces, séparés par l'hyperplan d'équation:

$$\sum_i w_i X_i - \theta = 0 \qquad (2.3)$$

Si l'on dispose d'un ensemble de points, soit bleus, soit rouges. Un automate à seuil ne pourra identifier leur couleur (les classer) que s'il est possible de séparer les points rouges des points bleus par un hyperplan. On dit alors que le problème de classification est *linéairement séparable*.

Les figures 2.1 et 2.2 montrent deux exemples classiques de problèmes séparables ou non séparables linéairement.

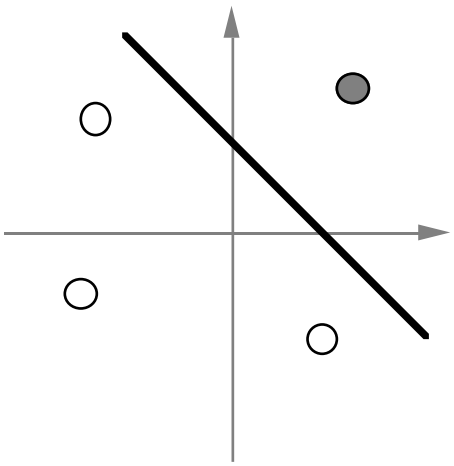


Fig 2.1- Un problème (à deux entrées) linéairement séparable. Si l'on considère les valeurs booléennes des signes des entrées, l'exemple ci dessus est un ET logique.

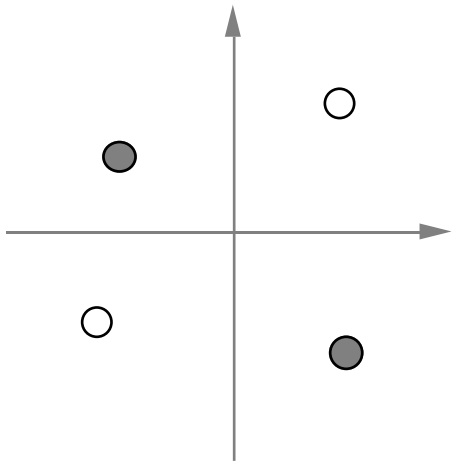


Fig 2.2- Un problème (à deux entrées) non séparable linéairement. Si l'on considère les valeurs booléennes des signes des entrées, l'exemple ci dessus est un OU EXCLUSIF ou XOR.

Afin d'avoir une idée des capacités d'un séparateur linéaire, il est pratique de compter $m_n(L)$, le nombre maximum de séparations possibles de la forme $\{\mathbf{w}^t\mathbf{x}-\beta>0\}$ dans un ensemble de L points en dimension n.

On peut calculer $m_n(L)$ par récurrence. Il est aisé de vérifier que

$$m_1(L)=2L \quad et \quad m_n(1)=2$$

Chaque point $\mathbf{x}_0$ définit un hyperplan vectoriel $\mathbf{w}^t\mathbf{x}_0-\beta=0$ de l'espace des paramètres $(\mathbf{w},\beta)$. Les L points fournissent donc une partition de cet espace de dimension $n+1$ par L hyperplans vectoriel. Chaque composant de cette partition regroupe les poids qui correspondent à une même dichotomie des L points.

Par définition, donc, $m_n(L)$ est égal au nombre maximum de régions séparées par L hyperplans vectoriel dans un espace de dimension $n+1$. Les intersections de ces L hyperplans et l'hyperplan vectoriel associé à un $L+1^{ème}$ point définissent dans ce dernier $m_{n-1}(L)$ régions. Chacune partage en deux une des régions de l'espace des poids définies par les L hyperplans. On a alors la récurrence suivante [1]:

$$m_n(L+1) = m_n(L) + m_{n-1}(L)$$

On vérifie alors que

$$m_n(L) = 2^L \ \ si \ \ L\leq n+1 \ , \ et \quad m_n(L) = 2\sum_{i=0}^{n} C_{L-1}^{i} \ < 2^L \ si \ \ L>n+1 \qquad (2.4)$$

---

1 **Cover T.M.:** *Geometrical and statistical properties of linear inequalities with application to pattern recognition* - IEEE Trans. Electronic Computer, Vol EC-14, n°3, pp 326-334, (1965)

En dimension n, un ensemble de plus de $n+1$ points ne peut être partagé de toutes les façons possibles par un séparateur linéaire. Lorsque L est grand, $m_n(L)$ est majoré par un polynôme degré $n+1$ en L.

Sur les 2^{2^n} fonctions booléennes à n entrées, *au plus* $m_n(2^n) \leq k(2^n)^{(n+1)}$ sont linéairement séparables, c'est à dire une proportion infime. Ceci apparaît comme une limitation importante, qui a justifié à l'époque l'abandon des recherches sur les perceptrons.

Cependant, ces résultats prouvent également que la dimension de Vapnik (cf. chp. 4) et Chervonenkis [1] des séparateurs linéaires est $n+1$, ce qui est a contrario la clef de leur capacité de généralisation!

2.2.1.3 Règles locales d'apprentissage.

Reste maintenant à déterminer les poids qui permettent de séparer nos points bleus de nos points rouges. Cela se fait par un processus itératif, consistant à ajuster les poids à chaque présentations de couples entrées-sorties $\left((X^*_i),\ S^*\right)$.

Deux règles d'ajustement des poids ont pris une importance particulière:

i La règle de Hebb.

La règle de Hebb est inspirée par [2]. Elle consiste à renforcer d'autant plus un poids que cette entrée est corrélée avec la sortie. Elle s'écrit

$$\Delta w_i = \varepsilon\, S^* \cdot X^*_i \qquad (2.5)$$

où ε est un nombre positif quelconque. La limitation de cette règle est la supposition implicite que les entrées (X^*_i) sont elles-mêmes décorrélées.

ii La règle du Delta ou de Widrow-Hoff.

La règle du Delta consiste à rapprocher progressivement la somme pondérée (2.2) de la sortie désirée S^*. On cherche donc à réduire la distance

$$C_{\mathbf{x}} = (S^* - \sum_k w_k\, X^*_k\ )^2 \qquad (2.6)$$

au moyen d'un algorithme de gradient. La règle du Delta s'écrit:

---

1 **Vapnik V.N., Chervonenkis A.Ya.:** *On the uniform convergence of relative frequencies of events to their probabilities* - Theory of Probability and its Applications, vol 16, n°2 pp 264-280 (1971)

2 **Hebb D.:** *Organization of behavior*, Science Edition (1961)

$$\Delta w_i = -\frac{\varepsilon}{2}\frac{\partial C_{\mathbf{x}}}{\partial w_i} = \varepsilon\,(S^* - \sum_k w_k X^*_k)\cdot X^*_i \qquad (2.7)$$

où ε est un nombre positif supposé faible. Si l'on développe l'expression (2.7), on obtient

$$\Delta w_i = \varepsilon\, S^*\cdot X^*_i - \sum_k w_k X^*_k\cdot X^*_i \qquad (2.8)$$

Les produits croisés $X^*_k X^*_i$ sont en moyenne nuls si les entrées sont décorrélées. On retrouve alors, *en moyenne*, la règle de Hebb.

2.2.1.4 Unités linéaires et sigmoïdes.

Dans l'équation (2.6), on a cherché à rapprocher la somme pondérée de la sortie idéale, et non la sortie réelle. Cette dernière ne prend en effet que deux valeurs, et il est donc impossible de la rapprocher *progressivement* de la sortie désirée.

On se livre donc souvent à une généralisation de la formule (2.2)

$$S = f\left(\sum_i w_i X_i\right) \qquad (2.9)$$

dans laquelle f est une fonction de $\Re$ dans $\Re$ que l'on appelle *fonction de transfert*. On utilise fréquemment trois types de fonctions (figure 2.3):

- Les fonctions à seuil, dont deux exemples sont la fonction de Heaviside et la fonction signe.
- La fonction identité. On parle alors d'unités linéaires.
- Les fonctions sigmoïdes. Ce sont simplement des fonctions à seuil molles, comme la tangente hyperbolique. On parle alors d'unités sigmoïdes.

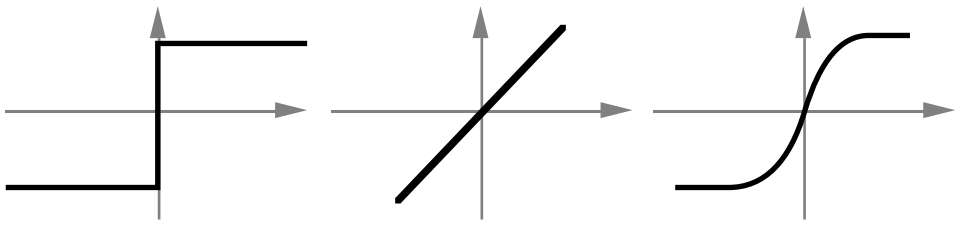


Fig 2.3 - Trois types de fonctions de transfert: une fonction à seuil, la fonction identité et une fonction sigmoïde

La règle du Delta revient donc à minimiser l'écart entre la sortie désirée et la sortie d'une unité linéaire. On peut également minimiser l'écart entre la sortie désirée et la sortie d'une unité sigmoïde, ce qui se rapproche de ce qu'il faudrait faire: minimiser l'écart entre la sortie désirée et la sortie d'une unité à seuil.

2.2.2 Autres types d'unités élémentaires.

Il n'y a pas de raisons apparentes de se limiter au unités quasi-linéaires. Dans [1] sont proposées des cellules *sigma-pi*, qui en fait calculent un polynôme de degré d. On a alors, en degré 2:

$$s = f\left(\sum_{i,k} w_{ik} X_i X_k + \sum_i b_i X_i + \theta\right) \qquad (2.10)$$

Une restriction importante de ce cas est représentée par les cellules qui calculent des distances, utilisée par exemple par [2].

$$s = f\left(\sum_{i,k} (w_i - X_i)^2\right) \qquad (2.11)$$

Ont également été proposées des cellules calculant des produits généralisés [3], à l'aide d'exponentielles

$$s = e^{\sum w_i \log(X_i)}$$

Toutes ces unités possèdent un algorithme local d'apprentissage comparable à la règle du delta. Il suffit chaque fois de minimiser l'écart entre la sortie désirée et la sortie réelle.

2.2.3 Réseaux et connexions.

Lorsque plusieurs de ces unités sont cascadées, les sorties des unes servant d'entrées aux autres, on parle de *réseaux* de neurones. On dit alors fréquemment que les poids sont associés aux connexions entre unités, et non aux unités elles-mêmes.

---

1 **Rumelhart D., Hinton G.E., Williams R.:** *Learning internal representations by error propagation* - in Parallel Distributed Processing, vol1: exploring the microstructure of cognition, D.Rumelhart, J.McClelland eds., MIT Press, (1986)

2 **Kohonen T., Barna G., Chrisley R.:** *Statistical Pattern Recognition with neural networks: Benchmarking Studies* - Second International Conference on Neural Networks, San Diego, IEEE proc. of ICNN 88, vol I, pp 61-68, (1988)

3 **Durbin R., Rumelhart D.E.:** *Product Units: A Computationally Powerful and Biologically Plausible Extension to Backpropagation Networks* - Neural Computation 1, pp 133-142, MIT Press (1989)

i Connexions complètes.

Il existe beaucoup de façons de relier les sorties et entrées de ces unités. Les plus simples sont les *connexions complètes*. Chaque unité possède autant d'entrées qu'il y a d'unités; chacune est connectée à la sortie d'une unité différente (fig 2.4).

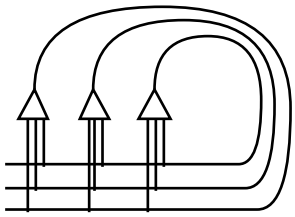


Fig 2.4 - Des connexions complètes

ii Boucles, et séquencement.

On dit qu'il a des boucles lorsque les sorties des certaines unités servent d'entrée à des unités situées en amont. C'est en particulier le cas pour des connexions complètes.

Se pose alors un problème de séquencement. A quels moment faut il recalculer la sortie de chaque unité?

- Il est facile de simuler le *séquencement asynchrone* en tirant au hasard à chaque instant la prochaine unité à recalculer.

- Le *séquencement synchrone* consiste à mettre à jour à chaque instant la sortie de toutes les cellules, en utilisant comme entrées les sorties de l'instant précédent.

On s'affranchit souvent des problèmes de séquencement en utilisant des motifs de connexions sans boucles, comme les connexions en couches.

iii Connexions en couches.

Dans le schéma *en couches* (fig 2.5), on regroupe les unités en couches successives. Chaque unité est connectée à toutes les unités de la couche suivante. Les unités d'une même couche ne sont pas connectées entre elles. La première couche s'appelle alors *couche d'entrée*, la dernière *couche de sortie*, les autres sont dites *couches cachées*.

On peut noter que la couche d'entrée ne sert à rien, si ce n'est d'introduire des données dans le réseau. Certains préfèrent ne pas l'appeler "couche", et ne pas appeler ses éléments "unités", ce qui est d'ailleurs plus logique.

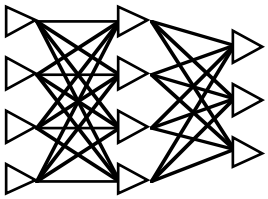


Fig 2.5- Des connexions en couches.

Il y a des variantes: on peut tolérer des connexions incomplètes d'une couche à l'autre, ou même des sauts: Quelques unités d'une première couches peuvent être connectées directement à n'importe quelle couche supérieure.

On peut également autoriser à nouveau des boucles, ou *feed-back* (fig 2.6)

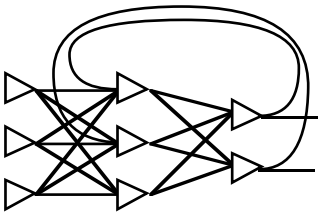


Fig 2.6 - Un réseau en couches, avec feed-back.

iv Connexions locales, poids partagés.

Les *connexions locales* sont une variante importante du modèle en couche. On dote chaque couche d'une topologie, et on ne connecte chaque cellule qu'aux cellules voisine de la couche précédente (fig 2.7).

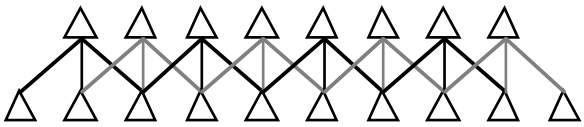


Fig 2.7 - Une couche de connexions locales, monodimensionnelles, avec des fenêtres de 3 unités se recouvrant de deux unités.

Mais le nec plus ultra sont les réseaux à *connexions contraintes,* ou à *poids partagés*. Il s'agit d'une simple adaptation du modèle en couche, mais on impose à certaines connexions d'être associées à des poids identiques.

Dans la figure 2.7, si on impose au trois poids et seuils de chaque unité d'être identique d'une unité à l'autre, on obtient un traitement invariant par translation, ce qui est souvent utile.

2.2.4 Synthèse.

En fin de compte, toute cette terminologie n'est qu'une façon de décrire un calcul numérique répétitif. Chaque unité est en fait une variable intermédiaire, les connexions représentent le graphe des dépendances entre ces variables.

Dans le cas des réseaux sans boucle, chaque combinaison d'une architecture de connexions et de modèles d'unités définit une classe de fonctions de $\Re^n$ dans $\Re^p$.

Plus que de connaître quelles fonctions appartiennent à une classe donnée, il nous intéresse de savoir quelle est l'adhérence de cette classe. C'est à dire quelles fonctions peuvent être approchée indéfiniment par une fonction de cette classe.

L'un des théorèmes les plus connus pour ce faire est le théorème de Stone-Weierstraß, qui montre par exemple que l'on peut approcher n'importe quelle fonction continue sur un compact avec un polynôme.

Certaines classes de réseaux possèdent des propriétés comparables. Cybenko [1] a démontré que toute fonction continue de $[-1,+1]^n$ dans $\Re$, pouvait être approchée uniformément par un réseau en couche possédant une couche cachée d'unités sigmoïdes et une couche de sortie d'unités linéaire.

Plus exactement, l'ensemble des fonctions de la forme

$$G(\mathbf{x}) = \sum_{k=1}^{N} v_k \; f(\mathbf{W}_k{}^T \mathbf{x} + \theta_k) \qquad (2.12)$$

où f est une fonction sigmoïde, est dense dans l'ensemble des fonctions continues sur $[-1,+1]^n$.

2.3 Quelques modèles classiques

Quelques modèles connexionnistes connus sont présentés dans cette section. Pour chaque modèle, on décrit non seulement l'organisation des unités en réseaux, mais aussi leurs algorithmes d'apprentissage.

---

1 **Cybenko G.:** *Approximation by Superpositions of a Sigmoidal Function.* Math. Control Systems Signals, vol 2, pp 303-314 (1989)

2.3.1 Perceptrons et Adalines.

Le perceptron et l'adaline sont les deux grandes tendances du début des années 60. Ils illustrent deux problématiques différentes, mais aboutissant à deux algorithmes extrêmement ressemblants.

2.3.1.1 Le Perceptron.

Introduit en 1957 par Rosenblatt [1], le perceptron est composé:

- D'une *rétine* dont le rôle est de capter l'information brute $\mathbf{x}$ de l'extérieur.
- D'une *aire d'association* qui effectue des prétraitements sur les données de la rétine. Ces prétraitements peuvent être représentés par une fonction $\varphi(\mathbf{x})$ quelconque .
- Et d'une *unité de décision*, chargée d'effectuer une classification sur ces données. Cette unité de décision est un automate linéaire à seuil. C'est la seule partie du perceptron soumise à l'apprentissage, grâce à un dérivé de la règle du Delta.

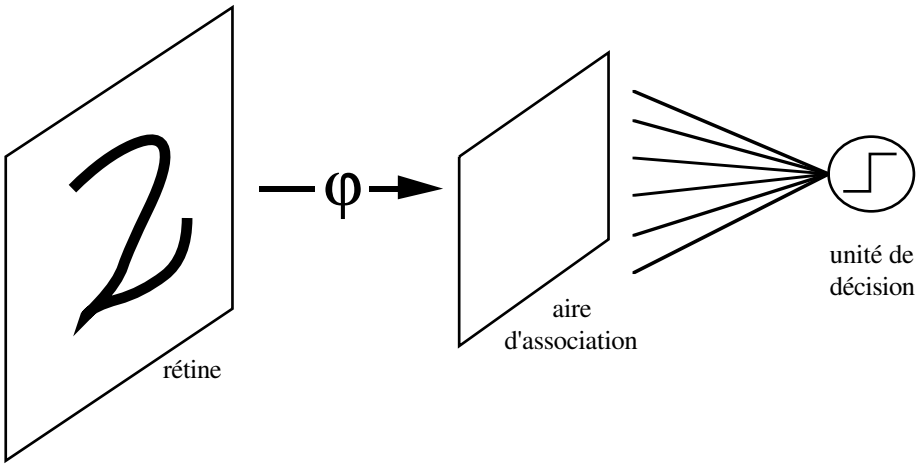


Fig 2.8 - Un perceptron.

L'algorithme d'apprentissage du perceptron ne repose pas sur une mesure quadratique de la distance (2.6), mais sur le fonction

$$C_{\mathbf{x}} = -\left(S^* - \mathbb{1}_{\Re+}(\mathbf{w}^T \varphi(\mathbf{x}))\right) \mathbf{w}^T \varphi(\mathbf{x}) \qquad (2.13)$$

qui présente la propriété d'être nulle dès que l'exemple $\mathbf{x}$ est bien classé, et positive dans le cas contraire. Comme dans le cas de la règle du Delta, en dérivant cette distance, et en *négligeant la discontinuité* de la fonction de Heaviside, on obtient la règle du Perceptron:

---

1 **Rosenblatt F.:** *The Perceptron: a perceiving and recognizing automaton* - Project PARA, Cornell Aeronautical Lab. Report 85-460-1. (january 1957)

$$\Delta \mathbf{w} = \varepsilon \, (S^* - \mathbb{1}_{\Re+}(\, \mathbf{w}^T \, \varphi(\mathbf{x})) \,) \cdot \varphi(\mathbf{x}) \qquad (2.14)$$

Cette règle d'apprentissage retenue pour l'unité de décision présente la propriété de trouver en temps fini une séparation linéaire, si c'est possible. De nombreuses démonstrations de cette convergence existent, reprises dans [1].

Signalons une propriété importante de la formule (2.14): Il n'y a d'adaptation des poids que si l'entrée $\mathbf{x}$ est mal classée. Cela signifie que l'hyperplan séparateur trouvé par le perceptron peut être fort voisin d'un exemple, provoquant alors de mauvaises classifications si les entrées sont bruitées.

De plus, lorsque la dimension de $\varphi(\mathbf{x})$ augmente, les problèmes linéairement séparables deviennent plus rares. Si l'on soumet à un perceptron un problème non linéairement séparable, l'algorithme d'apprentissage ne converge plus. Il n'y a aucun moyen de trouver une solution approchée optimale.

2.3.1.2 L'Adaline.

Ces deux inconvénients majeurs du perceptron proviennent tous deux de l'impossibilité de trouver *une solution optimale*,

- soit lorsque le problème est linéairement séparable, en ne trouvant pas un hyperplan séparateur à égale distance des deux classes.
- soit lorsqu'il ne l'est pas, en ne convergeant pas vers une solution approchée imparfaite, mais le moins possible.

Cela est du en fait au manque de finesse de la distance (2.13), qui est identiquement nulle dès qu'il y a bonne classification.

Dans l'adaline, Widrow [2] introduisit la distance quadratique et la règle du Delta. L'algorithme d'apprentissage de l'adaline consiste donc en la simple application de la règle du Delta:

$$\Delta \mathbf{w} = \varepsilon \, (S^* - \mathbf{w}^T \, \mathbf{x} \,) \cdot \mathbf{x} \qquad (2.15)$$

Dans certains cas, l'adaline ne trouve pas une séparation linéaire (Fig 2.9). Dans la majorité des cas, cependant, elle trouve une séparation linéaire à égale distance des classes. Si le problème n'est pas linéairement séparable, l'adaline converge tout de même vers une solution raisonnable.

---

1 **Duda R.O., Hart P.E.:** *Pattern classification and Scene analysis* - Wiley , (1973).

2 **Widrow B., Hoff M.E.:** *Adaptive switching circuits* - IRE WESCON Conv. record, part 4 1960, pp 96-104 (1960)

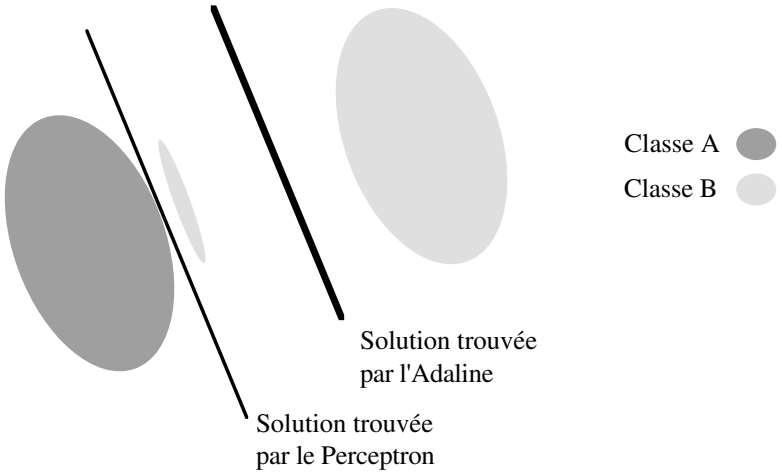


Fig 2.9 - Solutions trouvées par l'Adaline et le Perceptron pour un problème linéairement séparable complexe. Le Perceptron minimise le nombre d'erreur, l'Adaline détermine la meilleure séparation robuste entre les classes.

L'algorithme de l'Adaline est certainement l'archétype des techniques utilisées pour le traitement adaptatif du signal: filtrage adaptatif, annulation d'écho, boucle à verrouillage de phase…

2.3.2 Mémoires associatives linéaires.

Les mémoires associatives linéaires ont été principalement étudiées par Kohonen [1] [2] [3]. Considérons un ensemble d'unités linéaires de sorties $y(k)$ partageant les mêmes entrées $x(i)$.

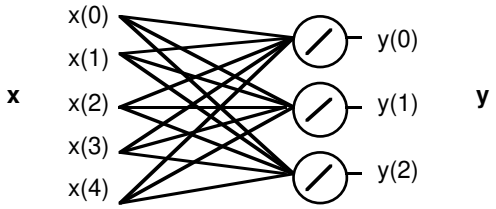


Fig 2.10 - Un associateur linéaire.

Cet ensemble permet d'effectuer des associations entre un vecteur d'entrée $\mathbf{x}$ et un vecteur de sortie $\mathbf{y}$. On appelle cela une *mémoire associative*.

---

1 **Kohonen T., Ruohonen M.:** *Representation of associated data by matrix operators* - IEEE Trans Computers (july 1973).

2 **Kohonen T.:** *An adaptive associative memory principle* - IEEE Trans Computers (April 1974)

3 **Kohonen T.:** *Self organisation and associative memories* - Springer series in information sciences, vol 8, Springer Verlag (1984)

Lorsque les entrées doivent être associées à elles mêmes, on appelle cela une *mémoire auto-associative*. L'intérêt d'une mémoire auto-associative réside dans sa résistance au bruit: on espère que présenter une entrée bruitée produira en sortie une version non bruitée des entrées.

Si on dispose de N exemples d'associations, déterminer la matrice de poids $\mathbf{W}$ revient à résoudre le système

$$\forall\, i \in \{ 1,\ldots N \}, \quad \mathbf{W}\, \mathbf{x}_i = \mathbf{y}_i \tag{2.16}$$

que l'on peut écrire à l'aide des matrices d'exemples $\mathbf{X}$ et $\mathbf{Y}$:

$$\mathbf{W}\, \mathbf{X} = \mathbf{Y} \tag{2.17}$$

La matrice $\mathbf{X}$ n'est pas en général carrée, donc n'est pas en général inversible. Cependant, on démontre qu'il existe une et une seule matrice $\mathbf{X}^+$, appelée *pseudo-inverse*, telle que

$$\mathbf{X}\mathbf{X}^+\mathbf{X} = \mathbf{X}, \quad \mathbf{X}^+\mathbf{X}\mathbf{X}^+ = \mathbf{X}^+, \quad \mathbf{X}^+\mathbf{X} \text{ et } \mathbf{X}\mathbf{X}^+ \text{ sont symétriques} \tag{2.18}$$

On démontre également que choisir une matrice de poids

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^+ + \mathbf{Z}\,(\,\mathbf{I} - \mathbf{X}\mathbf{X}^+\,) \tag{2.19}$$

où $\mathbf{Z}$ est une matrice quelconque de même taille que $\mathbf{W}$, minimise l'erreur quadratique moyenne:

$$\left|\, \mathbf{W}\ \mathbf{X} - \mathbf{Y} \,\right|^2 = \langle\, (\mathbf{W}\ \mathbf{x} - \mathbf{y})^2 \,\rangle \tag{2.20}$$

La solution $\mathbf{W}$ de norme minimale correspond au cas $\mathbf{Z}=\mathbf{0}$.

Il existe un algorithme récursif, l'algorithme de Greville [1], qui permet de calculer exactement $\mathbf{X}^+$. Cependant, il est souvent plus simple d'utiliser l'algorithme de Widrow-Hoff, (i.e. règle du Delta), qui minimise *en moyenne* l'erreur (2.20), et consiste à itérer:

$$\Delta\mathbf{W} = \varepsilon\, (\mathbf{y}-\mathbf{W}\mathbf{x})\, \mathbf{x}^T \tag{2.21}$$

où $(\mathbf{x},\mathbf{y})$ est un exemple d'association tiré au hasard. Cette procédure ne garantit cependant pas que l'on va obtenir une solution de norme minimale.

Pour ce faire, il suffit d'ajouter à chaque itération un léger bruit gaussien sur l'entrée $\mathbf{x}$. La valeur de la moyenne (2.20) sera donc augmentée de la variance de l'image par la matrice $\mathbf{W}$ de ce bruit gaussien, qui est minimale lorsque $\mathbf{W}$ est de norme minimale.

---

1 **Greville T.N.E.:** *Some applications of the pseudo inverse of a matrix*- SIAM Rev 2, pp 15-22 (1960)

2.3.3 Réseaux de Hopfield.

Considérons maintenant un réseau d'unités à seuils totalement connectées. Le calcul de l'état d'un tel réseau pose un problème de séquencement (cf. §2.2.3.*ii*). On appelle *trajectoire* l'évolution de l'ensemble des sorties des unités dans le temps. Un tel système converge-t-il vers un état stable ?

Si la matrice des poids $\mathbf{W}$ est *symétrique*, on montre [1] que la fonction

$$H(\mathbf{s}) = -\,\mathbf{s}^t\,\mathbf{W}\,\mathbf{s} = -\sum_{i,j} w_{ij}\, s_i\, s_j \qquad (2.22)$$

est décroissante sur la trajectoire, aussi bien dans le cas asynchrone que dans le cas synchrone. On en conclut que l'évolution du réseau se termine toujours dans un état stable, minimum local de l'énergie H.

On peut mettre à profit cette propriété pour stocker K vecteurs d'états $\mathbf{s}^{(k)}$, et réaliser ainsi une *mémoire auto-associative*. Il suffit de choisir les poids $\mathbf{W}$ de telle sorte que ces vecteurs, les *prototypes*, correspondent à un minimum local de l'énergie H.

On arrive parfois à ce résultat en appliquant la règle de Hebb, qui consiste en fait à réduire *en moyenne* la fonction de coût suivante

$$C = \left\langle \mathbf{H}(\mathbf{s}^{(k)}) \right\rangle = -\frac{1}{K}\sum_{k=1}^{K}\sum_{i,j} w_{ij}\, s_i^{(k)}\, s_j^{(k)} \qquad (2.23)$$

en effectuant, pour chaque exemple k

$$\Delta w_{ij} = -\frac{\partial}{\partial w_{ij}}\mathbf{H}(\mathbf{s}^{(k)}) = s^{(k)}_i\, s^{(k)}_j$$

En sommant cette équation sur tous les exemples, on obtient:

$$w_{ij} = \sum_{k=1}^{K} s^{(k)}_i\, s^{(k)}_j \qquad (2.24)$$

Si K reste faible devant le nombre de cellules, la fonction H possède un minimum local par prototype. Malheureusement, lorsque K augmente, apparaissent des *états parasites*, c'est à dire des états stables autres que les prototypes.

---

1 **Hopfield J.J.:** *Neural networks and physical systems with emergent collective computational abilities* - P.N.A.S. USA, Vol 79, pp 2554-2558, (1982)

Ce phénomène peut être réduit en calculant les poids au moyen d'une pseudo-inverse [1]. On obtient alors un système équivalent à une classique mémoire auto-associative linéaire.

Les modèles de Hopfield ont donné lieu à de nombreux développements théoriques utilisant des méthodes empruntées à la physique statistique ([2] par exemple), visant surtout à déterminer et améliorer leur capacité, c'est à dire le nombre d'états stables possibles pour un nombre d'unités donné.

2.3.4 Perceptrons multi-couches.

2.3.4.1 La rétro-propagation du gradient

Aucun des modèles ci-dessus n'est capable de traiter correctement des problèmes non séparables linéairement. Pour s'affranchir de cette contrainte, il faut un algorithme d'apprentissage pour réseaux multi-couches. La *rétro-propagation du gradient* est l'un de ces algorithmes

Un certain trouble subsiste cependant quant à savoir qui l'a inventé. Les publications habituellement citées sont [3], [4] et [5]. Les équations de la rétro-propagation étaient cependant déjà enseignées en contrôle dans [6], et avait été utilisées dans [7]. En fait, l'idée de génie appartient certainement à Leibniz (1675) et Newton (1687), qui ont inventé le calcul différentiel et par conséquent ce que l'on appelle aujourd'hui "règle de dérivation des fonctions composées".

ì *Le "credit assignment problem".*

Cela posait un problème en apparence insoluble: le *credit assignment problem*. On se donne un réseau en couches, et un ensemble d'exemples composés de paires entrées-sorties $\{(\mathbf{x}_k,\mathbf{s}_k)\}$.

---

1 **Personnaz L., Guyon I., Dreyfus G.:** *Collective computational properties of neural networks: new learning mechanism* - Phys. Rev. A, vol 34, pp 4217-4228, (1986)

2 **Peretto P.:** *Collective properties of neural networks: a statistical physics approach.* - Biological cybernetics. n°50, pp 51-62 (1984)

3 **Rumelhart D., Hinton G.E., Williams R.:** *Learning internal representations by error propagation* - in Parallel Distributed Processing, vol1: exploring the microstructure of cognition, D.Rumelhart, J.McClelland eds., MIT Press, (1986)

4 **Werbos P.J.:** *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD Thesis, Harvard University, Cambridge, MA.

5 **Parker D.B.:** *Learning logic* - Technical report TR-47, Sloan School of Management, MIT, Cambridge, MA

6 **Bryson A.E.Jr., Yu Chi Ho:** *Applied Optimal Control*, Blaisdel Publishing Company, (1969)

7 **Amari S.I:** *A theory of adaptive pattern classifiers* - IEEE Trans. on Elec. Com., EC16, pp 279-307, (1967)

Lorsqu'une sortie pour un exemple donné $\mathbf{x}_k$ est différente de la sortie désirée $\mathbf{s}_k$, faut-il modifier les poids de l'unité de sortie en faute, ou bien faut il tenter de modifier les poids d'une unité en amont, de façon à modifier sa sortie et donc de modifier la sortie de l'unité en tort?

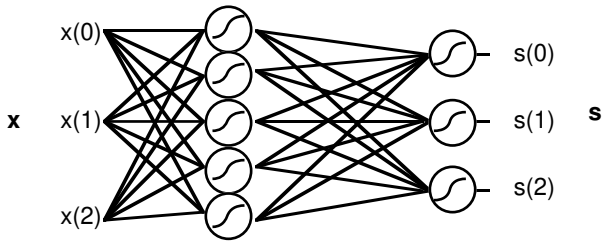


Fig 2.11- Un perceptron multi-couches avec 3 entrées, une couche de 5 unités cachées et 3 unités de sortie

L'algorithme de *rétro-propagation du gradient* apporte à ces questions une solution d'une simplicité déroutante: si l'on remplace les unités à seuil par des unités sigmoïdes, la fonction réalisée par le réseau est dérivable, et donc la distance entre la sortie réelle et la sortie désirée l'est aussi.

La rétro-propagation du gradient consiste donc à réduire une distance quadratique au moyen d'une descente de gradient dans l'espace des poids, qui repose sur un calcul astucieux des dérivées partielles.

ii Notations.

On considère un ensemble $\{k\}$ d'unités sigmoïdes, constituant un réseau sans boucles. On notera I l'ensemble des unités d'entrée, et O l'ensemble des unités de sortie. On notera également $\mathrm{Amont}(k)$ l'ensemble des unités dont les sorties servent d'entrées à l'unité k, et $\mathrm{Aval}(k)$ l'ensemble des unités qui utilisent comme entrée la sortie de k.

Par définition, on a:

$$\forall\, i \in O,\ \mathrm{Aval}(i) = \emptyset \quad \text{et} \quad \forall\, i \in I,\ \mathrm{Amont}(i) = \emptyset$$

On notera W_{ij} le poids de la connexion liant l'unité i à l'unité j.

iii Calcul des dérivées

Soit $(s_k)_{k\in I}$, un exemple et $(s^*_k)_{k\in O}$ les sorties désirées associées. On s'intéresse à réduire l'écart quadratique $C_{\mathbf{x}}$ entre les sorties réelles et les sorties désirées.

$$C_{\mathbf{x}} = \sum_{k\in O} (s^*_k - s_k)^2 \qquad (2.25)$$

avec

$$a_i = \sum_{k \in Amont(i)} W_{ki} s_k \quad \text{et} \quad s_i = f(a_i) \tag{2.26}$$

On calcule donc

$$-\frac{\partial C_{\mathbf{x}}}{\partial W_{ij}} = -\frac{\partial C_{\mathbf{x}}}{\partial a_j}\frac{\partial a_j}{\partial W_{ij}} = -\frac{\partial C_{\mathbf{x}}}{\partial a_j} a_i = b_j \, . \, a_i \tag{2.27}$$

en posant $b_j = -\frac{\partial C_{\mathbf{x}}}{\partial a_j}$.

Il reste donc à évaluer b_j.

$$\text{si } j \notin O,\ b_j = -f'(a_j)\frac{\partial C_{\mathbf{x}}}{\partial s_j} = f'(a_j) \sum_{k \in Aval(j)} -\frac{\partial C_{\mathbf{x}}}{\partial a_k}\frac{\partial a_k}{\partial s_j} = f'(a_j) \sum_{k \in Aval(j)} b_k \, w_{jk}$$

$$\text{si } j \in O,\ b_j = -\frac{\partial C_{\mathbf{x}}}{\partial a_j} = .- f'(a_j)\frac{\partial C_{\mathbf{x}}}{\partial s_j} = f'(a_j)\,(s^*_j - s_j) \tag{2.28}$$

En rassemblant (2.27) et (2.28), on obtient l'expression de la mise à jour des poids dans l'algorithme de rétro-propagation du gradient:

$$\Delta W_{ij} = \varepsilon \, b_j \, . \, a_i$$

$$\text{si } j \in O,\ b_j = f'(a_j)\,(s^*_j - s_j)$$

$$\text{si } j \notin O,\ b_j = f'(a_j) \sum_{k \in Aval(j)} b_k \, w_{jk} \tag{2.29}$$

2.3.4.2 Remarques d'ordre général, illustrées sur le cas du perceptron multi-couches.

i Descente de gradient stochastique.

Dans tout ce qui précède, il y a une ambiguïté sur la notion de descente de gradient. A chaque itération, on fait un pas dans le sens du gradient relatif à l'erreur $C_{\mathbf{x}}$. Mais cette erreur $C_{\mathbf{x}}$ dépend elle même de l'exemple choisi.

En fait, ces algorithmes ne sont pas strictement des minimisations par descente de gradient. On cherche en effet à minimiser une erreur globale C,

$$C = < C_{\mathbf{x}} > \tag{2.30}$$

qui est la moyenne sur tous les exemples de l'erreur $C_{\mathbf{x}}$ pour chaque exemple. On le fait en tirant au hasard un exemple, c'est à dire un terme de la moyenne C, que l'on *réduit* en modifiant les poids

d'une petite quantité opposée au gradient. La section 3.3 est consacrée à l'étude de la convergence d'une telle procédure, appelée *gradient stochastique*.

Pour éviter cette ambiguïté, les premières rétro-propagations du gradient calculaient le gradient total, c'est à dire:

$$\frac{\partial C}{\partial W_{ij}} = < \frac{\partial C_{\mathbf{x}}}{\partial W_{ij}} > \qquad (2.31)$$

et effectuaient une vraie minimisation par descente de gradient. Ce procédé présente de multiples inconvénients.

Le plus apparent, (mais aussi le moins gênant, en fait), est qu'il faut passer tous les exemples avant de pouvoir effectuer une mise à jour des poids. On peut penser cependant que ces mises à jour des poids sont plus efficaces que celles du gradient stochastique. Sur des cas concrets, la convergence s'avère plusieurs dizaines de fois plus longue que celle du gradient stochastique!

Un autre inconvénient, majeur cette fois ci, est que l'algorithme de gradient total n'a aucun moyen d'échapper à un minimum local. L'algorithme stochastique présente un aspect aléatoire qui le lui permet sous certaines conditions.

Le gradient stochastique a été utilisé dès 1960 par Widrow, pour l'adaline. Le Cun [1] fut l'un des premiers qui l'appliqua systématiquement pour la rétro-propagation. L'ambiguïté du vocabulaire maintient cependant une certaine confusion: la descente stochastique de gradient ne semble pas concerner plus d'un tiers des publications au sujet de la rétro-propagation.

ii Poids partagés.

Comme on le verra au chapitre 7, on utilise souvent des *réseaux à poids partagés*. Il s'agit de réseaux dont certaines connexions sont contraintes à posséder un même coefficient w.

Cela implique une modification de l'algorithme d'apprentissage, les poids partagés devant bien sûr être mis à jour de façon identique. Cela est facile dans le cas de la rétro-propagation du gradient.

Si on considère le vecteur (W_k) des poids du réseau. On notera $\sigma(i,j)$ l'indice du poids associé à la connexion reliant l'unité i à l'unité j.

---

1 **Le Cun Y.:** *Modèles Connexionnistes de l'Apprentissage*- Thèse de doctorat de l'Université de Paris 6, (1987)

L'erreur E_x en fonction des poids partagés est alors la composition d'une première fonction Φ, qui au vecteur de poids (W_k) associe la matrice des coefficients $(W_{\sigma(i,j)})$ associés à chaque connexion (i,j), et de la fonction d'erreur usuelle C_x lorsque les poids ne sont pas partagés.

$$(W_k) \overset{\Phi}{\rightarrow} (W_{\sigma(i,j)}) \overset{C_x(W_{\sigma(i,j)})}{\rightarrow} \xi_x(W_k) = C_x \circ \Phi(W_k) \qquad (2.32)$$

Il suffit alors d'applique la règle de dérivation des fonctions composées pour trouver:

$$\frac{\partial \xi_x}{\partial W_k} = \sum_{\sigma(i,j)=k} \frac{\partial C_x}{\partial W_{\sigma(i,j)}} \qquad (2.33)$$

La dérivée par rapport à un poids partagé est donc la somme des dérivées des poids non partagés.

iii Liens avec les techniques d'analyse des données.

Quelques auteurs ont établi des liens entre algorithmes connexionnistes et méthode d'analyse des données, aussi bien dans le cas non linéaire [1] que linéaire [2] [3]. On peut en résumer les résultats ainsi:

- Certains réseaux utilisés comme système de compression de données effectuent en fait une analyse en composantes principales (Bourlard&Kamp).

- Certains réseaux, linéaires, utilisés comme systèmes de classification supervisés, effectuent en fait une analyse discriminante. (Gallinari).

---

1 **Bourlard H., Kamp Y.:** *Auto-Association by Multilayer Perceptrons and Singular Value Decomposition* - Biological cybernetics, n°59, pp 291-294, (1988) (preprint september 1987)

2 **Baldi P., Hornik K.:** *Neural networks and principal component analysis: learning from example without local minima.* - Neural Networks, Vol 2, pp 53-58, (1989)

3 **Gallinari P., Thiria S., Fogelman Soulié F.:** *Multilayer Perceptrons and Data Analysis* - Procs of IEEE 2nd ICNN, San Diego, vol I, pp 391-401 (1988)

2.3.5 Learning Vector Quantization (LVQ).

L'algorithme LVQ [1], qui est parfois considéré comme une variante des "cartes topologiques" du même auteur, est un algorithme de classification supervisée dont l'efficacité et la simplicité sont remarquables [2].

Chaque classe est caractérisée par un ensemble fixé d'unités calculant des distances. Leurs poids $\mathbf{W}_i$, constituent des vecteurs de référence de même dimension que les entrées. Lorsque l'on désire classifier un vecteur $\mathbf{X}$, on sélectionne le vecteur de référence le plus proche $\mathbf{W}^*$, et on regarde la classe $\Phi(\mathbf{W}^*)$ qui lui est associée.

Si l'on construit le diagramme de Voronoï associé aux divers vecteurs de référence, les frontières de séparation entre classes sont constituées des portions des frontières du diagramme de Voronoï séparant deux vecteurs de références associés à deux classes différentes (fig 2.12).

Ce principe permet donc de réaliser des frontières non linéaires, et même non convexes, s'il y a plusieurs références par classe.

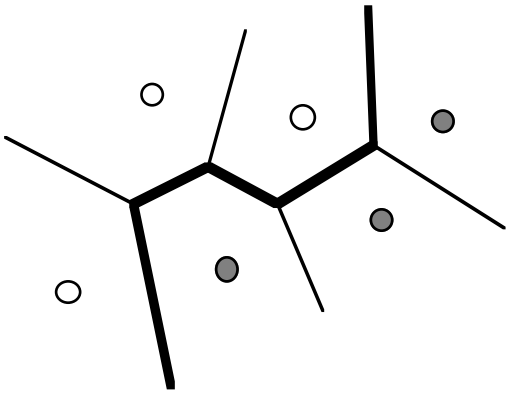


Fig 2.12 - Deux classes, caractérisées par six vecteurs de référence. Cette figure montre le diagramme de Voronoï, et en gras la frontière de séparation entre classes.

Connaissant la classe $\Phi_{\mathbf{X}k}$ associée à des exemples $\mathbf{X}_k$, un algorithme d'apprentissage doit ajuster au mieux les vecteurs de référence. Il en existe en fait deux variantes:

- L'algorithme *LVQ* consiste à tirer un exemple $\mathbf{X}$ au hasard, à déterminer le vecteur de référence le plus proche. Si ce vecteur est associé à la bonne classe, il sera rapproché de $\mathbf{X}$, sinon il en sera éloigné.

---

1 **Kohonen T., Barna G., Chrisley R.:** *Statistical Pattern Recognition with neural networks: Benchmarking Studies* - Second International Conference on Neural Networks, San Diego, IEEE proc. of ICNN, vol I, pp 61-68, (1988)

2 **McDermott E., Katagiri S.:** *Shift-invariant, Multi-category Phoneme Recognition using Kohonen's LVQ2* - Procedings of ICASSP89, S3.1, (1989)

$$\text{Si } \Phi_{\mathbf{x}} = \Phi(\mathbf{w}^*), \text{ alors } \Delta\mathbf{w}^* = \varepsilon\,(\mathbf{x} - \mathbf{w}^*)$$
$$\text{Si } \Phi_{\mathbf{x}} \neq \Phi(\mathbf{w}^*), \text{ alors } \Delta\mathbf{w}^* = -\,\varepsilon'\,(\mathbf{x} - \mathbf{w}^*) \qquad (2.34)$$

- L'algorithme *LVQ2* consiste de même à tirer un exemple $\mathbf{x}$ au hasard, à déterminer le vecteur de référence $\mathbf{w}^*$ le plus proche. Si la classe associée à ce vecteur est différente de la classe de $\mathbf{x}$, on ne repousse $\mathbf{w}^*$ que si un autre vecteur associé à la bonne classe se trouve dans le voisinage:

  On considère pour cela le second vecteur de référence le plus proche $\mathbf{w}^{**}$. Si celui ci est associé à la bonne classe, et si $\mathbf{x}$ se trouve assez près de la médiatrice de $\mathbf{w}^*$ et $\mathbf{w}^{**}$, on rapproche $\mathbf{w}^{**}$ et on repousse $\mathbf{w}^*$. Dans tous les autres cas, on ne fait rien.

$$\text{Si } \Phi_{\mathbf{x}} \neq \Phi(\mathbf{w}^*) \text{ et } \Phi_{\mathbf{x}} = \Phi(\mathbf{w}^{**}) \text{ et } (\mathbf{x}-\mathbf{w}^{**})^2 < (1+\delta)(\mathbf{x}-\mathbf{w}^*)^2 \text{ alors}$$
$$\Delta\mathbf{w}^* = -\,\varepsilon\,(\mathbf{x} - \mathbf{w}^*) \text{ et } \Delta\mathbf{w}^{**} = \varepsilon'\,(\mathbf{x} - \mathbf{w}^{**}) \qquad (2.35)$$

Ces deux algorithmes sont très simple, et permettent d'obtenir de bons résultats rapidement, à condition que les vecteurs de référence aient été initialisés convenablement.

Une procédure classique d'initialisation consiste à utiliser un algorithme K-means (cf §3.2.2) sur tous les exemples pour placer les références, puis à leur associer la classe dominante. Une autre procédure consiste à utiliser K-means sur tous les exemples de chaque classe, de façon à placer les références associées à cette classe.

2.4 La problématique de la généralisation

Les algorithmes ci-dessus apprennent à partir d'un nombre fini exemples, l'ensemble d'apprentissage.

On dit qu'ils généralisent s'ils sont capables de donner la bonne réponse si on présente une entrée qui n'appartient pas à l'ensemble d'apprentissage. On admet alors implicitement que le réseau a appris un concept, dont on possédait quelques exemples.

i Illustration

On ne peut évidemment pas généraliser ci ce concept est quelconque: Prenons le cas d'une fonction booléenne à trois entrées, apprise sur quatre exemples seulement.

x_1	x_2	x_3	s
0	0	0	**?**
0	0	1	**?**
0	1	0	**1**
0	1	1	**0**
1	0	0	**?**
1	0	1	**1**
1	1	0	**1**

1	1	1	**?**

Fig 2.13- Quatre exemples pour un problème booléen à trois entrées.

Généraliser, ce serait compléter la dernière colonne, en remplaçant les points d'interrogation par les bonnes valeurs. Seules les régularités que présentent les exemples peuvent donner des indices pour compléter ce tableau.

Un réseau connexionniste modélise cette fonction booléenne à l'aide d'une classe très limitée de fonctions, déterminées à quelques paramètres près, les poids.

Cette classe doit être assez vaste pour bien modéliser un ensemble d'exemples, tirés au hasard. Elle doit être assez restreinte pour forcer l'algorithme d'apprentissage à extraire des caractéristiques communes aux exemples, en lui interdisant de les "apprendre par cœur".

Ce compromis est apparu très clairement aux expérimentateurs. La performance sur un ensemble de test était liée au nombre de poids dans le réseau et au nombre d'exemples utilisés pour l'apprentissage.

ii Généralisation et interpolation.

On peut également comparer ce comportement à l'interpolation d'une fonction. On suppose qu'il existe une fonction inconnue f, le concept, dont on ne connait que quelques points, les exemples. Cela suffit il pour retrouver la fonction, et donc pouvoir déterminer sa valeur pour de nouvelles entrées?

La réponse est en général négative (fig 2.14). Il y a une infinité de façons de passer par un nombre fini de points. Cela n'est plus vrai si l'on recherche une fonction d'interpolation dans une classe assez restreinte, paramétrée par des poids.

Il faut alors disposer d'un ensemble d'exemples assez représentatif pour pouvoir définir, au sein de notre classe restreinte, la fonction la plus proche de la fonction inconnue f.

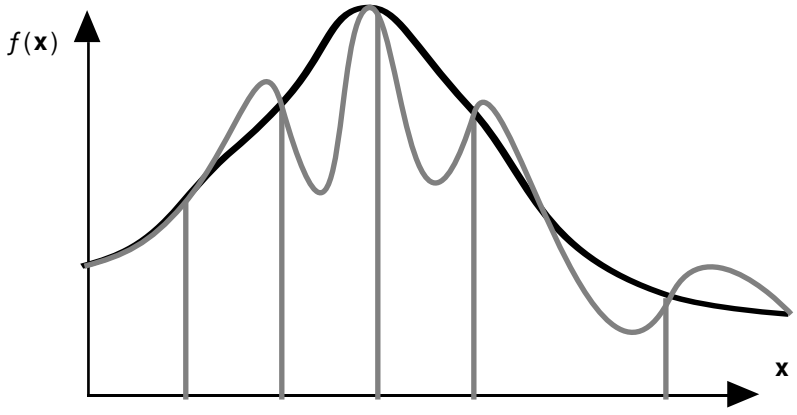


Fig 2.14 - Deux façons d'interpoler un ensemble de points par une courbe.

Ce schéma simple est compliqué parce qu'on apprend souvent à l'aide de données bruitées, et parce que ce que l'on apprend n'est pas une fonction: A chaque vecteur d'entrée n'est pas associée une unique réponse.

La nécessité d'une modélisation probabiliste constitue une différence qualitative entre apprentissage et interpolation; les ordres de grandeur des nombres d'exemples et de paramètres constituent de plus une différence quantitative.

2.5 Une problématique unifiée

Nous avons souligné, dans la section 2.1, que les algorithmes connexionnistes étaient une réponse algorithmique et mathématique à une problématique relevant des sciences humaines. Cela possède l'aspect très positif d'une étude pluridisciplinaire.

Les outils mathématiques, bien adaptés à l'étude des algorithmes, ne sont malheureusement pas adéquats pour aborder les concepts des sciences humaines, et donc pour appréhender l'unité profonde des algorithmes connexionistes. On adopte donc, pour chaque algorithme, une approche mathématique ad hoc.

La présentation des algorithmes dans ce chapitre tend au contraire à montrer qu'il est possible de trouver des problématiques mathématiques. Elles ne sont pas équivalentes à la problématique connexionniste, mais permettent d'étudier de façon unifiée beaucoup d'algorithmes connexionnistes, et quelques autres…

Nous en avons retenu deux.

- Lorsqu'on parle d'apprentissage, on a souvent en tête la recherche d'un optimum. Celui ci s'exprime par un critère global; minimiser, par exemple, le nombre d'erreurs.

  Or, on est souvent contraint d'apprendre exemple après exemple, c'est à dire d'optimiser à chaque étape une partie seulement de ce critère global. L'étude de ces procédures, curieusement efficaces, constitue la *problématique de l'apprentissage stochastique*, qui est l'objet du chapitre 3.

- Un autre aspect, souligné ci dessus, est la *problématique de la généralisation*. Est il possible d'apprendre avec un nombre fini d'exemples? Ce sera l'objet du chapitre 4.

Ces deux problématiques sont étroitement liées. Ce que l'on peut apprendre n'est pas très différent de ce que l'on sait généraliser…

3 Apprentissage stochastique.

3.1 Apprentissage et optimisation stochastique.

L'apprentissage se présente souvent comme un problème d'optimisation. On introduit ici une forme simple et générale d'algorithme d'optimisation stochastique: la descente de gradient stochastique.

Tout système de traitement de l'information est conçu pour remplir un certain but. Si l'on sait caractériser ce but analytiquement, il ne reste qu'à résoudre un système d'équations. On peut alors entièrement concevoir le système à l'avance.

Fréquemment, cependant, on ne sait pas caractériser ce but, mais on saurait mesurer la qualité avec laquelle il est atteint sur un système donné.

Par exemple, on ne sait pas exprimer analytiquement les mots contenus dans un signal de parole. En revanche, si l'on dispose d'un système de reconnaissance de la parole, il suffit d'un jeu d'essai pour en mesurer la qualité.

On peut alors envisager d'améliorer un système, en optimisant cette mesure de la qualité. C'est dans ce sens que l'on parle ici d'apprentissage et d'adaptation.

3.1.1 Optimisation stochastique.

3.1.1.1 Apprentissage et adaptation.

Il y a une certaine différence entre les notions d'apprentissage et d'adaptation.

- Dans le cas de *l'adaptation*, on améliore un système au cours de son utilisation. C'est le cas, par exemple, des algorithmes de traitement adaptatif du signal, ou d'un système de contrôle de bras de robot.

  L'intérêt de la démarche repose dans la capacité du système d'apprentissage à s'adapter à des variations lentes de l'environnement, comme l'évolution de la nature du bruit sur une ligne téléphonique, ou encore l'apparition de jeux dans les articulations d'un bras de robot.

- Dans le cas de *l'apprentissage*, on dispose par avance d'un certain nombre d'exemples, et on désire calculer un système figé, capable de remplir par la suite une tâche précise.

On peut imaginer également des systèmes mixtes, initialisés par apprentissage, et capables par la suite d'adaptation.

3.1.1.2 Nécessité de la procédure stochastique.

Le critère de qualité que l'on optimise est usuellement un critère global, comme minimiser une probabilité d'erreur, que l'on peut évaluer en observant comment le système réagit sur un ensemble représentatif des conditions extérieures possibles.

Or on ne souhaite pas, dans le cas d'un *algorithme d'adaptation,* attendre d'avoir collecté des statistiques complètes sur le fonctionnement du système depuis la dernière adaptation. On désire que l'adaptation ne dépende que du comportement présent du système, bon ou mauvais, qui est une piètre mesure du critère de qualité.

Les améliorations successives du systèmes possèdent donc une composante aléatoire, dépendant des conditions extérieures à un instant donné. On espère alors qu'en moyenne, on va améliorer le critère global, à l'aide de cet *algorithme stochastique*. En contrepartie, on a simplifié l'algorithme, et probablement accéléré l'adaptation.

Dans le cas d'un *algorithme d'apprentissage*, on dispose de tous les exemples, et on peut envisager de mesurer le critère de qualité. Cela ne dépend que de la puissance du calculateur, et non de phénomènes physiques extérieurs. Cependant, simuler un algorithme stochastique, en tirant au hasard les exemples, s'avère souvent plus simple et plus rapide!

Si on dispose en effet d'assez d'exemples pour apprendre quelque chose, ceux ci sont souvent extrêmement redondants. Un algorithme stochastique ne nécessitera alors que le *passage d'une faible partie des exemples* pour faire *en moyenne* ce qui exige le *passage de tous les exemples* pour un algorithme non stochastique.

3.1.2 Forme générale de l'optimisation stochastique.

3.1.2.1 Forme du critère global.

i Critère global.

On s'intéresse maintenant aux critères globaux de la forme [1] suivante :

$$C = E_{\mathbf{x}}(J(\mathbf{x},\mathbf{w})) \qquad (3.1)$$

où $\mathbf{x}$ est un vecteur représentant les conditions extérieures ou les exemples, et $\mathbf{w}$, un vecteur de paramètres qui doivent être adaptés.

On note $E_{\mathbf{x}}(\bullet)$ l'espérance mathématique associée à la variable aléatoire $\mathbf{x}$. Cette formule peut s'écrire sous forme intégrale en introduisant la densité $p(\mathbf{x})$ de probabilité associée à la variable aléatoire $\mathbf{x}$.

$$C = \int J(\mathbf{x},\mathbf{w})p(\mathbf{x})d\mathbf{x} \qquad (3.2)$$

La fonction $J(\mathbf{x},\mathbf{w})$ est le critère local, c'est à dire la mesure peu fiable du critère global sur un seul exemple. Notons également que l'on connait analytiquement $J(\mathbf{x},\mathbf{w})$, mais que la densité $p(\mathbf{x})$ est habituellement inconnue.

ii Illustration.

Prenons le cas d'un perceptron multi-couches comportant une seule sortie, chargée de classer les entrées en deux classes. Les vecteurs $\mathbf{x}$ auront donc la forme $(\mathbf{X}, C)$ où $\mathbf{X}$ est un vecteur d'entrée et C un nombre -1 ou $+1$ représentant la classe.

Si on note $f(\mathbf{X},\mathbf{w})$ la sortie du perceptron multi-couches en fonction de l'entrée X et des poids $\mathbf{w}$, on a

---

1 **Tsypkin Ya.:** *Foundations of the Theory of Learning Systems* - Mathematics in science and engineering, vol 101, Academic Press, (1973)

$$C = E_{(\mathbf{X},C)}\left((C - f(\mathbf{X},\mathbf{w}))^2 \right) = \int (C - f(\mathbf{X},\mathbf{w}))^2 \, P(C|\mathbf{X}) p(\mathbf{X}) \, dC d\mathbf{X} \quad (3.3)$$

ce qui est bien de la forme (3.1).

On remarquera également que l'information "$\mathbf{X}$ appartient à la classe C" est exprimée uniquement par la probabilité $P(C|\mathbf{X})$ qui multipliée par $p(\mathbf{X})$ représente la densité des couples $(\mathbf{X}, C)$. Il est donc essentiel de considérer que $p(\mathbf{x})$ *est inconnue.*

Il suffit de consulter les dizaines d'algorithmes proposés dans [1] pour se persuader de la généralité de ce formalisme.

3.1.2.2 Descente stochastique de gradient.

On se propose maintenant d'optimiser (3.2)

$$\inf_{\mathbf{w}} \int J(\mathbf{x},\mathbf{w}) p(\mathbf{x}) d\mathbf{x} \quad (3.4)$$

L'algorithme classique de descente de gradient pour (3.4), que l'on nomme ici *gradient total* ou *gradient déterministe*, consiste à itérer

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \varepsilon_t \nabla C(\mathbf{w}_{t-1}) = \mathbf{w}_{t-1} - \varepsilon_t \nabla \left(\int J(\mathbf{x},\mathbf{w}_{t-1}) p(\mathbf{x}) d\mathbf{x} \right) \quad (3.5)$$

en notant le gradient par rapport aux paramètres $\mathbf{w}$ à l'aide de l'opérateur nabla "∇". Dans cette équation, le *gain* ε_t peut être soit un réel positif, soit une matrice symétrique définie positive.

Cependant, nous avons déjà souligné que $p(\mathbf{x})$ est inconnue. On souhaite donc approcher (3.5) en estimant le gradient à l'aide d'un échantillon d'exemples $\{\mathbf{x}_k\}$ indépendants.

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \varepsilon_t \nabla C(\mathbf{w}_t) \approx \mathbf{w}_{t-1} - \frac{\varepsilon_t}{N} \sum_{k=1}^{N} \nabla J(\mathbf{x}_k,\mathbf{w}_{t-1}) \quad (3.5')$$

En effet, si $\nabla J(\mathbf{x},\bullet)$ est intégrable, la loi des grands nombres nous apprend que la moyenne empirique des gradients de J converge vers leur espérance mathématique. En outre, lorsque J est différentiable, et que son gradient est intégrable, l'espérance du gradient est égal au gradient de l'espérance.

$$\frac{1}{N} \sum_{k=1}^{N} \nabla J(\mathbf{x}_k,\mathbf{w}_{t-1}) \underset{N\to\infty}{\overset{p.s.}{\longrightarrow}} E_{\mathbf{x}}(\nabla J(\mathbf{x},\mathbf{w}_{t-1})) = \nabla E_{\mathbf{x}}(J(\mathbf{x},\mathbf{w}_{t-1})) = \nabla \int J(\mathbf{x},\mathbf{w}_{t-1}) p(\mathbf{x}) d\mathbf{x}$$

---

1 **Tsypkin Ya.:** *Adaptation and Learning in Automatic systems* - Mathematics in science and engineering, vol 73, Academic Press, (1971)

L'algorithme de descente de *gradient stochastique* est une simplification de (3.5'). Il consiste à tirer, à chaque étape, un exemple $\mathbf{x}_t$, et à appliquer

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \varepsilon_t \nabla J(\mathbf{x}_t, \mathbf{w}_{t-1}) \qquad (3.6)$$

L'algorithme de gradient stochastique présente un grand avantage: Il ne requiert ni de connaître $p(\mathbf{x})$, ni de l'évaluer sur un échantillon. C'est le comportement moyen de l'algorithme qui tient lieu d'évaluation de $p(\mathbf{x})$.

Cependant, rien n'assure a priori que la procédure (3.6) converge vers un minimum de C. Chaque pas de gradient, lorsque ε_t est assez petit, a pour effet de réduire la valeur de $J(\mathbf{x}_t,\mathbf{w})$ pour une situation $\mathbf{x}_t$ donnée. Cela ne réduit pas en général $J(\mathbf{x},\mathbf{w})$ pour toutes les situations $\mathbf{x}$, et donc ne réduit pas nécessairement l'erreur globale C.

On distinguera deux cas concernant le paramètre ε_t, dit *pas de gradient*:

- Le cas *à pas fixe*. Le pas de gradient possède une norme faible, mais reste constant. C'est le cas privilégié pour les algorithmes d'adaptation, qui doivent conserver une capacité à suivre de lentes évolutions des conditions extérieures.

- Le cas *à pas décroissant*. La norme du pas de gradient ε_t décroît au fur et à mesure de l'apprentissage, et tend vers 0. On dispose dans ce cas de théorèmes généraux de convergence. C'est le cas des algorithmes d'apprentissage, pour lesquels on désire obtenir la meilleure solution.

On considère souvent des pas décroissants satisfaisant les conditions

$$\sum_{t=0}^{+\infty} |\varepsilon_t| = +\infty , \qquad \sum_{t=0}^{+\infty} |\varepsilon_t|^2 < +\infty \qquad (3.7)$$

qui (cf. §3.3), constituent une hypothèse fréquente dans les théorèmes de convergence.

3.1.2.3 Problèmes de régularité.

Il est fréquent en pratique que $J(\mathbf{x},\bullet)$ soit *non différentiable* sur un *ensemble de mesure nulle*. Or, il n'est pas nécéssaire que ∇J soit strictement le gradient de J. Il suffit en effet dans les théorèmes de convergence (3.36) et (3.46), que ∇J vérifie la condition ci-dessous

$$\forall \mathbf{w}, \quad E_{\mathbf{x}}(\nabla J(\mathbf{x},\mathbf{w})) = \nabla E_{\mathbf{x}}(J(\mathbf{x},\mathbf{w})) = \nabla C \qquad (3.8)$$

Lorsque J possède une différentielle intégrable, les propriétés générales des intégrales de Lebèsgue assurent que (3.8) est vérifiée.

Si J est non différentiable sur un ensemble de mesure nulle, le théorème de la *convergence dominée* fournit une condition suffisante pour (3.8): Il suffit que les accroissements de J soient majorés au voisinage de tout point $(\mathbf{x},\mathbf{w})$ par une fonction intégrable $\Theta(\mathbf{x},\mathbf{w})$:

$$\forall \mathbf{x},\mathbf{w},\ \forall \mathbf{h} \in V(\mathbf{0}) \quad \frac{1}{|\mathbf{h}|} \left(J(\mathbf{x},\mathbf{w}+\mathbf{h}) - J(\mathbf{x},\mathbf{w}) \right) < \Theta(\mathbf{x},\mathbf{w}) \text{ } \textit{intégrable} \tag{3.9}$$

Cette condition suffisante permet de conclure dans la plupart des cas.

3.1.3 Hypothèses asymptotiques et généralisation.

L'étude d'un tel algorithme n'est pas séparée du problème de généralisation. On peut distinguer au moins trois façons d'étudier cette convergence:

- Dans *l'étude ordinaire* de la convergence, on suppose que l'on dispose d'un nombre K fini d'exemples $\mathbf{x}_k$. On applique alors l'algorithme (3.6) à des exemples tirés au hasard dans cet ensemble.

  On étudie en fait la convergence de l'algorithme, appliqué à la minimisation d'une fonctionnelle empirique $\tilde{C}$ définie sur les exemples $\mathbf{x}_k$, c'est à dire sur un probabilité $\tilde{p}(\mathbf{x})$ discrète.

  $$\tilde{C} = \int J(\mathbf{x},\mathbf{w})\ \tilde{p}(\mathbf{x}) d\mathbf{x} = \sum_{k=1}^{K} J(\mathbf{x}_k,\mathbf{w}) \tag{3.10}$$

  En fait, on désire seulement prouver que notre algorithme peut apprendre les K exemples dont on disposait.

- Dans *l'étude asymptotique*, qui est l'objet de ce chapitre, on suppose que l'on tire chaque fois un exemple issu d'une distribution de probabilité inconnue $p(\mathbf{x})$. On désire alors montrer que l'algorithme (3.6) converge vers un minimum de la fonctionnelle réelle C définie sur cette distribution sous-jacente inconnue (3.2).

- Dans *l'étude de la généralisation*, on suppose que l'on tire par avance un ensemble fini d'exemples, et que l'on applique à nouveau l'algorithme (3.6) à des exemples tirés au hasard dans cet ensemble. On souhaite alors montrer que la convergence de l'algorithme (3.6) vers un minimum de la fonctionnelle empirique $\tilde{C}$, s'accompagne de la convergence de C vers une valeur proche de son minimum.

Dans l'étude asymptotique, comme dans l'étude de la généralisation, on étudie la convergence de l'algorithme vers un système ayant un comportement optimal pour l'ensemble des nouvelles situations pouvant apparaître selon la distribution $p(\mathbf{x})$.

L'étude asymptotique permet d'appliquer élégamment des raisonnements statistiques, pour déterminer les propriétés de ces systèmes [1]. Malheureusement, les hypothèses asymptotiques sont éloignées de la réalité: on ne dispose ni d'un temps infini ni d'un nombre d'exemples pléthorique. L'étude de la généralisation consiste à tenir compte de ces limitations de ressources.

3.2 Exemples.

Un grand nombre d'algorithmes connexionnistes (cf chp. 2) ou statistiques entrent dans le cadre général de l'algorithme stochastique décrit dans la section précédente. On retrouve ainsi des algorithmes connus, comme l'adaline, le perceptron ordinaire ou multi-couches, ou l'algorithme "k-means".

3.2.1 Régressions et algorithmes connexionnistes.

Les algorithmes d'apprentissage de l'adaline et du perceptron multi-couches constituent deux illustrations typiques d'algorithmes du type (3.6).

Les vecteurs $\mathbf{x}$, représentant les interactions du réseau avec l'extérieur, y ont la forme $(\mathbf{X}, \mathbf{Y})$ où $\mathbf{X}$ est un vecteur d'entrées du réseau et $\mathbf{Y}$, le vecteur des sorties désirées associées à ces entrées.

Le réseau, adaline ou perceptron multi-couches, peut donc être représenté par une fonction $f(\mathbf{X},\mathbf{w})$, qui associe aux entrées un vecteur de sorties. On cherche alors à effectuer une *régression*, c'est à dire trouver la fonction $f(\mathbf{X},\mathbf{w}^*)$ de $\mathbf{X}$ qui approche le plus fidèlement $\mathbf{Y}$, selon une distance à définir.

i Adaline.

Dans le cas de l'adaline, f possède la forme

$$f(\mathbf{X},\mathbf{w}) = \mathbb{1}_{\Re_+}(\mathbf{w}^T \mathbf{X}) \qquad (3.11)$$

et on se propose de minimiser

---

1 **Le Cam L.:** *Asymptotic Methods in Statistical Decision Theory* - Springer Series in Statistics, Springer Verlag (1986)

$$C = \int (Y - \mathbf{w}^T\mathbf{X})^2 \, p(Y|\mathbf{X})p(\mathbf{X})dY\,d\mathbf{X} = \int (Y - \mathbf{w}^T\mathbf{X})^2 \, p(\mathbf{x})d\mathbf{x} \tag{3.12}$$

L'algorithme stochastique correspondant est donc la *règle du Delta*, comparable aux la formules (2.15) et (2.21).

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \varepsilon_t \nabla(Y - \mathbf{w}_{t-1}^T\mathbf{X})^2 = \mathbf{w}_{t-1} + 2\varepsilon_t(Y - \mathbf{w}_{t-1}^T\mathbf{X})\mathbf{X} \tag{3.13}$$

Remarque sur les notations: un exposant T représente la transposition. Un indice t représente l'indice temporel des itérations de l'algorithme. $\mathbf{w}_{t-1}^T$ est donc le transposé du vecteur de poids à l'itération précédente.

ii Cas du perceptron multi-couches.

Dans le cas du perceptron multi-couches, la fonction f dépend de l'architecture des connexions du réseau. Il s'agit alors d'une composée de sigmoïdes et de fonctions linéaires. La minimisation stochastique de

$$C = \int (Y - f(\mathbf{X},\mathbf{w}))^2 \, p(\mathbf{x})d\mathbf{x} \tag{3.14}$$

donne

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \varepsilon_t \nabla(Y - f(\mathbf{X},\mathbf{w}_{t-1}))^2 \tag{3.15}$$

d'où l'on extrait *l'algorithme de rétro-propagation stochastique* (cf §2.3.4).

iii Cas du Perceptron.

Le critère à optimiser dans le cas du perceptron (2.13) s'écrit

$$C = E_{\mathbf{x}}(J(\mathbf{x},\mathbf{w})) = \int -\left(Y - \mathbb{1}_{\Re_+}(\mathbf{w}^T\varphi(\mathbf{X}))\right) \mathbf{w}^T\varphi(\mathbf{X}) \; p(\mathbf{x})d\mathbf{x} \tag{3.16}$$

l'algorithme stochastique correspondant est alors

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \varepsilon_t \nabla\left((Y - \mathbb{1}_{\Re_+}(\mathbf{w}_{t-1}^T\varphi(\mathbf{X}))) \; \mathbf{w}_{t-1}^T\varphi(\mathbf{X})\right) \tag{3.17}$$

L'intégrande du coût C est continu mais non différentiable sur un ensemble de mesure nulle, l'hyperplan $\{\mathbf{w}^T\varphi(\mathbf{x})=0\}$. On peut cependant remarquer que

$$E_{\mathbf{x}}(\nabla J(\mathbf{x},\mathbf{w})) = \nabla E_{\mathbf{x}}(J(\mathbf{x},\mathbf{w}) = \nabla C$$

Les accroissements de J sont en effet bornés par $|2\varphi(\mathbf{X})|$, qui est intégrable si $\varphi(\mathbf{X})$ l'est. Le théorème de la convergence dominée permet de conclure (cf. §3.1.2.3).

On peut alors appliquer l'algorithme (3.17). On obtient alors la *règle du perceptron*:

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \varepsilon_t \, (\mathbf{Y} - \mathbb{1}_{\Re_+} (\mathbf{w}_{t-1}{}^{\mathsf{T}} \varphi(\mathbf{X}))) \varphi(\mathbf{X}) \qquad (3.18)$$

3.2.2 Minimisation de l'erreur de quantification.

Ce dernier exemple est destiné à montrer que des méthodes usuelles d'analyse de données sont aussi des techniques d'apprentissage stochastique.

Le but d'un algorithme de "clustering" est de partitionner un ensemble de points en nuages (ou clusters) bien isolés et compacts. En terme de distribution de probabilité, il s'agit de déterminer les pics de la distribution de probabilité (fig 3.1).

Représentons chaque nuage par son centroïde. La partition en nuages est définie par le diagramme de Voronoï associé à l'ensemble des centroïdes. Si on se fixe un nombre K de nuages, le but de l'algorithme est de déterminer ces K centroïdes $\mathbf{w}_k$.

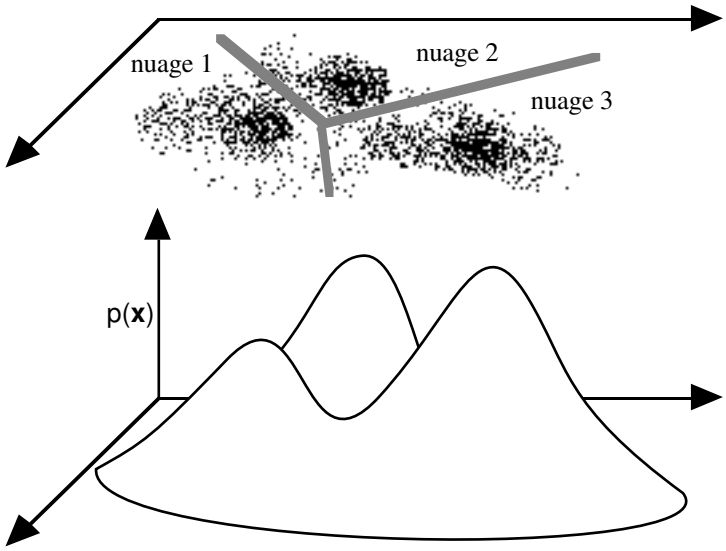


Fig 3.1- Une distribution de probabilité, un échantillon de points associés, et un découpage en nuages (clusters).

Pour ce faire, on peut minimiser l'erreur de quantification: Si l'on remplace chaque point par le centroïde de son nuage, on commet une certaine erreur, qui est la norme de l'écart entre ce point et le centroïde le plus proche. L'erreur globale s'écrit donc

$$C = \int \min_{k=1}^{K} (\mathbf{x} - \mathbf{w}_k)^2 \, p(\mathbf{x}) d\mathbf{x} \qquad (3.19)$$

On peut alors écrire l'algorithme stochastique

$$(\mathbf{w}_k)_t = (\mathbf{w}_k)_{t-1} - \varepsilon_t \nabla_{\mathbf{w}_k} \left(\min_{k=1}^{K} (\mathbf{x}-\mathbf{w}_k)^2 \right) \qquad (3.20)$$

Comme dans le cas du perceptron se pose un problème de régularité. La fonction Min est continue, mais non dérivable sur les frontières du diagramme de Voronoï. Sa dérivée possède donc une discontinuité sur ces points. Ici encore, la condition (3.9) est vérifiée, et on peut négliger ces points (cf. §3.1.2.3).

Si on note k^* l'indice du centroïde le plus proche du point $\mathbf{x}$, l'algorithme s'écrit alors

$$\begin{aligned} & (\mathbf{w}_{k^*})_t = (\mathbf{w}_{k^*})_{t-1} + \varepsilon_t (\mathbf{x}-(\mathbf{w}_{k^*})_{t-1}) \\ \forall\, k\neq k^*, \quad & (\mathbf{w}_k)_t = (\mathbf{w}_k)_{t-1} \end{aligned} \qquad (3.21)$$

Ceci est comparable à l'algorithme classique k-means de Mac Queen,(cf. [1]), qui s'écrit

$$\begin{aligned} & (\mathbf{w}_{k^*})_t = \tfrac{t-1}{t}(\mathbf{w}_{k^*})_{t-1} + \tfrac{1}{t}\mathbf{x} = (\mathbf{w}_{k^*})_{t-1} + \tfrac{1}{t}(\mathbf{x}-(\mathbf{w}_{k^*})_{t-1}) \\ \forall\, k\neq k^*, \quad & (\mathbf{w}_k)_t = (\mathbf{w}_k)_{t-1} \end{aligned} \qquad (3.22)$$

On peut également vérifier que chaque itération de cet algorithme rapproche, en moyenne, les centroïdes de la moyenne des points du nuage correspondant:

En calculant la mise à jour moyenne des centroïdes dans la formule (3.21), on obtient aisément:

$$\begin{aligned} E_{\mathbf{x}}\left((\mathbf{w}_{k^*})_t - (\mathbf{w}_{k^*})_{t-1}\right) &= \varepsilon_t \int_{\Omega_k} \left(\mathbf{x}-(\mathbf{w}_{k^*})_{t-1}\right) p(\mathbf{x})d\mathbf{x} \;= \\ &= \varepsilon_t P(\Omega_k) \left(E_{\mathbf{x}}(\mathbf{x} \mid \mathbf{x}\in\Omega_k) - (\mathbf{w}_{k^*})_{t-1} \right) \end{aligned} \qquad (3.23)$$

en notant Ω_k la partie de l'espace associée dans le diagramme de Voronoï au centroïde d'indice k.

3.3 Etude de la convergence.

On étudie ici la convergence de la forme générale d'algorithme stochastique (3.6) introduite en 3.1.

Dans le cas convexe, la méthode des fonctions de Lyapunov s'applique aussi bien à une approximation continue de la descente de gradient, aux algorithmes discrets de descente de gradient déterministe, et aux algorithmes stochastiques de descente

1 **Duda R.O., Hart P.E.:** *Pattern classification and Scene analysis*, Wiley (1973)

de gradient. Il faut cependant introduire à chaque étape des outils différents: intégrales, séries convergentes, et quasi-martingales.

On peut de même obtenir des résultats de convergence dans le cas général, et développer une argumentation physique montrant les liens de l'optimisation stochastique avec le recuit simulé.

Cette discussion s'applique à tous les algorithmes dérivés de l'équation (3.6). Elle constitue donc une preuve simultanée de la convergence de tous les algorithmes connexionnistes ou statistiques dont il est question dans cet ouvrage.

Cette section fait parfois appel à des développements mathématiques complexes. En effet, l'étude de la convergence des algorithmes de descente de gradient présente des difficultés croissantes.

- L'approche la plus simple consiste à étudier *l'algorithme en temps continu*. Il s'agit alors de déterminer les attracteurs stables de l'équation différentielle

$$\frac{d\mathbf{w}}{dt} = - \nabla C(\mathbf{w}) \qquad (3.24)$$

- Lorsque l'on étudie un *algorithme discret de descente de gradient* de type (3.5), il faut prendre certaines précautions concernant le pas de gradient, i.e. le pas de discrétisation temporelle. On utilise alors souvent un pas de gradient décroissant. Il ne doit cependant pas décroître trop vite…

- Enfin, l'étude des *descentes de gradient stochastique* de type (3.6) combine cette difficulté avec une nouvelle: l'apparition de comportements aléatoires dans l'algorithme. Cependant, il existe une méthode puissante, introduite dans [1]. Cette méthode est fondée sur l'utilisation de quasi-martingales.

Commençons, dans un premier temps, par étudier ces trois aspects sur un cas simple.

3.3.1 Un cas simple, incluant le cas convexe.

On considère usuellement le cas convexe comme le cas le plus simple d'optimisation. La fonction globale C, convexe, dérivable, possède alors un unique minimum $\mathbf{w}^*$. En fait, dans les calculs, on utilise une hypothèse plus faible que la convexité, qui s'exprime:

$$\forall \varepsilon>0, \quad \liminf_{|\mathbf{w}-\mathbf{w}^*|>\varepsilon} (\mathbf{w}-\mathbf{w}^*).\nabla C(\mathbf{w}) > 0 \qquad (3.25)$$

---

1 **Gladyshev E.G.:** *On stochastic approximation*. Theory of Probability and its Applications, vol 10, pp 275-278, (1965)

La positivité du produit scalaire $(\mathbf{w}-\mathbf{w}^*).\nabla C(\mathbf{w})$ signifie que l'opposé du gradient est toujours dirigé vers le minimum. La limite inférieure sert à assurer que ce produit n'est nul qu'en $\mathbf{w}^*$, et que la fonction C ne présente pas de pathologies au voisinage de ce minimum. Prouver la convergence du gradient ∇C vers 0 suffit alors à prouver la convergence de $\mathbf{w}$ vers $\mathbf{w}^*$.

Cette condition (3.25) est vérifiée dans le cas de la plupart des dispositifs linéaires, et notamment de l'adaline.

3.3.1.1 Etude en temps continu.

On recherche ici les attracteurs stables de l'équation différentielle (3.23). Pour cela, on définit une *fonction de Lyapunov*.

$$h(t) = (\mathbf{w}-\mathbf{w}^*)^2 \tag{3.26}$$

et on peut calculer sa dérivée

$$\frac{dh}{dt}(t) = 2(\mathbf{w}-\mathbf{w}^*)\frac{d\mathbf{w}}{dt} = -2\,(\mathbf{w}-\mathbf{w}^*).\nabla C(\mathbf{w}) < 0 \quad si\ (\mathbf{w}-\mathbf{w}^*)^2>0 \tag{3.27}$$

La fonction h est donc positive, décroissante, et donc converge vers une limite h_∞ finie. De plus, si $\limsup h'(t) = \alpha<0$, alors, pour t assez grand, $h'(t)<\alpha/2<0$ et donc $h(t)$ tendrait vers $-\infty$. Comme $h(t)$ est positif par définition, on a

$$\limsup_{t\to\infty} h'(t) = \liminf_{t\to\infty} (\mathbf{w}-\mathbf{w}^*).\nabla C(\mathbf{w}) = 0$$

Et donc, d'après la condition (3.25),

$$\liminf_{t\to\infty} (\mathbf{w}_t-\mathbf{w}^*)^2 = \liminf_{t\to\infty} h(t) = h_\infty = 0$$

La fonction de Lyapunov $h(t)$ tend donc nécessairement vers 0. Ce qui montre que $\mathbf{w}^*$ est l'unique attracteur stable de (3.23).

3.3.1.2 Etude en temps discret et gradient total.

On étudie maintenant l'algorithme en temps discret de descente du gradient total (3.5). On utilise un analogue discret de la méthode de Lyapunov. On pose

$$h_t = (\mathbf{w}-\mathbf{w}^*)^2 \tag{3.28}$$

et on calcule les accroissements de cette suite, à l'aide de (3.5).

$$h_{t+1} - h_t = - 2\,\varepsilon_t\,(\mathbf{w}_t-\mathbf{w}^*)\,\nabla C(\mathbf{w}_t) + \varepsilon_t^2\,\nabla C(\mathbf{w}_t)^2 \le \varepsilon_t^2\,\nabla C(\mathbf{w}_t)^2 \qquad (3.29)$$

Supposons maintenant que $\nabla C(\mathbf{w}_t)^2$ est borné. Si on utilise un pas de gradient vérifiant la condition (3.7),

$$\sum_t \varepsilon_t = +\infty, \qquad \sum_t \varepsilon_t^2 < +\infty$$

les accroissements de la suite h_t sont alors majorés par le terme d'une série convergente. En particulier, les accroissements positifs de la suite h_t forment une série convergente. Comme h_t est positive, la série des accroissements négatifs, dont la somme est décroissante et minorée, converge également. La suite h_t est donc convergente.

On peut arriver au même résultat avec une hypothèse plus faible sur $\nabla C(\mathbf{w}_t)^2$:

$$\nabla C(\mathbf{w})^2 < A^2+B^2(\mathbf{w}-\mathbf{w}^*)^2 \qquad (3.30)$$

On a alors

$$h_{t+1} - (1+\varepsilon_t^2B^2)h_t \;\le\; \varepsilon_t^2A^2$$

Posons

$$\Pi_t = \prod_{k=1}^{t-1}(1+\varepsilon_k^2B^2)^{-1} \underset{t\to\infty}{\longrightarrow} \Pi_\infty, \quad \textit{et définissons} \quad h'_t = h_t\Pi_t$$

L'existence de la limite Π_∞ est évidente si on considère le logarithme de Π_t. En remplaçant dans (3.29), on obtient

$$h'_{t+1} - h'_t \le \varepsilon_t^2\Pi_{t+1}\,A^2 \qquad (3.31)$$

Les accroissements de h'_t sont donc majorés par une série convergente. En particulier, ses accroissements positifs le sont, et forment donc une série convergente. Comme h'_t est positive, la série des accroissements négatifs, dont la somme est décroissante et minorée, converge également. La suite h'_t est donc convergente.

La suite h_t converge donc également. Sa limite ne peut être que zéro. En effet, de (3.29) on déduit que la série de terme $\varepsilon_t(\mathbf{w}_t-\mathbf{w}^*)\nabla C(\mathbf{w}_t)$ est elle aussi convergente. Comme la série de terme ε_t est divergente, et comme $(\mathbf{w}_t-\mathbf{w}^*)\nabla C(\mathbf{w})$ est positif par hypothèse, on a

$$\liminf_{t\to\infty} (\mathbf{w}_t-\mathbf{w}^*)\nabla C(\mathbf{w}_t) = 0$$

et donc, d'après la condition (3.25)

$$\lim_{t\to\infty} \inf (\mathbf{w}_t - \mathbf{w}^*)^2 = \lim_{t\to\infty} \inf h_t = 0$$

On a donc montré que h_t tend vers 0, et donc que l'algorithme converge vers le minimum de C. Cependant, deux hypothèses supplémentaires ont du être faites:

- L'une (3.7) sur les pas de gradients est propre à la nature discrète de l'algorithme.
- L'autre (3.30) représente une concession sur la nature de la fonction C: Ses gradients doivent être bornés, ou bien ne pas croître plus que linéairement avec la norme de $\mathbf{w}-\mathbf{w}^*$.

La situation se dégrade encore lorsque l'on considère les algorithmes stochastiques. Le schéma de la démonstration reste identique. Cependant, on ne peut majorer que les espérances de notre série h_t. On utilise alors des résultats généraux sur les quasi-martingales, qui permettent de conclure que h_t converge presque sûrement, c'est à dire que

$$P\left\{ \lim_{t\to\infty} h_t \textit{ existe} \right\} = 1$$

3.3.1.3 Quasi-martingales.

L'objet de cette sous-section est d'introduire les résultats fondamentaux sur les quasi-martingales [1], [2] qui s'avèrent utiles pour l'étude de la convergence des algorithmes stochastiques.

i Définition.

Soit $(h_t)_{t\geq 0}$ un processus aléatoire dans $\Re^p$.

On notera F_t la famille des événements observables à l'instant t, i.e. les événements observables sur $h_0, h_1, \ldots h_t$.

On notera également $E(X \mid F_t)$ l'espérance de la variable X sachant tout ce qui s'est produit jusqu'à l'instant t, i.e. $E(X \mid h_0, h_1, \ldots h_t)$.

---

1 **Fisk D.L.:** *Quasi-martingales.* Trans. Amer. Math. Soc, n°120, pp 359-388 (1965)

2 **Neveu J.:** *Martingales à temps discret,* Masson, (1972)

Définition:

Le processus $(h_t)_{t\geq 0}$ *est une quasi-martingale ssi*

$$\sum_t E \left| E(h_{t+1} - h_t \mid F_t) \right| < +\infty \qquad (3.32)$$

On dispose en outre d'une propriété caractéristique fort utile, dans le cas où (h_t) est un processus aléatoire à valeurs dans $\Re$.

Propriété caractéristique:

Supposons h_t *réel.*

Soit F_t *l'événement* $\{E(h_{t+1} - h_t \mid F_t) > 0\}$,

1) le processus $(h_t)_{t\geq 0}$ *est une quasi-martingale ssi*

$$i) \quad \sum_t E(1_{F_t} (h_{t+1} - h_t)) < +\infty$$

$$ii) \quad \sum_t E(1_{F_t^c} (h_{t+1} - h_t)) > -\infty$$

2) de plus, si

$$iii) \quad \inf_t E(h_t) > -\infty$$

La condition $i)$ *seule implique que* (h_t) *est une quasi-martingale* (3.33)

En effet, on peu écrire

$$E \left| E(h_{t+1} - h_t \mid F_t) \right| = E(1_{F_t} E(h_{t+1} - h_t \mid F_t)) - E(1_{F_t^c} E(h_{t+1} - h_t \mid F_t))$$

Comme F_t est un événement de F_t, l'indicatrice 1_{F_t} est une constante pour l'espérance conditionnelle $E(\bullet|F_t)$. On a donc

$$E(1_{F_t} E(h_{t+1} - h_t \mid F_t)) = E(E(1_{F_t} (h_{t+1} - h_t) \mid F_t)) = E(1_{F_t} (h_{t+1} - h_t))$$

En remplaçant, on démontre le point $1)$.

$$E \left| E(h_{t+1} - h_t \mid F_t) \right| = E(1_{F_t} (h_{t+1} - h_t)) - E(1_{F_t^c} (h_{t+1} - h_t))$$

On démontre le point $2)$ en remarquant que l'inégalité suivante montre que (3.33.i) et (3.33.iii) impliquent (3.33.ii)

$$\inf_t E(h_t) - h_0 \leq \sum_n E(h_{t+1}) - E(h_t) = \sum_n E(1_{F_t}(h_{t+1} - h_t)) + \sum_n E(1_{F_t^c}(h_{t+1} - h_t))$$

ii Théorème de convergence p.s.

Le théorème fondamental s'énonce

Théorème:

Si $(h_t)_{t\geq 0}$ *est une quasi-martingale à valeurs dans* $\Re^p$*, telle que*

$$\mathrm{Sup}_t\ E(|h_t|) < \infty$$

alors il existe une variable aléatoire h_∞*, à valeurs dans* $\Re^p$ *telle que*

$$h_t \underset{t\to\infty}{\longrightarrow} h_\infty \quad \textit{presque sûrement (p.s.)} \tag{3.34}$$

La démonstration de ce théorème est assez complexe. Elle se trouve non seulement dans les références citées ci-dessus, mais aussi dans [1] qui constitue de plus une brillante introduction à l'étude des approximations stochastiques à l'aide de martingales.

On utilisera en fait un corollaire qui découle immédiatement de (3.33) et (3.34).

Corollaire:

Si $(h_t)_{t\geq 0}$ *est un processus aléatoire à valeurs dans* $\Re^+$*, tel que*

i) $\mathrm{Sup}_t\ E(h_t) < \infty$

ii) $\sum_t E(\mathbb{1}_{F_t}\ (h_{t+1}-h_t)) < +\infty$

alors il existe une variable aléatoire h_∞*, à valeurs dans* $\Re^+$ *telle que*

$$h_\infty = \lim_{t\to\infty} h_t \quad \textit{presque sûrement (p.s.)} \tag{3.35}$$

Ces deux conditions sont donc suffisantes pour la convergence presque sûre d'un processus aléatoire.

- La première est une condition de bornitude, parfois difficile à établir; elle signifie que les termes du processus "ne s'éloignent pas trop".
- La seconde exprime que la série formée par les accroissements de notre processus, dont l'espérance sachant le passé est positive, converge.

3.3.1.4 Etude de l'algorithme stochastique.

Enonçons tout de suite le théorème de convergence:

---

1 **Metivier M.:** *Martingales et convergence p.s. d'algorithmes stochastiques,* Dans "Outils et modèles mathématiques pour l'automatique, l'analyse de système et le traitement du signal", Editions du CNRS, vol 1, partie 3, "Techniques probabilistes et automatique et traitement du signal", pp 529-552 (1981)

Théorème:

On suppose que la fonctionnelle C*, définie en (3.2) est continue, dérivable, et qu'il existe un unique minimum* $\mathbf{w}^*$ *vérifiant (3.28), i.e.*

$$\forall \varepsilon>0, \quad \liminf_{|\mathbf{w}-\mathbf{w}^*|>\varepsilon} (\mathbf{w}-\mathbf{w}^*).\nabla C(\mathbf{w}) > 0$$

On suppose de plus que ∇J *est une fonction intégrable, et vérifie la condition* $\forall \mathbf{w}, E_{\mathbf{x}}(\nabla J(\mathbf{x},\mathbf{w})) = \nabla C(\mathbf{w})$.

Les assertions suivantes constituent une condition suffisante de convergence presque sûre de l'algorithme d'optimisation stochastique (3.6) vers l'unique minimum $\mathbf{w}^*$ *de* C.

$$\text{i)} \quad \sum_{t=0}^{+\infty} \varepsilon_t = +\infty, \qquad \sum_{t=0}^{+\infty} \varepsilon_t^2 < +\infty$$

$$\text{ii)} \quad E_{\mathbf{x}}\left(\nabla J(\mathbf{x},\mathbf{w})^2\right) < A^2 + B^2(\mathbf{w}-\mathbf{w}^*)^2 \qquad (3.36)$$

On retrouve les deux conditions présentées en 3.3.1.2. La condition ii) a cependant légèrement évolué: elle concerne les gradients de J, et non le gradient de C.

En s'inspirant de la démonstration de convergence pour le gradient réel, posons

$$h_t = (\mathbf{w}_t-\mathbf{w}^*)^2 \qquad (3.37)$$

En utilisant la formule (3.6), on obtient

$$h_{t+1} = h_t - 2\varepsilon_t(\mathbf{w}_t-\mathbf{w}^*)\nabla J(\mathbf{x}_t,\mathbf{w}_t) + \varepsilon_t^2 \nabla J(\mathbf{x}_t,\mathbf{w}_t)^2$$

On prend alors l'espérance conditionnelle de cette équation sachant F_t. Comme on sait que $E(\nabla J(\mathbf{x}_t,\mathbf{w}_t)|F_t) = E_{\mathbf{x}}(\nabla J(\mathbf{x},\mathbf{w}_t)) = \nabla C(\mathbf{w}_t)$,

$$E(h_{t+1}|F_t) = h_t - 2\varepsilon_t(\mathbf{w}_t-\mathbf{w}^*)\nabla C(\mathbf{w}_t) + \varepsilon_t^2 E(\nabla J(\mathbf{x}_t,\mathbf{w}_t)^2|F_t)$$

On a alors, en appliquant ii) puis (3.28)

$$E(h_{t+1}|F_t) - h_t \le - 2\varepsilon_t(\mathbf{w}_t-\mathbf{w}^*)\nabla C(\mathbf{w}_t) + \varepsilon_t^2 A^2 + \varepsilon_t^2 B^2 h_t \qquad (3.38)$$

$$E(h_{t+1}|F_t) - (1+\varepsilon_t^2 B^2)h_t \le \varepsilon_t^2 A^2$$

Comme dans le cas du gradient réel, on utilise une astuce pour éliminer B. On pose:

$$\Pi_t = \prod_{k=1}^{t-1}(1+\varepsilon_k^2 B^2)^{-1} \underset{t\to\infty}{\longrightarrow} \Pi_\infty, \text{ et définissons } \quad h'_t = h_t \Pi_t$$

On a alors

$$E(h'_{t+1}|F_t) - h'_t = E(h'_{t+1} - h'_t|F_t) \leq \varepsilon_t^2 \prod_{t+1} A^2 \qquad (3.39)$$

et enfin

$$E(\mathbb{1}_{F_t}\ (h'_{t+1}-h'_t))) = E(\mathbb{1}_{F_t} E(h'_{t+1}-h'_t \mid F_t)) \leq \varepsilon_t^2 \prod_{t+1} A^2$$

qui est une série convergente. De plus, (h'_t), positive, est bornée en moyenne. En effet en prenant l'espérance de (3.39), on obtient:

$$E(h'_{t+1}) - E(h'_t) \leq \varepsilon_t^2 \prod_{t+1} A^2$$

$E(h'_t)$-$E(h'_0)$ est donc inférieure à la somme d'une série convergente, donc bornée. Le corollaire (3.35) nous permet donc de conclure h'_t converge p.s., donc que h_t converge p.s. également.

Il reste à montrer que ces suites convergent bien vers 0. En prenant la somme des espérances de (3.38), on a

$$E(h_\infty)\text{-}E(h_0) = \sum_t E(h_{t+1})-E(h_t) \leq$$
$$- 2\sum_t \varepsilon_t E\Big((\mathbf{w}_t-\mathbf{w}^*)\nabla C(\mathbf{w}_t)\Big) + \sum_t \varepsilon_t^2 (A^2+B^2E(h_t))$$

En se rappelant que $E(h_t)$ est bornée, on voit que la série positive

$$\sum_t \varepsilon_t E\Big((\mathbf{w}_t\text{-}\mathbf{w}^*)\nabla C(\mathbf{w}_t)\Big) < +\infty$$

Comme la série de terme ε_t est divergente, et comme $(\mathbf{w}_t\text{-}\mathbf{w}^*)\nabla C(\mathbf{w}_t)$ est positif par hypothèse, on a

$$\liminf_{t\to\infty} (\mathbf{w}_t\text{-}\mathbf{w}^*)\nabla C(\mathbf{w}_t) = 0$$

et donc, d'après la condition (3.25), on conclut que

$$\liminf_{t\to\infty} (\mathbf{w}_t\text{-}\mathbf{w}^*)^2 = \liminf_{t\to\infty} h_t = 0$$

ce qui démontre le théorème.

La technique utilisée montre comment l'utilisation de quasi-martingales permet de préserver précisément, dans le cas stochastique, la structure des démonstrations effectuées dans le cas continu et dans le cas du gradient total.

3.3.2 Cas général.

Les choses deviennent plus complexe si l'on abandonne les hypothèses d'unicité du minimum, et l'hypothèse (3.28).

On cherche alors à montrer la convergence de l'algorithme vers un attracteur stable de (3.23), c'est à dire, en général, vers un minimum local.

Dans le cas continu, comme dans le cas du gradient total, les $\mathbf{w}_t$ *restent confinés dans un même bassin d'attraction* lorsque ε_t devient faible. On démontre alors [1] qu'il existe une fonction U de classe C^2 sur ce bassin d'attraction D, de frontière ∂D, telle que:

$$\begin{aligned} &i) \quad U(\mathbf{w}^*) = 0;\ U(\mathbf{w}) > 0,\ \forall \mathbf{w} \in D,\ \mathbf{w} \neq \mathbf{w}^* \\ &ii) \quad U'(\mathbf{w}) \nabla C(\mathbf{w}) > 0,\ \forall \mathbf{w} \in D,\ \mathbf{w} \neq \mathbf{w}^* \\ &iii) \quad U(\mathbf{w}) \rightarrow +\infty \ \textit{si}\ \ \mathbf{w} \rightarrow \partial D \ \textit{ou}\ \ \mathbf{w}^2 \rightarrow +\infty \end{aligned} \qquad (3.40)$$

Cette fonction permet alors de définir $h_t = U(\mathbf{w}_t)$, et de conclure.

Dans le cas stochastique, on n'obtient pas aisément une telle garantie de confinement. On constate d'ailleurs expérimentalement que ces algorithmes peuvent abandonner un minimum local pour en rejoindre un autre, (souvent plus profond).

L'ouvrage [2] étudie, dans un cadre extrêmement général, la convergence des approximations stochastiques vers les attracteurs stables de leur équation différentielle moyenne. Les résultats qu'il expose reposent sur de nombreuses hypothèses, dont une ((H7) p.238 et (1.6.5) p.240) est une généralisation de (3.40).

Il ne reste plus alors qu'à tenter de majorer la probabilité que notre configuration de poids échappe à un attracteur. On n'obtient plus alors un résultat de convergence presque sûre, mais une majorant de la probabilité de ne pas converger…

3.3.2.1 Convergence vers un minimum local.

Dans notre cadre plus restreint, il est possible de se dispenser d'une telle hypothèse, en utilisant C elle même comme fonction de Lyapunov. Il faut tout d'abord que C *soit minorée*. Posons:

---

1 **Krasovskii A.A:** *Dynamics of Continuous Self-Organizing Systems* - Fizmatgiz Moscow (en Russe) (1963)

2 **Benveniste A., Metivier M., Priouret P.:** *Algorithmes adaptatifs et approximations stochastiques* - Masson (1987)

$$h_t = C(\mathbf{w}_t) - C_{min} \tag{3.41}$$

ou C_{min} est un minorant de C. On a alors, en appliquant (3.6) et en effectuant un développement limité de C par rapport à $\mathbf{w}_t$, et en supposant que *les dérivées secondes de C sont bornées*.

$$h_{t+1} - h_t \leq -\varepsilon_t \nabla C(\mathbf{w}_t)^t . \nabla J(\mathbf{x}_t, \mathbf{w}_t) + \varepsilon_t^2 \nabla J(\mathbf{x}_t, \mathbf{w}_t)^2 K$$

On prend ensuite l'espérance sachant F_t de cette inéquation, en supposant comme d'habitude, que $E_{\mathbf{x}}(\nabla J(\mathbf{x},\mathbf{w})) = \nabla C(\mathbf{w})$ pour tout $\mathbf{w}$.

Si le moment quadratique de ∇J est majoré de la façon suivante,

$$E_{\mathbf{x}}(\nabla J(\mathbf{x},\mathbf{w})^2) \leq \alpha + \beta C(\mathbf{w}) = \alpha + \beta C_{min} + \beta h_t$$

on obtient

$$E(h_{t+1}|F_t) - (1+\varepsilon_t^2 \beta K) h_t \leq -\varepsilon_t (\nabla C(\mathbf{w}_t))^2 + \varepsilon_t^2 (\alpha + \beta C_{min}) K \tag{3.42}$$

$$E(h_{t+1}|F_t) - (1+\varepsilon_t^2 \beta K) h_t \leq \varepsilon_t^2 (\alpha + \beta C_{min}) K$$

en posant à nouveau

$$\Pi_t = \prod_{k=1}^{t-1} (1+\varepsilon_k^2 \beta K)^{-1} \underset{t\to\infty}{\longrightarrow} \Pi_\infty, \text{ et en définissant } \quad h'_t = h_t \Pi_t$$

on obtient

$$E(h'_{t+1}|F_t) - h'_t = E(h'_{t+1} - h'_t|F_t) \leq \varepsilon_t^2 \Pi_{t+1} (\alpha + \beta C_{min}) K \tag{3.43}$$

et comme ci avant,

$$E(\mathbb{1}_{F_t} (h'_{t+1} - h'_t))) = E(\mathbb{1}_{F_t} E(h'_{t+1} - h'_t \mid F_t)) \leq \varepsilon_t^2 \Pi_{t+1} (\alpha + \beta C_{min}) K$$

qui est une série convergente. De plus, (h'_t), positive par définition, est bornée en moyenne. En effet en prenant l'espérance de (3.43), on obtient

$$E(h'_{t+1}) - E(h'_t) \leq \varepsilon_t^2 \Pi_{t+1} (\alpha + \beta C_{min}) K$$

$E(h'_t) - E(h'_0)$ est donc inférieure à la somme d'une série convergente, donc bornée. Le corollaire (3.35) nous permet donc de conclure que h'_t converge p.s., donc que h'_t et $C(\mathbf{w}_t)$ convergent p.s.

En outre, en prenant la somme des espérances de (3.42)

$$E(h_\infty)-E(h_0) = \sum_t E(h_{t+1})-E(h_t) \leq$$
$$- \sum_t \varepsilon_t E\left(\nabla C(\mathbf{w}_t)^2\right) + \sum_t \varepsilon_t^2 K(\alpha+\beta C_{min}+\beta E(h_t))$$

En se rappelant que $E(h_t)$ est bornée, on voit que la série positive

$$\sum_t \varepsilon_t E\left(\nabla C(\mathbf{w}_t)^2\right) < \infty \qquad (3.44)$$

converge. Pour affiner ce résultat, on pose.

$$g_t = (\nabla C(\mathbf{w}_t))^2 \qquad (3.45)$$

et on recommence, avec un développement limité en supposant que *les dérivées troisièmes de* C *sont bornées.*

$$g_{t+1} \leq g_t - 2\varepsilon_t \nabla J(\mathbf{x}_t,\mathbf{w}_t)^t \nabla^2 C(\mathbf{w}_t) \nabla C(\mathbf{w}_t) + \varepsilon_t^2 M \nabla J(\mathbf{x}_t,\mathbf{w}_t)^2$$

$$E(g_{t+1}|F_t) \leq g_t - 2\varepsilon_t \nabla C(\mathbf{w}_t)^t \nabla^2 C(\mathbf{w}_t) \nabla C(\mathbf{w}_t) + \varepsilon_t^2 M (\alpha + \beta C(\mathbf{w}_t))$$

Comme $C(\mathbf{w}_t)$ est une suite convergente, le terme $(\alpha+\beta C(\mathbf{w}_t))$ est majoré. De plus,

$$\left|\varepsilon_t \nabla C(\mathbf{w}_t)^t \nabla^2 C(\mathbf{w}_t) \nabla C(\mathbf{w}_t)\right| \leq \varepsilon_t K (\nabla C(\mathbf{w}_t))^2 = \varepsilon_t K g_t$$

On écrit alors

$$E(\mathbb{1}_{F_t} (g_{t+1}-g_t))) = E(\mathbb{1}_{F_t} E(g_{t+1}-g_t|F_t)) \leq -2E(\varepsilon_t K g_t) + E(\varepsilon_t^2 M (\alpha+\beta C(\mathbf{w}_t)))$$

Le membre de droite est la somme de deux séries convergentes (cf 3.44). De plus, la suite (g_t) est positive par définition, et bornée en moyenne, grâce à l'hypothèse faite pour écrire (3.42). D'après le corollaire (3.35), elle converge donc presque sûrement.

De plus, la divergence de la série ε_t et (3.44) impliquent qu'elle ne peut converger que vers 0.

On peut alors énoncer le résultat suivant.

Théorème:

On suppose que la fonctionnelle $C(\mathbf{w})$*, définie en (3.2) est trois fois dérivable, et que ses dérivées secondes et troisièmes sont bornées.*

On suppose de plus que ∇J *est une fonction intégrable, et vérifie la condition* $\forall \mathbf{w},\ E_{\mathbf{x}}(\nabla J(\mathbf{x},\mathbf{w})) = \nabla C(\mathbf{w})$.

Si les assertions suivantes sont vérifiées,

$$\text{i)} \quad \sum_{t=0}^{+\infty} \varepsilon_t = +\infty, \quad \sum_{t=0}^{+\infty} \varepsilon_t^2 < +\infty$$

$$\text{ii)} \quad E_{\mathbf{x}}\left(\nabla J(\mathbf{x},\mathbf{w})^2\right) < \alpha + \beta C(\mathbf{w}), \quad \alpha,\beta>0$$

$$\text{iii)} \quad C(\mathbf{w}) \geq C_{min}$$

alors

$$C(\mathbf{w}_t) \underset{t\to\infty}{\longrightarrow} C_\infty \ \textit{p.s., et} \qquad (\nabla C(\mathbf{w}_t))^2 \underset{t\to\infty}{\longrightarrow} 0 \ \textit{p.s.} \tag{3.46}$$

La condition ii) exprime cette fois-ci que les dérivées locales, en moyenne, ne croissent pas plus que linéairement avec C. La condition iii) exprime que C est minoré, ce qui est d'ailleurs souhaitable. Il faut en outre que les dérivées secondes et troisièmes de C soient bornées.

En notant $sing(C)$ l'ensemble des points singuliers de C, i.e. tels que $\nabla C=0$, et en notant $d(\mathbf{w},sing(C))$ la distance séparant $\mathbf{w}$ et le point singulier le plus proche, on obtient le résultat suivant:

Corollaire:

Même hypothèses que le théorème (3.46). Si, de plus,

$$\inf_{d(\mathbf{w},sing(C))>\varepsilon} (\nabla C(\mathbf{w}))^2 > 0$$

alors

$$\lim_{t\to\infty} d(\mathbf{w}_t,sing(C)) = 0 \ \textit{p.s.} \tag{3.47}$$

Ce corollaire prouve la convergence de $\mathbf{w}_t$ vers l'ensemble des points singuliers. Ce n'est pas complètement satisfaisant, car ces points singuliers peuvent être des maxima locaux ou des points selles. Notons également que l'hypothèse supplémentaire, introduite en (3.47) implique que

$$\lim_{\mathbf{w}^2\to\infty} \inf (\nabla C(\mathbf{w}))2 > 0 \tag{3.48}$$

c'est à dire que la dérivée est strictement positive à l'infini. Cela est faux dans de nombreux cas.

Dans les perceptrons multi-couches, par exemple, (3.48) n'est vérifiée que si on utilise une sigmoïde penchée, par exemple $(th(x)+\varepsilon x)$. Cela ne semble cependant pas avoir d'influence en pratique !

3.3.2.2 Analogie avec le recuit simulé.

Le résultat (3.47) n'est pas complètement satisfaisant. Les points singuliers peuvent en effet être des maxima locaux, des points selles, ou même de mauvais minima locaux.

Voici un ensemble "d'intuitions physiques", qui tendent à montrer qu'avec le temps, la probabilité que $\mathbf{w}_t$ augmente au voisinage d'un minimum local, et diminue au voisinage d'un maximum local.

Notons $q_t(\mathbf{w})$ la densité de probabilité de $\mathbf{w}_t$. On sait que sous certaines hypothèses, le théorème (3.47) s'applique, et donc que le support de $q_t(\mathbf{w})$ tend vers $sing(C)$ lorsque t tend vers l'infini.

On considérera, en première approximation, que

$$\nabla J(\mathbf{x},\mathbf{w}) = \nabla C(\mathbf{w}) + \xi$$

où ξ est un bruit gaussien de variance σ^2. Cette variance est en général non nulle, sauf lorsque tous les $\nabla J(\mathbf{x},\mathbf{w})$ sont égaux pour un $\mathbf{w}$ donné, et en particulier lorsqu'ils admettent un même minimum local.

Entourons un point quelconque d'une petite boule (cf fig 3.2), et notons q_t la probabilité que $\mathbf{w}_t$ appartienne à cette boule. Supposons également que ε_t est assez petit pour que $\varepsilon_t \nabla J(\mathbf{x},\mathbf{w}_t)$ soit faible devant le diamètre de la boule. Seuls alors sont susceptibles d'entrer ou de quitter la boule les $\mathbf{w}_t$ proches de la surface de la boule.

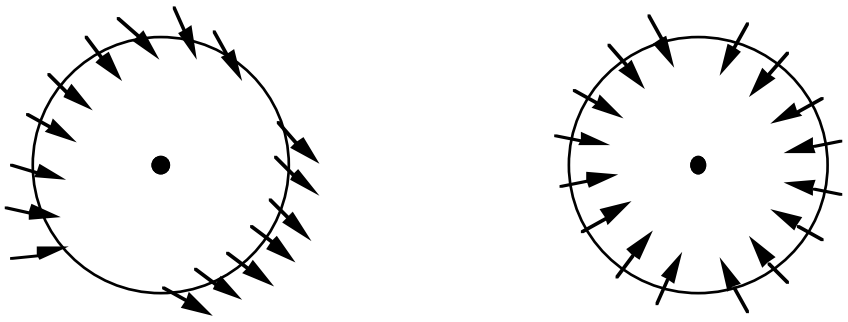


Fig 3.2- Deux points entourés d'une boule. A gauche un point ordinaire. A droite, un maximum local. Les flèches représentent les flux des gradients.

Chaque point de la surface possède une densité $q_t(\mathbf{w}_s)$. Il est responsable d'un accroissement infinitésimal $\varepsilon_t q_t(\mathbf{w}_s)\left(\nabla C(\mathbf{w}_s)+\xi\right)\mathbf{n}\,d\mathbf{w}_s$ de la probabilité q_t de notre boule, où $\mathbf{n}$ est le vecteur normal à la surface de la boule. L'accroissement moyen $q_{t+1}-q_t$ est donc proportionnel à la moyenne du flux de ces produits à travers la surface de la boule, c'est à dire:

$$q_{t+1}-q_t = \int_B \varepsilon_t \, \mathrm{div}\left(q_t(\mathbf{w})\nabla C(\mathbf{w}_s)\right) d\mathbf{w} = \int_B \left(\nabla q_t(\mathbf{w})\nabla C(\mathbf{w}) + q_t(\mathbf{w})\, \mathrm{div}\nabla C(\mathbf{w})\right) \varepsilon_t d\mathbf{w} \qquad (3.49)$$

L'équation (3.49) permet de deviner l'évolution de q_t.

- Autour d'un *minimum local*, $\mathrm{div}\nabla C(\mathbf{w})$, est fortement positif. De plus, $\nabla C(\mathbf{w})$ est a peu près nul. Si elle est non nulle, la densité $q_t(\mathbf{w})$ augmente.

- Autour d'un *maximum local*, $\mathrm{div}\nabla C(\mathbf{w})$, est fortement négatif. De plus, $\nabla C(\mathbf{w})$ est a peu près nul. Si elle est non nulle, la densité $q_t(\mathbf{w})$ décroît. Le terme $\nabla q \nabla C$ peut alors être négligé, et on a alors $\Delta q_t(\mathbf{w}) \approx -K q_t(\mathbf{w})$. La décroissance de $q_t(\mathbf{w})$ est donc exponentielle vers 0 !

- Autour d'un *point selle*, le signe de $\mathrm{div}\nabla C(\mathbf{w})$ reste indéterminé. On ne peut pas dire grand chose.

Supposons maintenant que ε_t décroisse assez lentement pour que l'on ait, comme dans le cas du recuit simulé [1], une équilibre quasi-statique. On a alors une équation d'équilibre:

$$\nabla q_t(\mathbf{w})\nabla C(\mathbf{w}) + q_t(\mathbf{w})\, \mathrm{div}\nabla C(\mathbf{w}) = 0 \qquad (3.50)$$

On sait de plus que la densité est faible au voisinage des maxima, et forte au voisinage des minima. On peut alors tirer des conclusions sur les gradients de probabilité.

- Lorsque la *courbure est positive*, $\mathrm{div}\nabla C(\mathbf{w})$ est positif. Si la densité $q_t(\mathbf{w})$ n'est pas nulle, son gradient est de sens opposé à ∇C. En intégrant, on remarque que *la densité $q_t(\mathbf{w})$ est soit nulle, soit d'autant plus forte que C est faible*. Cela est vrai en particulier sur les minima locaux.

- Lorsque la *courbure est négative*, $\mathrm{div}\nabla C(\mathbf{w})$ est négatif. Si la densité $q_t(\mathbf{w})$ n'est pas nulle, son gradient est de même sens que ∇C. La densité $q_t(\mathbf{w})$ devrait être d'autant plus forte que C est fort. Mais cela devrait aussi être vrai pour les maxima locaux, et nous avons vu qu'il n'en est rien. On en déduit que *la densité est nulle*.

On remarque que le système est en équilibre instable si la densité au voisinage d'un minimum est identiquement nulle. Or, l'approximation stochastique *a introduit un terme de bruit*, ξ, qui rendait impossible le confinement des $\mathbf{w}_t$. Ce même terme de bruit, lorsque sa variance n'est pas nulle, élimine cet équilibre instable de la densité.

---

1 **Kirkpatrick S., Gelatt C.D.Jr, Vecchi M.P.:** *Optimisation by Simulated Annealing* - Science, vol 220, N° 4598, pp 671-680, (1983)

Au fur et à mesure que l'algorithme converge, le support de cette densité converge vers l'ensemble des points singuliers de C, avec une probabilité nulle sur les maxima, et avec une probabilité sur les minima d'autant plus forte que la valeur de C y est faible.

On constate donc un phénomène comparable à celui du recuit simulé: *L'algorithme a tendance à converger vers de bons minima locaux*. C'est en fait la perte de la propriété de confinement qui nous apporte cette bonne nouvelle, maintes fois confirmée par l'expérience [1], et d'une utilité pratique certaine (cf §5.1).

Une formalisation plus rigoureuse et plus satisfaisante de ces raisonnements semble cependant difficile à établir. Il faudrait, en toute rigueur, introduire des outils pour traiter l'aléatoire stochastique dans ces équations. De plus, comme pour le recuit simulé, il n'est pas très rigoureux de supposer que notre système évolue de façon "quasi-statique".

3.3.3 Conclusion

L'enjeu de l'étude mathématique de la convergence des algorithmes de descente stochastique de gradient est de taille: Il s'agit de prouver simultanément la convergence d'un très grand nombre d'algorithmes connexionnistes ou statistiques, présents ou futurs. La méthode de Lyapunov et les propriétés des quasi-martingales permettent d'aborder ce problème ardu, et d'énoncer des théorèmes de convergence généraux.

En outre, ces algorithmes stochastiques possèdent d'importantes propriétés nouvelles, comme le montre l'analogie avec le recuit simulé. Une approche mathématique rigoureuse reste à établir.

En pratique, on se contente de constater que ces algorithmes convergent. On souhaite surtout en améliorer la rapidité. Les preuves mathématiques de convergence sont malheureusement trop abstraites pour donner des indices très utilisables. En prenant quelques précautions, ceux que l'on glane dans le cas de l'algorithme de gradient continu (cf. chp. 5) suffisent souvent à réduire significativement le temps d'apprentissage.

---

1 **Bourrely J.:** *Parallelization of a Neural learning algorithm on a Hypercube* - In "Hypercube and distributed computers", Elsiever Science Publishing, North Holland (1989)

3.4 Application au problème de reconnaissance des formes.

On traite ici le problème d'algorithmes d'apprentissage pour la reconnaissance des formes. Il serait idéal d'optimiser le risque moyen, mais cela s'avère impossible. Trois approches peuvent alors être envisagées:

- *Optimiser une approximation du risque moyen. On montrera que c'est la cas, par exemple, de l'algorithme LVQ2.*

- *Evaluer les vraisemblances conditionnelles de nos classes, à l'aide du principe du maximum de vraisemblance. C'est l'approche privilégiée des statistiques paramétriques*

- *Evaluer les probabilités a posteriori, au moyen du principe des moindres carrés. C'est l'approche utilisée dans l'adaline, et dans le perceptron multi-couches.*

Les systèmes d'apprentissage et d'adaptation ne se limitent pas aux problèmes de reconnaissance des formes (K-means, §3.2.2, est un exemple d'autre application), mais ceux ci donnent une idée des difficultés que l'on rencontre et des solutions que l'on peut apporter.

Le système est confronté à des *entrées* ou *formes*, appartenant à plusieurs *classes*. Il s'agit de déterminer la classe d'une forme, d'une façon optimale. On peut déjà identifier plusieurs problèmes:

- La *nature des formes*. On considère souvent qu'elles peuvent être représentées par un point dans un espace vectoriel. Ce n'est pas nécessairement vrai. Dans le cas de la reconnaissance de la parole, par exemple, on a affaire à un signal, c'est à dire une séquence de vecteurs, virtuellement infinie.

- Le *nombre de classes*. On supposera dans ce chapitre que le nombre de classes est fini. Ce n'est pas toujours vrai non plus: Toujours dans le cas de la reconnaissance de la parole continue, les classes sont, par exemple, des séquences de mots de longueur arbitraire.

- L'*ambiguïté des formes*. Une même forme peut appartenir à plusieurs classes, selon l'état d'une variable extérieure non accessible.

  Cela apparaît souvent lorsque, l'ensemble des informations permettant de classifier nos formes est trop grand pour être traité efficacement. On utilise alors des *prétraitements*, qui réduisent le volume de cette information, sans trop affecter les informations pertinentes pour la classification. Cependant, cela introduit des ambiguïtés.

Il serait prématuré de conclure que les prétraitements sont nuisibles: En simplifiant le système, ils réduisent à la fois le nombre d'exemples et le temps nécessaires à l'apprentissage. (cf. chp. 4).

Nous supposerons dans ce chapitre que le nombre de classes est fini. Cette hypothèse n'est pas absolument nécessaire. Le chapitre 10 traite de l'utilisation des principes dégagés ici à la reconnaissance de la parole, dans laquelle les classes, en nombre dénombrable, sont des séquences de symboles. De plus, la nature des dispositifs impose souvent que les entrées soient vectorielles. Ce n'est pas une obligation, la section 6.5 et le chapitre 10 en fournissent des exemples.

3.4.1 Le risque moyen.

Reprenons le formalisme de l'illustration présentée en 3.1., équation (3.3).

Les vecteurs $\mathbf{x}$ auront donc la forme $(\mathbf{X}, C)$ où $\mathbf{X}$ représente les entrées et C une variable discrète prise dans l'ensemble $\{C_1,\ldots C_N\}$ représentant la classe. On notera $f(\mathbf{X},\mathbf{w})$ la classe identifiée par notre dispositif de reconnaissance des formes, en fonction de l'entrée X et de paramètres $\mathbf{w}$, sur lesquels porte l'apprentissage.

On suppose alors qu'il existe une matrice $J(C_i,C_j)$, représentant le coût ou le bénéfice associé à la classification dans la classe C_j d'une forme appartenant en fait à la classe C_i.

Les termes diagonaux de cette matrice, correspondant à une bonne classification sont usuellement négatifs ou nuls. Les termes non diagonaux sont positifs, et mesurent la gravité de l'erreur commise.

On peut alors écrire la fonction de *risque moyen*:

$$C^{risk} = E_{\mathbf{x}}\big(J(C, f(\mathbf{X},\mathbf{w})) \big) = \int J(C, f(\mathbf{X},\mathbf{w}))\, p(\mathbf{x})\, d\mathbf{x} \qquad (3.51)$$

Ce qui est bien de la forme (3.1). On remarquera que $p(\mathbf{x})$ peut s'écrire des deux façons suivantes:

$$p(\mathbf{x}) = p(\mathbf{X},C) = P(C).p(\mathbf{X}|C) = p(\mathbf{X})\, P(C|\mathbf{X}) \qquad (3.52)$$

L'appartenance de X à la classe C est donc exprimée uniquement par la probabilité $p(\mathbf{x})$, au travers des *vraisemblances conditionnelles* $p(\mathbf{X}|C)$ ou bien des *probabilités a posteriori* $p(C|\mathbf{X})$. Il est donc crucial de considérer que la probabilité $p(\mathbf{x})$ est inconnue.

On ne peut évaluer C^{risk} qu'en constituant un échantillon de couples $(\mathbf{X}_k,C_k)$, et en calculant un coût empirique:

$$\tilde{C}^{risk} = \sum_k J(C_k, f(\mathbf{X}_k,\mathbf{w})) \qquad (3.53)$$

Cela constitue le moyen le plus général pour évaluer la performance du système.

3.4.2 Minimisation du risque moyen.

3.4.2.1 Le risque moyen ne se prête pas à la minimisation.

L'algorithme stochastique (3.6) appliqué à la minimisation de C, s'écrit alors

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \varepsilon_t \nabla J(C, f(\mathbf{X},\mathbf{w})) \tag{3.54}$$

Cette expression pose malheureusement des *problèmes de régularité*. La fonction $J(C,f(\mathbf{X},\mathbf{w}))$ est une fonction de $\mathbf{X}$ en paliers, c'est à dire constante, sauf sur les frontières de classification établies par f où elle est discontinue.

Réciproquement, chaque forme $\mathbf{X}$ induit une partition $\{ f_{(\mathbf{X},\bullet)}^{-1}(C_k) \}$ de l'espace des $\mathbf{w}$ en régions $\Omega_k(\mathbf{X})$ pour lesquelles $\mathbf{X}$ est rangé dans la classe C_k. On peut alors écrire

$$J(C,f(\mathbf{X},\mathbf{w})) = \sum_k J(C,C_k)\ \mathbb{1}_{\Omega_k(\mathbf{X})}(\mathbf{w}) \tag{3.55}$$

d'où

$$\nabla J(C,f(\mathbf{X},\mathbf{w})) = \sum_k J(C,C_k)\ \nabla \mathbb{1}_{\Omega_k(\mathbf{X})}(\mathbf{w}) \tag{3.56}$$

C'est donc une somme de masses de Dirac supportées par les frontières entre les régions $\Omega_k(\mathbf{X})$. On ne peut donc pas appliquer (3.54), car *le gradient de J est presque partout nul*.

Dans le cas de l'algorithme déterministe (3.5), on peut calculer ∇C^{risk}:

$$\nabla C^{risk} = \sum_{i<j} \int_{\Lambda_{i,j}} (J(C,C_i)-J(C,C_j))\ \mathbf{n}_{ij}^{(\mathbf{X})}(\mathbf{w})\ p(\mathbf{x}|X\in\Lambda_{ik})d\mathbf{x} \tag{3.57}$$

en notant Λ_{ik} les frontières de classification établies par f entre les entrées classées C_i et C_j, et $\mathbf{n}_{ij}^{(\mathbf{X})}(\mathbf{w})$ le vecteur normal à la frontière dans l'espace des $\mathbf{w}$ entre $\Omega_i(\mathbf{X})$ et $\Omega_j(\mathbf{X})$. Ce gradient n'est plus presque partout nul: L'intégration possède en effet des vertus régularisantes.

Cependant, $p(\mathbf{x}|X\in\Lambda_{ik})$ est tout aussi inconnu que $p(\mathbf{x})$. On pouvait en (3.5') évaluer C^{risk} en définissant un coût empirique défini sur un échantillon $\{\mathbf{x}_k\}$. Mais, comme la probabilité de l'événement $\{\mathbf{x}\in\Lambda_{ik}\}$ est nulle, aucun élément de notre échantillon ne nous renseigne sur l'intégrale (3.57).

Plus qu'un problème de dérivation, la non continuité de $J(C,f(\mathbf{X},\mathbf{w}))$ nous empêche, en pratique, d'évaluer le gradient du risque moyen. La seule solution consiste donc à ne pas optimiser directement le risque moyen, mais des fonctionnelles ayant soit le même minimum, soit un minimum voisin.

3.4.2.2 Approximations continues du risque moyen: LVQ2.

La solution la plus simple consiste à remplacer l'intégrande (3.55) du risque moyen par une approximation continue.

C'est presque le cas du perceptron: L'intégrande du critère du perceptron (3.14), par exemple, est continu, presque partout dérivable, et identiquement nul lorsqu'il y a bonne classification. Il s'éloigne cependant sensiblement de (3.55) dans le cas d'une mauvaise classification, où il est linéaire en $\mathbf{W}$ et $\mathbf{X}$.

Il est cependant possible d'utiliser explicitement une approximation continue de (3.55), en utilisant un approximation continue des indicatrices des ensembles $\Omega_k(\mathbf{X})$. C'est le cas, par exemple, de l'algorithme LVQ2 (cf. §2.3.5).

Dans cet algorithme, chaque classe est caractérisée par un ensemble fixé de points de référence $\mathbf{w}_k$. Lorsque l'on désire classer un vecteur $\mathbf{X}$, on sélectionne le vecteur de référence le plus proche $\mathbf{w}_{k^*(\mathbf{X})}$, et on regarde la classe $\Phi(k^*(\mathbf{X}))$ qui lui est associée.

On supposera maintenant que la matrice des coûts J est entièrement composée de 1, sauf sa diagonale qui est nulle. Cela revient à considérer toutes les erreurs avec une égale gravité. La formule (3.55) s'écrit donc

$$J(C,f(\mathbf{X},\mathbf{w})) = \sum_{C_k \neq C} \mathbb{1}_{\Omega_k(\mathbf{X})}(\mathbf{w})$$

On peut alors de l'approcher de la façon suivante, en notant $k^{**}(\mathbf{X})$ *l'indice du point le plus proche associé à la bonne classe*.

$$J^{lvq}(\mathbf{x},\mathbf{w}) = \sum_{C_k \neq C} \mathrm{Min}\left\{ \mathbb{1}_{\Omega_k(\mathbf{X})}(\mathbf{w}),\ \frac{(\mathbf{X}-\mathbf{w}_{k^{**}(\mathbf{X})})^2-(\mathbf{X}-\mathbf{w}_{k^*(\mathbf{X})})^2}{\delta\ (\mathbf{X}-\mathbf{w}_{k^*(\mathbf{X})})^2} \right\} \qquad (3.58)$$

Lorsque une entrée $\mathbf{X}$ est mal classée, on approche le risque au voisinage de la frontière de classification proportionnellement à l'écart relatif entre les carrés des distances entre $\mathbf{X}$ et d'une part le vecteur de référence le plus proche associé à la bonne classe, d'autre part le vecteur de référence le plus proche, associé à une mauvaise classe (Fig 3.3).

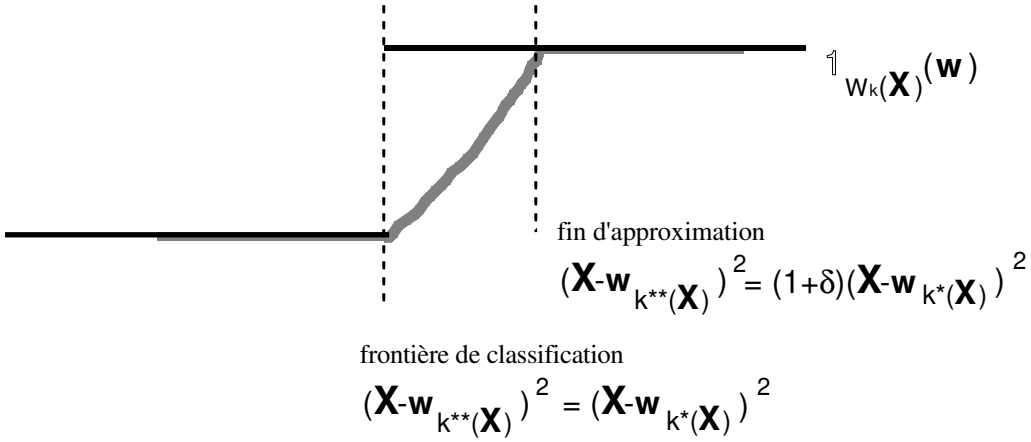


Fig 3.3 - L'approximation du risque moyen dans LVQ2. En noir, la fonction théorique de risque moyen, au voisinage d'une frontière de classification. En gris, l'approximation (3.58).

Le paramètre positif δ contrôle la finesse de l'approximation: plus il est faible, plus J^{lvq} est proche de la valeur exacte (3.55).

Chaque pas de l'algorithme de minimisation stochastique consiste alors à tirer un exemple $\mathbf{x}=(\mathbf{X},C)$, et à appliquer

Si l'exemple est mal classé,

Si $(\mathbf{X}-\mathbf{w}_{k^{**}(\mathbf{X})})^2 < (1+\delta)(\mathbf{X}-\mathbf{w}_{k^{*}(\mathbf{X})})^2$

$$\Delta_t \mathbf{w}_{k^{**}(\mathbf{X})} = \varepsilon_t K_1 (\mathbf{X}-\mathbf{w}_{k^{**}(\mathbf{X})})$$

$$\Delta_t \mathbf{w}_{k^{*}(\mathbf{X})} = -\varepsilon'_t K_2 (\mathbf{X}-\mathbf{w}_{k^{*}(\mathbf{X})}) \qquad (3.59)$$

avec

$$K_1 = \frac{1}{\delta(\mathbf{X}-\mathbf{w}_{k^{*}(\mathbf{X})})^2}, \quad K_2 = K_1 \frac{(\mathbf{X}-\mathbf{w}_{k^{**}(\mathbf{X})})^2}{(\mathbf{X}-\mathbf{w}_{k^{*}(\mathbf{X})})^2}$$

Il est intéressant de comparer cette formule à l'algorithme LVQ2 donné par l'équation (2.35). Il y a deux différences mineures:

- La présence des scalaires K_1 et K_2. Leur l'écart relatif est au plus δ, à cause des conditions d'applications de la formule (3.59).
- La définition de $k^{**}(\mathbf{X})$ est légèrement différente: Il s'agit ici de l'indice du point le plus proche associé à la bonne classe, et non de l'indice du second point le plus proche, si il est associé à la bonne classe.

En pratique, la condition $(\mathbf{X}-\mathbf{w}_{k^{**}(\mathbf{X})})^2<(1+\delta)(\mathbf{X}-\mathbf{w}_{k^{*}(\mathbf{X})})^2$ est rarement vérifiée lorsque ces deux points sont différents. Si δ est assez faible, cette différence ne modifie le fonctionnement de l'algorithme que dans quelques cas extrêmes.

Enfin, on remarque que plus δ est faible, plus la proportion d'itérations de l'algorithme (3.59) qui modifient effectivement les $\mathbf{w}$ est faible. En pratique, on peut commencer avec un δ élevé, et le réduire progressivement. Le remède le plus fréquent consiste à initialiser l'algorithme avec une solution acceptable, et le laisser l'améliorer.

3.4.2.3 Approximations probabilistes.

Une seconde solution consiste à minimiser un critère voisin, qui, sous certaines conditions possède le même minimum. En effet, ayant une forme $\mathbf{X}$, il suffit de choisir comme valeur de $f(\mathbf{X},\mathbf{w})$ la classe qui minimise le terme

$$\sum_{C} J(C,\ f(\mathbf{X},\mathbf{w}))\ p(\mathbf{x}) \qquad (3.60)$$

qui représente le terme à intégrer sur $\mathbf{X}$ dans le risque moyen (3.51). Il est aisé de voir que cette stratégie conduit au risque moyen minimum.

Dans l'équation (3.58), la densité $p(\mathbf{x})$ est inconnue. On peut la décomposer de deux façons, comme en (3.52)

$$p(\mathbf{x}) = p(\mathbf{X},C) = P(C).p(\mathbf{X}|C) = p(\mathbf{X})\ P(C|\mathbf{X})$$

Cela suggère deux méthodes:

- Estimer les vraisemblances $p(\mathbf{X}|C)$. Il est en effet assez facile d'évaluer les probabilités $P(C)$ par comptage.
- Estimer les probabilités a posteriori $P(C|\mathbf{X})$. En effet, la densité $p(\mathbf{X})$ peut être mise en facteur dans la formule (3.60). On n'a pas besoin de la connaître pour déterminer la meilleure classe.

Ces deux méthodes partagent un même défaut: On optimise en fait un critère de qualité sur l'approximation de ces probabilités, et non le risque moyen. Si on n'utilise pas un système assez fin pour évaluer ces probabilités avec une précision suffisante, les optima de ces deux critères peuvent être différents.

Elle possèdent également un avantage commun: Elles fournissent des informations plus riches que l'appartenance à une classe. Ces informations statistiques peuvent être utilement mises à profit en aval du système de reconnaissance.

Les deux sous sections suivantes sont consacrées à l'étude de ces deux méthodes.

3.4.3 Estimation des vraisemblances conditionnelles.

3.4.3.1 Problématique.

Cette méthode consiste en fait à *déterminer la densité inconnue* $p(\mathbf{x})$. Pour cela, on décompose la densité comme suit

$$p(\mathbf{x}) = p(\mathbf{X},C) = P(C).p(\mathbf{X}|C)$$

Les *probabilités a priori,* $P(C)$, ne posent pas problème. Soit on les connaît a priori, soit on les obtient par comptage. En revanche, déterminer les vraisemblances conditionnelles $p(\mathbf{X}|C)$ est plus complexe.

On ne sait en général déterminer une densité à partir d'exemples que si on la connaît à quelques paramètres près. Pour chaque classe C_k, on suppose donc que la densité $p(\mathbf{X}|C_k)$ appartient à une famille paramétrique de fonctions $p_k(\mathbf{X},\mathbf{w})$.

Bien sûr, on n'est que très rarement certain que la densité $p(\mathbf{X}|C_k)$ appartient bien à notre classe de fonction. Certaines familles de fonctions, qualifiées de *robustes*, ont un bon comportement si la densité réelle vérifie certaines conditions. Par exemple, les gaussiennes sont robustes pour des densités à variance bornée.

On cherche alors à déterminer les paramètres $\mathbf{w}^*$ qui correspondent à la densité réelle. La fonction f se déduit alors de (3.60):

$$f(\mathbf{X},\mathbf{w}^*) = \arg\min_{C_k} \sum_j J(C_j,C_k)P(C_j)p_j(\mathbf{X},\mathbf{w}^*) \qquad (3.61)$$

Dans le cas fréquent où les termes non diagonaux de J sont égaux et ses termes diagonaux nuls, c'est à dire lorsque toutes les erreurs sont de même gravité, on peut écrire:

$$f(\mathbf{X},\mathbf{w}^*) = \arg\min_{C_k} \sum_{j\neq k} P(C_j)p_j(\mathbf{X},\mathbf{w}^*) = \arg\max_{C_k} P(C_k)p_k(\mathbf{X},\mathbf{w}^*) \qquad (3.62)$$

3.4.3.2 Principe du maximum de vraisemblance.

L'approche la plus courante pour déterminer une densité à l'intérieur d'une classe paramétrée est le principe du *maximum de vraisemblance* [1] (*Maximum Likelihood Estimator, MLE*), dont voici une version apocryphe. Ce principe est, par exemple, utilisé par les modèles de Markov Cachés exposés dans la section 6.5.

En utilisant l'inégalité $\log(x) \leq x-1$, on peut écrire pour chaque classe C_k

$$\begin{aligned}\int \log \frac{p_k(\mathbf{X},\mathbf{w})}{p(\mathbf{X}|C_k)} p(\mathbf{X}|C_k) d\mathbf{X} &\leq \int \left(\frac{p_k(\mathbf{X},\mathbf{w})}{p(\mathbf{X}|C_k)} - 1 \right) p(\mathbf{X}|C_k) d\mathbf{X} \\ &\leq \int p_k(\mathbf{X},\mathbf{w}) - p(\mathbf{X}|C_k)\, d\mathbf{X} \\ &\leq \int p_k(\mathbf{X},\mathbf{w}) d\mathbf{X} - \int p(\mathbf{X}|C_k) d\mathbf{X} = 0\end{aligned}$$

On en déduit donc que

$$\int \log\big(p_k(\mathbf{X},\mathbf{w})\big)\ p(\mathbf{X}|C_k) d\mathbf{X} \leq \int \log\big(p(\mathbf{X}|C_k)\big)\, p(\mathbf{X}|C_k) d\mathbf{X} \qquad (3.63)$$

et cette inégalité est stricte, sauf si $p_k(\mathbf{X},\mathbf{w}) = p(\mathbf{X}|C_k)$ presque partout.

Si la *vraisemblance conditionnelle est exactement de la forme* $p_k(\mathbf{X},\mathbf{w})$, le paramètre $\mathbf{w}^*$ correspondant peut être déterminé en recherchant le maximum de la fonctionnelle:

$$C_k^{mle} = \int \log\big(p_k(\mathbf{X},\mathbf{w})\big)\ p(\mathbf{X}|C_k) d\mathbf{X}$$

Cette expression peut être interprétée assez facilement. Soit un échantillon de L exemples $(\mathbf{X}_i)$ indépendants dans la classe C_k. La fonctionnelle empirique est alors

$$\tilde{C}_k^{mle} = \sum_{i=1}^{L} \log p_k(\mathbf{X}_i,\mathbf{w}) = \log \prod_{i=1}^{L} p_k(\mathbf{X}_i,\mathbf{w})$$

Maximiser $\tilde{C}_k^{mle}$ consiste donc à maximiser l'estimation calculée avec les $p_k(\mathbf{X},\mathbf{w})$ de la vraisemblance de notre échantillon $(\mathbf{X}_i)$. C'est pourquoi on appelle cette méthode "maximum de vraisemblance".

Il nous faut en fait maximiser simultanément tous les C_k^{mle}. En intégrant l'inégalité (3.63) sur les classes, on obtient

---

1 **Wilks S. S.**: *Mathematical Statistics*, New-York: Wiley (1962)

$$\sum_k P(C_k) \int \log\left(p_k(\mathbf{X},\mathbf{w})\right)\, p(\mathbf{X}|C_k) d\mathbf{X} = \int \log\left(p_C(\mathbf{X},\mathbf{w})\right)\, p(\mathbf{x}) d\mathbf{x}$$
$$\leq \sum_k P(C_k) \int \log\left(p(\mathbf{X}|C_k)\right)\, p(\mathbf{X}|C_k) d\mathbf{X} = \int \log\left(p(\mathbf{X}|C)\right)\, p(\mathbf{x}) d\mathbf{x}$$

Si il existe un paramètre $\mathbf{w}^*$ tel que $p_k(\mathbf{X},\mathbf{w})=p(\mathbf{X}|C_k)$ pour toute classe C_k, il suffit donc, pour le déterminer, de maximiser le critère global C^{mle} suivant:

$$C^{mle} = \sum_k P(C_k)\ C_k{}^{mle} = \sum_k P(C_k) \int \log\left(p_k(\mathbf{X},\mathbf{w})\right) p(\mathbf{X}|C_k) d\mathbf{X}$$

$$C^{mle} = \int \log\left(p_C(\mathbf{X},\mathbf{w})\right)\, p(\mathbf{x}) d\mathbf{x} \qquad (3.64)$$

L'algorithme stochastique correspondant s'écrit alors:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \varepsilon_t \nabla \log p_C(\mathbf{X},\mathbf{w}) = \mathbf{w}_t + \varepsilon_t \frac{\nabla p_C(\mathbf{X},\mathbf{w})}{p_C(\mathbf{X},\mathbf{w})}$$

Lorsque chaque classe de fonctions $p_k(\mathbf{X},\mathbf{w})$ peut utiliser son propre jeu de paramètre, ce principe présente une propriété très pratique. On peut en effet conduire séparément l'optimisation de chaque $C_k{}^{mle}$, et donc déterminer chaque vraisemblance conditionnelle avec seulement des exemples de la classe correspondante! Si le besoin apparaît, en cours d'utilisation, d'ajouter une nouvelle classe, il suffit de déterminer une nouvelle vraisemblance conditionnelle, sans devoir réentrainer les autres.

Cela montre également les limites d'un tel apprentissage: Le système n'apprend pas des caractéristiques permettant de discriminer deux classes, mais essaie de construire des modèle de formes pour chaque classe. On dit qu'un tel algorithme est *peu discriminant*.

3.4.4 Estimation des probabilités a posteriori.

3.4.4.1 Problématique.

Comme ci dessus, cette approche consiste à associer à une forme $\mathbf{X}$ donnée la classe qui minimise le terme (3.60). Cependant, on décompose $p(\mathbf{x})$ d'une autre façon:

$$p(\mathbf{x}) = p(\mathbf{X},C) = P(C|\mathbf{X})\ p(\mathbf{X})$$

Pour une forme donnée, $p(\mathbf{X})$ est bien sûr constant. Il suffit donc d'évaluer les *probabilités a posteriori* $P(C|\mathbf{X})$.

On cherche donc, dans une famille de fonctions paramétriques $P_k(\mathbf{X},\mathbf{w})$, la fonction de cette classe qui approche le mieux $P(C_k|\mathbf{X})$. Une mesure de distance entre ces deux fonctions sera

$$D_k = \int (P_k(\mathbf{X},\mathbf{w})-P(C_k|\mathbf{X}))^2\ p(\mathbf{x})\ d\mathbf{x} \tag{3.65}$$

Ceci est déjà une différence fondamentale avec le cas précédent: Pour appliquer le principe du maximum de vraisemblance, on supposait, peut être à tort, que notre densité *appartenait* à une certaine famille paramétrique. On recherche ici le membre d'une famille paramétrique qui *approche le mieux* les probabilités a posteriori.

La fonction de décision f se déduit également de (3.60):

$$f(\mathbf{X},\mathbf{w}^*) = \arg\min_{C_k} \sum_{C_j} J(C_j,\ C_k)\ P_j(\mathbf{X},\mathbf{w}^*) \tag{3.66}$$

Dans le cas fréquent où les termes non diagonaux de J sont égaux et ses termes diagonaux nuls, on a:

$$f(\mathbf{X},\mathbf{w}^*) = \arg\max_{C_k} P_k(\mathbf{X},\mathbf{w}^*) \tag{3.67}$$

3.4.4.2 Principe des moindres carrés.

On ne peut, cependant, minimiser le critère D_k. En effet, celui ci dépend de $p(C|\mathbf{X})$ que l'on suppose inconnu. Cependant, il est classique de minimiser un critère des moindre carrés pour atteindre le même résultat. (Mean Square Estimator, MSE)

Définissons en effet, avec $\Phi_k(C)$ valant 1 si $C=C_k$, et valant 0 dans les autres cas.

$$C^{mse}_k = \int (\Phi_k(C)-P_k(\mathbf{X},\mathbf{w}))^2\ p(\mathbf{x})d\mathbf{x} \tag{3.68}$$

On voit alors que

$$\begin{aligned} C^{mse}_k &= \int \Big((\Phi_k(C)-P(C_k|\mathbf{X}))+(P(C_k|\mathbf{X})-P_k(\mathbf{X},\mathbf{w}))\Big)^2\ p(\mathbf{x})d\mathbf{x} \\ &= \int \Big(\Phi_k(C)-P(C_k|\mathbf{X})\Big)^2\ p(\mathbf{x})d\mathbf{x} + \int \Big(P(C_k|\mathbf{X})-P_k(\mathbf{X},\mathbf{w})\Big)^2\ p(\mathbf{x})d\mathbf{x} \\ &\quad + \int \Big(\Phi_k(C)-P(C_k|\mathbf{X})\Big)\Big(P(C_k|\mathbf{X})-P_k(\mathbf{X},\mathbf{w})\Big)\ p(\mathbf{x})d\mathbf{x} \end{aligned}$$

Or le dernier terme de cette somme est nul. En effet, en séparant les intégrations sur $\mathbf{X}$ et C, il s'écrit

$$\int_{\mathbf{X}} \Big(P(C_k|\mathbf{X})-P_k(\mathbf{X},\mathbf{w})\Big) \left(\sum_C \Big(\Phi_k(C)-P(C_k|\mathbf{X})\Big)\ P(C|\mathbf{X})\right) p(\mathbf{X})d\mathbf{X}$$

Or, il est aisé de constater que la somme sur C est nulle. On en conclut donc que

$$C^{mse}_k = \int \left(\Phi_k(C)-P(C_k|\mathbf{X})\right)^2 p(\mathbf{x})d\mathbf{x} + D_k = C_{constante} + D_k$$

Il est donc équivalent de minimiser C^{mse}_k (3.68) ou D_k (3.65). En revanche, minimiser C^{mse}_k ne requiert pas la connaissance de $P(C|\mathbf{X})$.

Définissons maintenant un coût global, qui est la somme de tous les coûts C_k. En regroupant les Φ_k et P_k dans des vecteurs $\mathbf{\Phi.}$ et $\mathbf{P.}$, on a

$$C^{mse} = \sum C^{mse}_k = \int \left(\mathbf{\Phi.}(C)-\mathbf{P.}(\mathbf{X},\mathbf{w})\right)^2 p(\mathbf{x})\, d\mathbf{x} \qquad (3.69)$$

On retrouve le critère des moindres carrés qu'utilisent l'adaline et le perceptron multi-couches. On minimise en fait la distance quadratique entre des sorties obtenues $\mathbf{P.}(\mathbf{X},\mathbf{w})$ et des sorties désirées $\mathbf{\Phi.}(C)$.

Une régression avec un *critère de moindre carrés* résulte donc en l'*approximation simultanée des probabilités à posteriori.*

L'algorithme s'écrit alors

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \varepsilon_t \nabla_{\mathbf{w}} \mathbf{P.}(\mathbf{X},\mathbf{w})\ (\mathbf{\Phi.}(C)-\mathbf{P.}(\mathbf{X},\mathbf{w})) \qquad (3.70)$$

On voit dans (3.67) que la précision dans l'approximation des probabilités a posteriori n'est utile pour la classification que dans les situation ambiguës, lorsqu'il faut départager deux classes de probabilités voisines.

En général, la meilleure approximation des probabilités a posteriori n'est donc pas la fonction de notre famille qui produit les meilleures frontières entre classe. Ceci est parfaitement illustré dans le cas de l'adaline. La figure 2.9 montre que la frontière déterminée par la meilleure approximation linéaire des probabilités a posteriori, et la meilleure frontière linéaire peuvent être différentes.

3.4.4.3 Contraintes de normalisation

Dans certains cas, on n'utilise pas les probabilités a posteriori pour effectuer directement une classification, mais pour effectuer des calculs de probabilités. On peut, par exemple, vouloir fixer des seuils de rejet: le système peut alors déclarer qu'il ne peut classifier avec une confiance suffisante.

Dans ces applications, il est capital de respecter la contrainte

$$\sum_k P_k(\mathbf{X},\mathbf{w}) = 1 \qquad (3.71)$$

Il existe deux façons de la faire:

- La méthode dirigiste consiste à choisir les familles paramétriques $P_k(\mathbf{X},\mathbf{w})$ de façon à ce que la contrainte (3.69) soit automatiquement respectée.
- La méthode laxiste consiste à ne pas imposer de contraintes: Comme les P_k doivent approcher les probabilités a posteriori, leur somme devrait tendre vers 1.

Chacune de ces deux méthodes appelle quelques remarques…

i Méthode dirigiste: Maximum de probabilité a posteriori.

Le respect automatique de la contrainte (3.71) permet d'utiliser le principe du maximum de vraisemblance pour déterminer les $P_k(\mathbf{X},\mathbf{w})$. On peut en effet établir une inégalité comparable à (3.63). Pour une forme $\mathbf{X}$ quelconque,

$$\sum_k \log \frac{P_k(\mathbf{X},\mathbf{w})}{P(C_k|\mathbf{X})} P(C_k|\mathbf{X}) \le \sum_k P_k(\mathbf{X},\mathbf{w}) - \sum_k P(C_k|\mathbf{X}) = 0$$

D'où l'on déduit

$$\sum_k \log\,(P_k(\mathbf{X},\mathbf{w}))\; P(C_k|\mathbf{X}) \le \sum_k \log\,(P(C_k|\mathbf{X}))\; P(C_k|\mathbf{X}) \qquad (3.72)$$

Cette inégalité est stricte, sauf si $P_k(\mathbf{X},\mathbf{w})=P(C_k|\mathbf{X})$ presque sûrement. En l'intégrant sur les formes $\mathbf{X}$, on obtient, comme dans la section 3.4.3.2, l'inégalité globale suivante

$$\int \log\,(P_C(\mathbf{X},\mathbf{w}))\; p(\mathbf{x})d\mathbf{x} \le \int \log\,(P(C|\mathbf{X}))\; p(\mathbf{x})d\mathbf{x}$$

Si il existe un paramètre $\mathbf{w}^*$ tel que $P_k(\mathbf{X},\mathbf{w}^*)=P(C_k|\mathbf{X})$ pour toute forme $\mathbf{X}$, il suffit donc, pour le déterminer, de maximiser le critère global C^{map}.

$$C^{map} = \int \log\big(P_C(\mathbf{X},\mathbf{w})\big)\; p(\mathbf{x})d\mathbf{x} \qquad (3.73)$$

Cette variante du principe du maximum de vraisemblance est parfois nommée *Maximum de probabilité A Posteriori (MAP)*. L'algorithme stochastique correspondant est alors

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \varepsilon_t \nabla \log P_C(\mathbf{X},\mathbf{w}) = \mathbf{w}_t + \varepsilon_t \frac{\nabla P_C(\mathbf{X},\mathbf{w})}{P_C(\mathbf{X},\mathbf{w})} \qquad (3.74)$$

Le principe du maximum de probabilité a posteriori est parfois plus facile à appliquer que le principe des moindres carrés. Dans l'algorithme ci dessus, seul le calcul du gradient de la probabilité a posteriori de la classe observée est nécessaire. Dans (3.70), il est nécessaire de calculer les gradients de toutes les probabilités a posteriori.

Le prix de cette diminution de complexité est évidemment l'hypothèse extrêmement forte de l'appartenance des probabilités a posteriori $P(C_k|\mathbf{X})$ à la classe de fonctions $P_k(\mathbf{X},\mathbf{w})$.

ii Méthode laxiste: Convergence vers les contraintes.

On entend parfois le raisonnement suivant: "Comme les P_k approchent les probabilités a posteriori, leur somme tend vers 1.".

Ce raisonnement est malheureusement faux. En effet, les P_k ne tendent pas vers les probabilités a posteriori, mais vers les fonctions $P_k(\bullet,\mathbf{w})$ qui approchent le mieux les probabilités a posteriori, au sens des moindres carrés. Rien ne prouve que la somme de ces fonctions vaut 1.

Cependant, même si l'approximation des probabilités a posteriori est médiocre, il est possible, sous certaines conditions, que la somme des P_k tende vers 1 rapidement.

En effet, si les sorties désirées d_k pour chaque exemple vérifient la propriété $\sum\lambda_k d_k=1$, on a, en notant s_k les sorties obtenues:

$$\Delta\mathbf{w} = -\varepsilon_t\nabla_{\mathbf{w}}\left(\sum_j (s_j-d_j)^2\right) = -\varepsilon_t\sum_j \nabla s_j\,(s_j-d_j)$$

$$\Delta s_k = \nabla s_k{}^T\,\Delta\mathbf{w} = -\varepsilon_t\sum_j \nabla s_k{}^T\nabla s_j\,(s_j-d_j)$$

d'où

$$\Delta\left(\sum_k \lambda_k s_k - 1\right) = -\varepsilon_t\sum_{j,k} \lambda_k\nabla s_k{}^T\nabla s_j\,(s_j-d_j)$$

Si la matrice $[\nabla s_k{}^T.\nabla s_j]$, symétrique, est *définie positive*, toutes ses valeurs propres sont comprises entre α et β, strictement positifs.

On peut alors borner les accroissements de la somme $\sum\lambda_k s_k$

$$-\varepsilon_t\,\alpha\,(\sum_k\lambda_k s_k-\sum_k\lambda_k d_k) \le \Delta\left(\sum_k\lambda_k s_k - 1\right) \le -\varepsilon_t\,\beta\,(\sum_k\lambda_k s_k-\sum_k\lambda_k d_k)$$

Si les pas de gradient ε_t vérifient les conditions usuelles, le comportement de la somme $\sum \lambda_k s_k$ peut être déduit de l'équation differentielle moyenne (3.75).

$$-\alpha \left(\sum_k \lambda_k s_k - 1\right) \leq \frac{1}{\varepsilon_t} \frac{d}{dt}\left(\sum_k \lambda_k s_k - 1\right) \leq -\beta \left(\sum_k \lambda_k s_k - 1\right) \qquad (3.75)$$

En intégrant, $\sum \lambda_k s_k - 1$ est alors compris entre deux fonctions tendant vers 0.

En conclusion, si la matrice $[\, \nabla P_i^T . \nabla P_j \,]$ est définie positive, la somme des P_k tendent vers 1. Cela est en particulier vrai *si les paramètres de chaque* P_k *sont distincts*; c'est par exemple le cas de l'adaline.

3.4.5 Conclusion.

Trois grandes familles d'algorithmes d'apprentissage pour la reconnaissance des formes ont été identifiées:

i) L'évaluation des vraisemblance fournit des *informations probabilistes*, qui permettent de prendre une décision optimale. De plus, elle offre la possibilité d'ajouter de nouvelles classes simplement en disposant d'exemples de ces nouvelles classes.

   Cette méthode requiert l'estimation des vraisemblances conditionelles $p(X|C)$ et des probabilités a priori des classes $P(C)$. Elle revient donc à déterminer entièrement la densité inconnue $p(\mathbf{x})$.

ii) L'évaluation des probabilités a posteriori repose sur *des informations statistique*s *moins fines* que l'apprentissage des vraisemblances, mais suffisantes pour prendre une décision optimale. Il n'est nécessaire que d'estimer les $P(C|X)$, et non les $p(X)$, qui représentent une part significative d'inconnu sur $p(\mathbf{x})$, en particulier si les formes X ne sont pas discrètes.

   On perd cependant la possibilité d'ajouter des classes incrémentalement.

iii) L'optimisation d'une approximation du risque moyen*fournit une décision* en déterminant directement la classe optimale, c'est à dire celle qui présente le moins de risques en moyenne.

En général, le paramètre $\mathbf{w}$ qui permet à notre système d'approcher le mieux les vraisemblances conditionnelles ou les probabilités a posteriori est différent du paramètre $\mathbf{w}^*$ qui fournit le risque moyen minimum.

En s'écartant de la minimisation idéale du risque moyen, nos algorithmes d'apprentissage effectuent une optimisation différente, et captent des informations qui ne sont pas utiles pour déterminer les frontières de classification: Ils sont *moins discriminants*.

On est alors contraint d'utiliser des systèmes bien plus fins si l'on veut obtenir une performance semblable. De tels systèmes plus fins nécessitent intuitivement plus d'exemples pour l'apprentissage. C'est ce que confirment les théorèmes de Vapnik, (cf. §4.4).

Le tableau 3.3 résume la situation.

Critère \ Méthode	**i)** *Evaluer les vraisemblances conditionnelles*	**ii)** *Evaluer les probabilités a posteriori*	**iii)** *Minimiser une approximation du risque moyen*
Informations statistiques en sortie	Oui	Oui	Non
Possibilité d'ajouter facilement des classes	Oui	Non	Non
Capacité discriminante	3ème	2ème	1er

Fig 3.3 - Tableau comparatif de trois méthodes d'apprentissage pour la reconnaissance des formes.

La méthode i), l'approximation des vraisemblances, est très largement utilisée dans le contexte des statistiques paramétriques.

La plupart des perceptrons multi-couches appliqués à des problèmes de classification utilisent la méthode ii) et le principe des moindres carrés.

La méthode iii) n'est guère utilisée que par le perceptron et LVQ2.

4 Apprendre avec un nombre limité d'exemples.

4.1 Ressources limitées.

En pratique, on dispose rarement d'un nombre illimité d'exemples. On optimise donc un critère empirique, défini sur ces exemples, et on espère atteindre un point voisin de l'optimum du critère réel.

Les hypothèses asymptotiques signifient à la fois que l'on peut tirer des exemples $\mathbf{x}$ de la distribution $p(\mathbf{x})$ pour un coût négligeable, et que l'on est prêt à itérer indéfiniment un algorithme d'optimisation.

Ce sont des hypothèses trop optimistes. En effet,

- Dans le cas d'algorithmes d'adaptation, les exemples $\mathbf{x}$ sont fournis gratuitement, avec une périodicité définie, lors du fonctionnement du système. Il faut, en revanche, s'assurer que l'algorithme va tendre assez rapidement vers l'optimum, c'est à dire que C sera proche du minimum après le plus petit nombre de tirages d'exemples $\mathbf{x}$.

  Un système de réduction de bruit sur une ligne téléphonique, par exemple, ne dispose que de quelques secondes pour réagir lorsque la nature du bruit de fond change.

- Dans le cas d'algorithmes d'apprentissage, on est disposé à attendre longtemps, mais pas indéfiniment. De plus, les exemples ont été collectés préalablement, et leur nombre est presque toujours limité.

Dans un système de détection de pannes, par exemple, on cherche à identifier les anomalies de fonctionnement d'un boîtier électronique à l'aide de de son comportement extérieur. Pour créer chaque exemple d'apprentissage, il faut démonter un boîtier défectueux et tester tous ses composants.

Vladimir Vapnik est, a ma connaissance, le premier auteur qui se soit écarté significativement des hypothèses asymptotiques. L'ouvrage [1] constitue le fondement de ce chapitre. Il contient des versions commentées (Chapitres 6, 7 et leurs appendices, pages 139-231) des démonstrations, assez techniques, des théorèmes de convergence uniforme.

L'intérêt pratique de ces théorèmes ne réside pas dans leurs aspects quantitatifs. Ils mettent cependant en lumière, et expliquent des phénomènes bien corroborés par l'expérience.

4.1.1 Apprentissage avec répétitions.

Dans le cas d'un algorithme d'apprentissage, on dispose d'un ensemble d'exemples $\{\mathbf{x}_k\}$ tirés au hasard. Ceux ci constituent un échantillonnage de la densité $p(\mathbf{x})$. On peut alors appliquer un algorithme d'apprentissage stochastique en tirant au hasard un exemple dans notre échantillon.

Chaque exemple sera donc utilisé plusieurs fois pour appliquer l'algorithme d'adaptation. Dans les deux cas, cela revient à utiliser une densité empirique discrète

$$\tilde{p}(\mathbf{x}) = \frac{1}{L} \sum_{k=1}^{L} \delta(\mathbf{x}-\mathbf{x}_k) \qquad (4.1)$$

où δ représente comme toujours la masse de Dirac. On effectue donc la minimisation d'une fonctionnelle empirique

$$\tilde{C} = \int J(\mathbf{x},\mathbf{w})\ \tilde{p}(\mathbf{x})d\mathbf{x} = \frac{1}{L} \sum_{k=1}^{L} J(\mathbf{x}_k,\mathbf{w}) \qquad (4.2)$$

et on espère que la fonctionnelle réelle (3.2)

$$C = \int J(\mathbf{x},\mathbf{w})\ p(\mathbf{x})d\mathbf{x}$$

va prendre des valeurs proches du minimum lorsque l'on minimise la fonctionnelle empirique (4.2).

---

1 **Vapnik V.N.:** *Estimation of Dependences Based on Empirical Data* - Springer Series in Statistics, Springer Verlag (1982)

4.2 Difficultés du problème.

On étudie ici un problème typique: la détermination approchée d'une densité inconnue à partir d'exemples. Le théorème de Glivenko-Cantelli permet d'approcher arbitrairement bien les fonctions de répartition. En revanche, un problème de continuité nous interdit de faire de même avec les densités.

4.2.1 Restauration d'une densité inconnue.

La méthode d'évaluation des vraisemblances pour le problème de reconnaissance des formes a été présentée dans la section 3.4.3.. On a souligné alors qu'il s'agissait d'une approche *peu discriminante*: Il est possible d'ajouter une classe, en ayant seulement à effectuer un apprentissage partiel avec des exemples de cette classe.

Un tel algorithme fonctionne en construisant une approximation de la densité $p(\mathbf{x})$. Comme on considère que ce nombre d'exemples est limité, on notera $\varphi_n(\mathbf{x})$ une approximation obtenue à partir de n exemples

Il y a plusieurs façons de mesurer la proximité de $\varphi_n(\mathbf{x})$ et de la densité $p(\mathbf{x})$. Dans un système de décision, cependant, on utilise les valeurs de cette approximation $\varphi_n(\mathbf{x})$, et non des expressions intégrales. On s'intéresse donc à approcher $p(\mathbf{x})$ au sens de la *mesure uniforme*:

$$d(\varphi_n,p) = \sup_{\mathbf{x}} \left|\varphi_n(\mathbf{x})-p(\mathbf{x})\right| \tag{4.3}$$

On cherche donc à identifier les conditions sous lesquelles la quantité (4.3) tend vers 0 lorsque n tend vers l'infini. Cette propriété dépend évidemment de la façon d'obtenir l'approximation à partir de n exemples.

4.2.2 Théorème de Glivenko-Cantelli.

On note $\Re^p_+$ le quadrant positif de l'espace vectoriel $\Re^p$. On définit alors la *fonction de répartition* d'une variable aléatoire vectorielle $\mathbf{x}$ par

$$F(\xi) = P\{\xi-\mathbf{x}\in \Re^p_+\} = \int \mathbb{1}_{\Re^p_+}(\xi-\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{4.4}$$

C'est en fait la probabilité que $\mathbf{x}$ soit situé "en dessous à gauche" de ξ.

Si l'on dispose d'un échantillon $\{\mathbf{x}_k\}$ de la variable $\mathbf{x}$, on définit la *fonction de répartition empirique* en comptant la proportion d'éléments de l'échantillon qui se trouvent en dessous à gauche de ξ. En d'autres termes, c'est la fréquence de l'événement $\{\xi - \mathbf{x} \in \Re^p_+\}$ dans l'échantillon.

Le théorème de Glivenko-Cantelli exprime la possibilité, si l'on dispose d'un échantillon suffisant, d'évaluer la fonction de répartition de cette variable, avec une précision fixée.

Théorème (Glivenko-Cantelli [1])

Soit $\mathbf{x}$ *une variable aléatoire*, $F(\xi)=P\{\xi - \mathbf{x} \in \Re^p_+\}$ *sa fonction de répartition. Soit* $F_n(\xi)$ *la fonction de répartition empirique de* $\mathbf{x}$, *mesurée sur un échantillon aléatoire* $\{\mathbf{x}_1 \ldots \mathbf{x}_n\}$. *Alors*

$$P\left\{ \sup_\xi \left| F(\xi) - F_n(\xi) \right| \underset{n\to\infty}{\longrightarrow} 0 \right\} = 1 \qquad (4.5)$$

Si on se fixe un taux d'erreur admissible ε, il existe une taille d'échantillon n_0 au delà de laquelle la différence entre la fonction de répartition et son approximation empirique sera presque sûrement inférieure à ε.

Ce théorème signifie donc que l'on peut, si on dispose d'un échantillon suffisant, évaluer les probabilités d'une famille d'événements $\{\xi - \mathbf{x} \in \Re^p_+\}$ aussi précisément que l'on veut. *Cela est en particulier vrai pour les fonctions de répartition.*

Ce résultat fort nous permet-il d'évaluer la densité p avec précision? Il faut pour cela résoudre l'équation intégrale suivante pour trouver les densités φ_n.

$$\int \mathbb{1}_{\Re^p_+}(\xi - \mathbf{x}) \varphi_n(\mathbf{x}) d\mathbf{x} \; = F_n(\xi) \qquad (4.6)$$

Cette équation est en général instable. L'opérateur qui à φ_n associe F_n est continu pour la mesure uniforme, mais sa réciproque ne l'est en général pas. La convergence uniforme de F_n vers F ne garantit absolument pas la convergence uniforme de φ_n vers p.

Voici un contre-exemple: En dimension 1, on a $\varphi = dF/d\xi$. Or la dérivation n'est pas un opérateur continu pour la mesure uniforme. Par exemple, lorsque n tend vers l'infini, la suite $F_n = {}^1/_n \sin(nx)$ tend uniformément vers 0. La suite de ses dérivées $F'_n = \cos(nx)$ ne tend pas vers 0.

Il n'y a donc *pas d'espoir de résoudre cette équation de façon approchée,* si on ne dispose que d'une valeur approchée de F, aussi bonne soit elle. En conclusion, il n'est pas possible, sans hypothèses supplémentaires, d'estimer uniformément une *densité*.

---

1 **Cantelli F.P.:** *Sulla determinazione empirica della legi di probabilita*, Giornale dell'Instituto Italiano degli Attuari, n°4 (1933)

Cependant, le théorème de Glivenko-Cantelli montre que l'on peut estimer les *probabilités* de certains évènements, de la forme $\{\xi - \mathbf{x} \in \Re^p_+\}$. Ce résultat est généralisé par le théorème de Vapnik et Chervonenkis dans la section 4.4.1.

4.2.3 Interprétation intuitive.

On retrouve, et ce n'est pas un hasard, une problématique comparable à celle de la généralisation et de l'interpolation, abordées dans la section 2.4..

Quel que soit le nombre d'exemples, rien ne garantit que la densité inconnue $p(\mathbf{x})$ ne contient pas de *grande déviations* (Fig 4.1), que les exemples ne permettent pas de détecter.

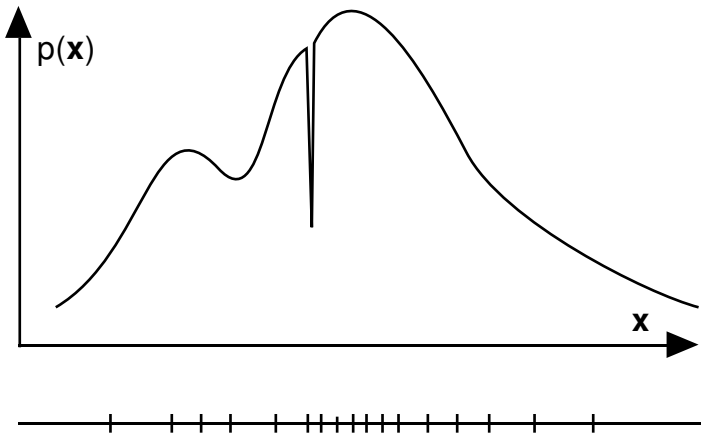


Fig 4.1 - Un échantillon sur une densité de probabilité ne permet pas toujours de détecter une grande déviation.

En revanche, ces grandes déviations ont moins d'influence sur la probabilité d'un évènement $\int \mathbb{1}_E(\mathbf{x})p(\mathbf{x})d\mathbf{x}$, ou même sur un coût moyen $\int J(\mathbf{x})p(\mathbf{x})d\mathbf{x}$.

Plus une famille paramétrée d'évènements $\{E_\mathbf{w}\}$ ou de coûts locaux $\{J(\bullet,\mathbf{w})\}$ est restreinte, moins elle a de chance de souffrir de ces grandes déviations.

Il y a donc un compromis à faire sur la taille de ces familles, c'est à dire la finesse de notre système d'apprentissage. Elle doit être assez vaste pour contenir une solution acceptable, assez restreinte pour autoriser l'apprentissage à partir d'un nombre limité d'exemples.

4.3 Un problème de convergence uniforme.

Evaluer l'erreur que l'on commet en optimisant un critère empirique et non le critère réel, requiert un résultat de convergence uniforme en probabilité du critère empirique vers le critère réel.

4.3.1 Limite en probabilité de la fonctionnelle empirique.

Le processus d'apprentissage consiste donc à minimiser la fonctionnelle empirique $\tilde{C}_L(\mathbf{w})$ définie sur L exemples $\{\mathbf{x}_k\}$ aléatoires.

$$\tilde{C}_L(\mathbf{w}) = \int J(\mathbf{x},\mathbf{w})\ \tilde{p}(\mathbf{x})d\mathbf{x} = \frac{1}{L}\sum_{k=1}^{L} J(\mathbf{x}_k,\mathbf{w})$$

La loi faible des grand nombres nous apprend que si $J(\mathbf{x},\mathbf{w})$ est une variable aléatoire réelle, la fonctionnelle empirique converge en probabilité vers la fonctionnelle réelle, i.e.

$$\forall\varepsilon>0,\quad P\left\{\ |\tilde{C}_L(\mathbf{w}) - C(\mathbf{w})| > \varepsilon\ \right\} \underset{L\to\infty}{\longrightarrow} 0 \qquad (4.7)$$

Ce résultat est affiné par l'inégalité de Hoeffding [1], qui s'applique si on connait un majorant τ de $(\sup C(\mathbf{w}) - \inf C(\mathbf{w}))$

$$\forall\varepsilon>0,\quad P\left\{\ |\tilde{C}_L(\mathbf{w}) - C(\mathbf{w})| > \tau.\varepsilon\ \right\} \le 2e^{-2\varepsilon^2 L} \qquad (4.8)$$

Mais on ne sait pas pour autant que le minimum du coût empirique converge vers le minimum du coût réel. Rien ne prouve en effet que cette convergence se fait à la même vitesse pour deux valeurs de $\mathbf{w}$. Rien ne laisse donc supposer que

$$C(\mathbf{w}_1) < C(\mathbf{w}_2) \iff \tilde{C}_L(\mathbf{w}_1) < \tilde{C}_L(\mathbf{w}_2) \quad \textit{pour } L \textit{ assez grand}$$

4.3.2 Convergence uniforme en probabilité.

Si l'écart maximal entre la fonctionnelle empirique $\tilde{C}_L(\mathbf{w})$ et la fonctionnelle réelle $C(\mathbf{w})$ est majoré,

$$\sup_{\mathbf{w}} |\tilde{C}_L(\mathbf{w}) - C(\mathbf{w})| \le \varepsilon$$

alors l'écart entre l'optimum empirique $\tilde{C}_L(\mathbf{w}^*_{emp}) = \inf \tilde{C}_L$ et l'optimum réel $\inf C$ est également majoré.

$$|\ C(\mathbf{w}^*_{emp}) - \inf C\ | \le |\ C(\mathbf{w}^*_{emp}) - \tilde{C}_L(\mathbf{w}^*_{emp})| + |\ \inf \tilde{C}_L - \inf C\ | \le 2\varepsilon \qquad (4.9)$$

---

1 **Hoeffding W.:** *Probability inequalities for sums of bounded random variables* - J. Amer. Statist. Ass., vol 58, pp 13-30 (1963)

On sait alors que le minimum de la fonctionnelle empirique est proche du minimum de la fonctionnelle réelle. La notion de convergence du coût empirique $\tilde{C}_L(\mathbf{w})$ vers le coût réel $C(\mathbf{w})$ dont nous avons besoin s'écrit donc

$$\forall \varepsilon>0, \quad P\left\{ \sup_{\mathbf{w}} |\tilde{C}_L(\mathbf{w}) - C(\mathbf{w})| \geq \varepsilon \right\} \underset{L\to\infty}{\longrightarrow} 0 \qquad (4.10)$$

On l'appelle *convergence uniforme en probabilité*.

Grâce aux théorèmes de Vapnik et Chervonenkis, on connaît la condition nécessaire et suffisante sur C pour qu'une telle convergence ait lieu. Cette condition dépend de la densité $p(\mathbf{x})$, qui est inconnue. Ces mêmes théorèmes fournissent heureusement une *condition suffisante* ne dépendant que de la forme de $J(\mathbf{x},\mathbf{w})$.

4.4 Théorèmes de Vapnik et Chervonenkis.

Les deux théorèmes suivants généralisent le théorème de Glivenko-Cantelli en donnant des conditions suffisantes de convergence uniforme en probabilité d'une part des fréquences empiriques vers les probabilités, et d'autre part des moyennes empiriques vers les espérances mathématiques.

Ces conditions suffisantes sont liées à une mesure de la capacité de nos systèmes d'apprentissage.

4.4.1 Conditions suffisantes de convergence uniforme des fréquences des événements vers leur probabilité.

On commence par étudier le cas où la fonction $J(\bullet,\mathbf{w})$ est la fonction indicatrice d'un événement $S(\mathbf{w})$, c'est à dire

$$J(\mathbf{x},\mathbf{w}) = 1 \text{ } si \text{ } \mathbf{x}\in S(\mathbf{w}),\ 0 \text{ } si \text{ } x\notin S(\mathbf{w}).$$

La fonctionnelle empirique $\tilde{C}_L(\mathbf{w})$ représente donc la fréquence de l'événement $S(\mathbf{w})$, mesurée sur un échantillon aléatoire de taille L. La fonctionnelle réelle $C(\mathbf{w})$ mesure la probabilité de cet événement.

On peut d'ores et déjà conclure positivement dans certains cas:

- Si la famille d'événements $S(\mathbf{w})$ est finie, le résultat (4.7) implique de façon évidente (4.10).

- Le théorème de Glivenko-Cantelli enseigne que (4.10) est vraie lorsque l'on considère la famille d'événements infinie $S(\xi)=\{\xi-\mathbf{x}\in\Re^p_+\}$. On a en effet, d'après (4.5) convergence presque sûre, qui implique la convergence en probabilité que nous recherchons (4.10).

Mais il existe également beaucoup de cas où l'on peut conclure négativement.

- Si on considère la famille de tous les événements, c'est à dire la famille de toutes les parties de $\Re^p$. Si la densité $p(\mathbf{x})$ est continue, les complémentaires de nos échantillons de taille L constituent des événements de probabilité 1. Or leurs fréquences empirique est toujours 0.

On recherche donc une condition sur la famille d'événements $S(\mathbf{w})$ pour que la propriété (4.10) soit vraie. On souhaite de plus que cette condition soit indépendante de la densité $p(\mathbf{x})$!

4.4.1.1 Capacité d'un ensemble d'événements.

Le théorème de convergence uniforme des fréquences des événements vers leur probabilités [1] détermine une borne sur les probabilités, qui dépend du *nombre maximum de séparations* $m_S(L)$ sur L points associée à notre classe d'événements S:

Soit un ensemble de points $\{\mathbf{x}_k, 1\le k\le L\}$. Chaque événement sépare cet ensemble en points appartenant à cet événement, et points qui n'y appartiennent pas. Notre famille d'événements S partage donc ces points d'un nombre fini de façons

$$0 \le M_S(\{\mathbf{x}_k, 1\le k\le L\}) \le 2^L \qquad (4.11)$$

Le nombre maximum de séparations $m_S(L)$ sur L points associé à notre classe d'événements S est donc

$$0 \le m_S(L) = \sup_{\{\mathbf{x}_k, 1\le k\le L\}} M_S(\{\mathbf{x}_k, 1\le k\le L\}) \le 2^L \qquad (4.12)$$

La formule (2.4) donne l'expression de ce nombre pour des séparations linéaires, c'est à dire pour la classe des événements $\{\mathbf{w}^T\mathbf{x}>\beta\}$, c'est à dire des demi-espaces.

Il existe un analogue de la formule (2.4) dans le cas général: Vapnik et Chervonenkis montrent qu'il existe une limite $\dim_{VC}(S)$, *finie ou infinie*, telle que

---

1 **Vapnik V.N., Chervonenkis A.Ya.:** *On the uniform convergence of relative frequencies of events to their probabilities* - Theory of Probability and its Applications, vol 16, n°2, pp 264-280 (1971)

si $L \le \dim_{VC}(S)$, $\quad m_S(L) = 2^L$

si $L > \dim_{VC}(S) \quad m_S(L) \le \sum_{i=1}^{\dim_{VC}(S)} C_L^i$ (4.13)

La grandeur $\dim_{VC}(S)$ s'appelle *dimension de Vapnik et Chervonenkis* ou, plus brièvement, *capacité* de la famille d'événements S. Lorsque cette capacité est finie, le nombre maximum de séparation, pour $L > \dim_{VC}(S)$, est majoré par

$$m_S(L) \le 1.5 \frac{L^{\dim_{VC}(S)}}{\dim_{VC}(S)!} \tag{4.14}$$

c'est à dire par un polynôme de degré $\dim_{VC}(S)$.

Déterminer la capacité d'une famille d'événement est parfois fastidieux et difficile. Voici quelques exemples simples.

- La formule (2.4) nous apprend que la capacité de la classe des séparateurs linéaires dans un espace de dimension n est $n+1$.

- Cela s'étend à toutes les classes d'événements de la forme $\{\Sigma w_k \varphi_k(\mathbf{x}) > \beta\}$, où les K fonctions $\varphi_k(\mathbf{x})$ sont indépendantes. La capacité d'une telle classe est alors $K+1$. C'est en particulier le cas des polynômes de degré inférieur à d.

- La classe des événements délimités par un polyèdre convexe possède une capacité infinie. Il suffit en effet de considérer L points situés sur la surface d'une boule. Toute partie P de ces points est contenue dans le polyèdre ayant pour sommets les points de P. Ce polyèdre exclut tous les autres points. Pour tout L, le nombre maximum de séparations $m_S(L)$ est donc 2^L.

4.4.1.2 Théorème de convergence uniforme.

Une suite laborieuse de majorations permet de montrer, en notant $\tilde{P}_L\{S(\mathbf{w})\}$ la fréquence de l'événement $S(\mathbf{w})$ que

$$P\left\{ \sup_{\mathbf{w}} |\tilde{P}_L\{S(\mathbf{w})\} - P\{S(\mathbf{w})\}| \ge \varepsilon \right\} \le \eta_{L,\varepsilon} = 6\, m_S(2L)\, e^{-\varepsilon^2 L/4} \tag{4.15}$$

Lorsque la capacité de la classe S est finie et vaut h, $m_S(2L)$ est majoré par un polynôme en L (4.14). On a alors

$$P\left\{ \sup_{\mathbf{w}} |\tilde{P}_L\{S(\mathbf{w})\} - P\{S(\mathbf{w})\}| \ge \varepsilon \right\} \le \eta_{L,\varepsilon} = 9 \frac{(2L)^h}{h!} e^{-\varepsilon^2 L/4}$$

La borne $\eta_{L,\varepsilon}$ tend vers 0 lorsque L augmente. On a donc *convergence uniforme des fréquences des événements vers leur probabilités*. Ce résultat constitue une généralisation du théorème de Glivenko-Cantelli à une classe arbitraire d'événements, de capacité finie.

Il existe en fait plusieurs versions de cette majoration: Plusieurs auteurs [1], [2] ont proposé des bornes différentes, qui s'adaptaient mieux à leurs problèmes. A l'aide de la formule de Stirling $\ln(h!) \approx h.\ln(h)-h$, on peut réécrire cette équation sous la forme suivante:

Théorème (Vapnik-Chervonenkis, [3])

Si $\dim_{VC}(S)=h$ *et* $L>h$, *alors*

$$P\left\{ \sup_{\mathbf{w}} |\tilde{P}_L\{S(\mathbf{w})\} - P\{S(\mathbf{w})\}| \geq 2\sqrt{\frac{h(\ln\frac{2L}{h}+1)-\ln\frac{\eta}{9}}{L}} \right\} \leq \eta \quad (4.16)$$

Ayant un échantillon de taille L, avec une fiabilité η fixée, ce théorème donne un majorant de l'écart entre la fonctionnelle empirique et la fonctionnelle réelle.

4.4.2 Convergence uniforme des moyennes empiriques vers les espérances mathématiques.

Continuons donc en abordant le cas ou $J(\mathbf{x},\mathbf{w})$ est une fonction réelle arbitraire. Cela nous permet de vérifier, par exemple, qu'il est possible d'obtenir par apprentissage un système qui évalue, par exemple, des probabilités a posteriori.

L'idée consiste en fait à appliquer le théorème (4.15) à un ensemble d'événements bien choisis. On commence par supposer que la fonction J est bornée.

$$\forall\ \mathbf{x},\mathbf{w},\quad J_{min} \leq J(\mathbf{x},\mathbf{w}) \leq J_{max}$$

on notera

$$\tau = J_{max} - J_{min}$$

---

1 **Devroye L.:** *Bounds for the uniform deviation of empirical measures* - J. Multivariate Anal., vol 12, pp 72-79 (1982)

2 **Alexander K.S.:** *Probability inequality for empirical processes and a law of the iterated logarithm* - Ann. Prob. vol 12, pp 1041-1067, (1984).

3 **Vapnik V.N.:** *Estimation of Dependences Based on Empirical Data* - Springer Series in Statistics, Springer Verlag (1982)

En considérant les événements de la forme

$$S(\mathbf{w},\lambda) = \left\{ \mathbf{x} \ \textit{tels que} \ \ J(\mathbf{x},\mathbf{w}) > \lambda \right\} \quad \lambda \in [J_{min}, J_{max}]$$

on peut écrire les fonctionnelles réelles et empiriques sous la forme d'intégrales de Lebesgue:

$$C(\mathbf{w}) \ = J_{min} + \lim_{n\to\infty} \sum_{i=1}^{n} \frac{\tau}{n} \ P\left\{S\left(\mathbf{w},J_{min}+\frac{i\tau}{n}\right)\right\}$$

$$\tilde{C}_L(\mathbf{w}) = J_{min} + \lim_{n\to\infty} \sum_{i=1}^{n} \frac{\tau}{n} \ \tilde{P}\left\{S\left(\mathbf{w},J_{min}+\frac{i\tau}{n}\right)\right\}$$

On a donc

$$|\tilde{C}_L(\mathbf{w}) - C(\mathbf{w})| \le \lim_{n\to\infty} \sum_{i=1}^{n} \frac{\tau}{n} \ \left|\tilde{P}\left\{S\left(\mathbf{w},J_{min}+\frac{i\tau}{n}\right)\right\} - P\left\{S\left(\mathbf{w},J_{min}+\frac{i\tau}{n}\right)\right\} \right|$$

En appliquant (4.15), on obtient une majoration de la probabilité de *convergence uniforme des moyennes empiriques vers les espérances mathématiques*.

$$P\left\{ \sup_{\mathbf{w}} |\tilde{C}_L(\mathbf{w}) - C(\mathbf{w})| \ge \tau.\varepsilon \right\} \le \eta_{L,\varepsilon} = 6\, m_S(2L)\, e^{-\varepsilon^2 L/4} \qquad (4.17)$$

Ce résultat est en fait une généralisation de l'inégalité de Hoeffding (4.8), appliquée uniformément à une classe de fonctions.

Lorsque la capacité de la classe $S(\mathbf{w},\lambda)$ est finie et vaut h, il est possible de donner à ce résultat une forme comparable au théorème (4.16).

Théorème (Vapnik-Chervonenkis, [1])

Si $\dim_{VC}\left(S(\mathbf{w},\lambda)\right)=h$ *et* $L>h$, *alors, avec* $\tau = J_{max}-J_{min}$,

$$P\left\{ \sup_{\mathbf{w}} |\tilde{C}_L(\mathbf{w}) - C(\mathbf{w})| \ge 2\tau \sqrt{\frac{h(\ln\frac{2L}{h}+1)-\ln\frac{\eta}{9}}{L}} \right\} \le \eta \qquad (4.18)$$

---

1 **Vapnik V.N.:** *Estimation of Dependences Based on Empirical Data* - Springer Series in Statistics, Springer Verlag (1982)

Ce résultat généralise (4.16) à toutes les fonctions de coût de la forme (3.2), c'est à dire à tous les algorithmes d'apprentissages étudiés ici.

4.4.3 Optimisation du coût empirique.

On peut alors utiliser la formule (4.9), et écrire

$$P\left\{ \; | \, C(\mathbf{w}^*_{emp}) - \inf C \, | \leq 4\tau \sqrt{\frac{h(\ln\frac{2L}{h}+1)-\ln\frac{\eta}{9}}{L}} = 2\varepsilon \right\} \geq 1-\eta$$

Avec une confiance $1-\eta$ donnée, l'optimum empirique se rapproche de l'optimum réel lorsque le nombre d'exemples augmente, et ce, d'autant plus vite que la capacité de notre système d'apprentissage est faible.

Cela peut s'exprimer de plusieurs manières, conformes au sens commun.

- Avec une confiance $1-\eta$ et pour un système de capacité h donnée, disposer de plus d'exemples garantit une meilleure précision ε.
- Une précision donnée ε sera atteinte plus sûrement par un système de capacité finie h si l'on dispose de plus d'exemples.
- Disposer de plus d'exemples permet d'utiliser un système plus fin pour obtenir une précision donnée ε avec une confiance donnée $1-\eta$.
- Avec une confiance $1-\eta$ et une précision ε donnée, le nombre d'exemples suffisant est d'autant plus fort que la capacité h de notre système d'apprentissage est forte.

Un système plus fin possède en général un optimum réel $\inf C$ meilleur. Il permet alors, *si l'on dispose de plus d'exemples*, d'obtenir un optimum empirique $C(\mathbf{w}^*_{emp})$ meilleur. En revanche, si l'on ne dispose pas d'assez d'exemples, l'optimum empirique peut être moins bon (cf. Fig 4.3).

Il y a donc un compromis à faire entre la capacité du système d'apprentissage utilisé et le nombre d'exemples nécéssaires.

4.5 Cas de la reconnaissance des formes.

Les méthodes de reconnaissance des formes identifiées en 3.4. sont ici passées au crible des théorèmes de convergence uniforme. Le coût réel appartient alors à un intervalle de confiance centré sur le coût empirique.

Ces théorèmes nous permettent de comparer les capacités de généralisation de nos trois méthodes d'apprentissage pour la reconnaissance des formes. Ils permettent par ailleurs [1] d'évaluer de même les principaux algorithmes statistiques de reconnaissance des formes.

4.5.1 Minimisation d'une approximation du risque moyen.

Dans le cas de la reconnaissance des formes, le risque moyen s'écrit, en combinant (3.51) et (3.55)

$$C^{risk}(\mathbf{w}) = \int \sum_k J(C, C_k)\ \mathbb{1}_{\Omega_k(\mathbf{X})}(\mathbf{w})\ p(\mathbf{x})\ d\mathbf{x}$$

On note $\vartheta_{kq}(\mathbf{w})$ l'événement "une forme appartenant à la classe q a été classée dans la classe k". Cet événement s'écrit donc

$$\vartheta_{kq}(\mathbf{w}) = \{\mathbf{w} \in \Omega_k(\mathbf{X})\} \cap \{C = C_q\} \tag{4.19}$$

On peut alors écrire le coût de la façon suivante

$$C^{risk}(\mathbf{w}) = \int \sum_{k,q} J(C_q, C_k)\ \mathbb{1}_{\vartheta_{kq}(\mathbf{w})}(\mathbf{x})\ p(\mathbf{x})\ d\mathbf{x}$$

qui nous permet d'appliquer le théorème (4.18), en notant h la capacité de la classe d'événements $\vartheta_{kq}(\mathbf{w})$, et $\tau = \max J(C_q, C_k) - \min J(C_q, C_k)$:

$$P\left\{ \sup_{\mathbf{w}} |\tilde{C}_L^{risk}(\mathbf{w}) - C^{risk}(\mathbf{w})| > 2\tau \sqrt{\frac{h(\ln\frac{2L}{h}+1) - \ln\frac{\eta}{9}}{L}} \right\} < \eta \tag{4.20}$$

Lorsque L croit, l'écart avec laquelle le risque empirique approche le risque moyen tend vers 0. Ce résultat est qualitativement très positif:

- Il prouve qu'il est possible de généraliser si l'on dispose d'un nombre d'exemples grand devant la capacité h de notre système.

- Il montre comment le phénomène de généralisation croît avec le nombre d'exemples, et diminue lorsqu'augmente une mesure de la capacité du système de classification.

---

1 **Devroye L.:** *Automatic Pattern Recognition: A Study of the Probability of Error* - IEEE Trans. on Pattern Anal. and Mach. Intell., vol 10, n° 4, pp 530-543, (1988)

Si l'on sait trouver le minimum global du risque empirique, la formule (4.9) nous permettrait alors de comparer, avec une probabilité $1-\eta$, le risque final $C^{risk}(\mathbf{w}_L{}^{*risk})$ et le risque moyen optimal $\inf C^{risk}$ dont notre système est capable.

$$C^{risk}(\mathbf{w}_L{}^{*risk}) < \inf C^{risk} + 4\tau \sqrt{\frac{h(\ln\frac{2L}{h}+1)-\ln\frac{\eta}{9}}{L}} \qquad (4.21)$$

En fait, nous avons vu que l'on ne sait minimiser que des approximations du risque empirique. Dans le cas de LVQ2, par exemple, cette approximation est contrôlée par un paramètre δ, qu'il est possible de diminuer progressivement. Si la densité $p(\mathbf{X})$ est assez régulière, on réduit l'écart entre le l'approximation et le risque moyen au voisinage du minimum.

De plus, comme on ne dispose que d'un temps fini pour effectuer l'optimisation, on n'atteint pas exactement le minimum du risque empirique. La formule (4.21) ne constitue donc qu'une approximation raisonnable de la situation réelle.

4.5.2 Evaluation des probabilités a posteriori.

Cette méthode détermine les paramètres $\mathbf{w}$ pour lesquels les fonctions $P_k(\mathbf{X},\mathbf{w})$ approchent le mieux les probabilités a posteriori $P(C_k|\mathbf{X})$ au sens des moindres carrés. On peut alors utiliser ces approximations pour classer les entrées $\mathbf{X}$.

Si la classe d'événements

$$S(\mathbf{w},\lambda) = \left\{ \left(\Phi_\bullet(C)-\mathbf{P}_\bullet(\mathbf{X},\mathbf{w}) \right)^2 > \lambda \right\}$$

possède une capacité finie h', on peut comparer, avec une probabilité $1-\eta$, l'erreur quadratique moyenne à l'optimum de l'erreur empirique $C^{mse}(\mathbf{w}_L{}^{*mse})$ avec l'erreur quadratique optimale $\inf C^{mse}$ dont est capable notre système d'évaluation des probabilités a posteriori.

$$C^{mse}(\mathbf{w}_L{}^{*mse}) < \inf C^{mse} + 8 \sqrt{\frac{h'(\ln\frac{2L}{h'}+1)-\ln\frac{\eta}{9}}{L}} \qquad (4.22)$$

Mais cela nous renseigne peu sur le risque moyen $C^{risk}(\mathbf{w}_L{}^{*mse})$ correspondant, lorsque l'on classifie au moyen de la formule de décision (3.66). En effet, $C^{mse}(\mathbf{w}_L{}^{*mse})$ s'écrit à une constante près

$$C^{mse}(\mathbf{w}_L{}^{*mse}) = \sum_k \int_{\mathbf{X}} (P_k(\mathbf{X},\mathbf{w}_L{}^{*mse})-P(C_k|\mathbf{X}))^2 \; p(\mathbf{X})d\mathbf{X} \qquad (4.23)$$

Un majorant de cette expression ne permet pas en effet de déterminer l'écart maximum de son intégrande:

$$\sup_{\mathbf{X}} \left(P_k(\mathbf{X},\mathbf{w}_L{}^{*mse})-P(C_k|\mathbf{X})\right)^2$$

Nous sommes ici pénalisés par notre approche indirecte du problème: Passer d'une évaluation des probabilités a posteriori à une minimisation du risque moyen pose à nouveau un problème de grandes déviations. Comme toujours, on peut utiliser des hypothèses fortes sur la régularité de la densité $p(\mathbf{X})$. Il suffit par exemple de supposer

$$\left| \nabla_{\mathbf{X}} \left((P_k(\mathbf{X},\mathbf{w}_L{}^{*mse})-P(C_k|\mathbf{X}))^2 \, p(\mathbf{X}) \right) \right| < M \qquad (4.24)$$

Une hypothèse comme (4.24) est assez raisonnable, lorsque l'on considère des entrées $\mathbf{X}$ soumises à un bruit gaussien. Les probabilités $P(C_k|\mathbf{X})$, sont alors convolées avec la densité $n(\xi)$ d'une gaussienne, ce qui a un effet régularisant.

Un petit écart δ sur $(P_k(\mathbf{X},\mathbf{w}_L{}^{*mse})-P(C_k|\mathbf{X}))^2$ au point $\mathbf{X}_0$ implique alors que cette valeur reste supérieure à $\delta/2$ sur une boule centrée sur $\mathbf{X}_0$, de rayon $\delta/2M$. On peut alors écrire

$$\sup_{\mathbf{X}} \left(P_k(\mathbf{X},\mathbf{w}_L{}^{*mse})-P(C_k|\mathbf{X})\right)^2 < \frac{\alpha M}{p(\mathbf{X})} \sqrt[n]{C^{mse}(\mathbf{w}_L{}^{*mse})} \qquad (4.25)$$

où n est la dimension du vecteur $\mathbf{X}$, et α un coefficient de proportionnalité. En combinant (3.66), (4.22) et (4.25), on peut obtenir un majorant complexe de l'écart entre le risque à l'optimum empirique $\mathbf{w}_L{}^{*mse}$ et le risque minimal dont est capable le système. Ce majorant diminue avec le coût optimal $\inf C^{mse}$, et avec la probabilité de classification ambiguë, c'est à dire la probabilité que le majorant (4.25) ne soit pas assez fin pour permettre une décision certaine.

Quoi qu'il en soit, la formule (4.20) reste néanmoins valable. On a toujours, avec une probabilité $1-\eta$,

$$C^{risk}(\mathbf{w}_L{}^{*mse}) < \tilde{C}^{risk}(\mathbf{w}_L{}^{*mse}) + 2\tau \sqrt{\frac{h(\ln\frac{2L}{h}+1)-\ln\frac{\eta}{9}}{L}} \qquad (4.26)$$

où h est maintenant la capacité de la classe d'événements $\vartheta_{kq}(\mathbf{w})$ exprimée par (4.19). La formule (4.20) fournit une majoration du risque moyen pour tout $\mathbf{w}$, et en particulier pour $\mathbf{w}_L{}^{*mse}$.

En conclusion, (4.26) nous apprend que les risques moyens $C^{risk}(\mathbf{w}_L{}^{*mse})$ et empiriques $\tilde{C}^{risk}(\mathbf{w}_L{}^{*mse})$ sont voisins.

Cependant, *on ne peut pas affirmer que ceux ci sont proches de* $\inf C^{risk}$. La figure 2.9 montre d'ailleurs qu'une telle affirmation serait fausse dans le cas de l'adaline.

S'écarter de la minimisation du risque moyen présente des inconvénients qui n'apparaissent que lorsque l'on tient compte des limites sur le nombre de nos exemples.

4.5.3 Evaluation des vraisemblances.

Cette méthode détermine les paramètres $\mathbf{w}$ pour lesquels les fonctions $p_k(\mathbf{X},\mathbf{w})$ approchent les vraisemblances conditionnelles $p(\mathbf{X}|C_k)$ au moyen du principe du maximum de vraisemblance.

Si la *vraisemblance conditionnelle est exactement de la forme* $p_k(\mathbf{X},\mathbf{w}^*)$, la théorie du maximum de vraisemblance [1] affirme, pour certaines classes de fonctions p_k, dont les gaussiennes et les mixtures de gaussiennes, que les fonctions $p_k(\mathbf{X},\mathbf{w}_L{}^{*mle})$ maximisant la vraisemblance d'un échantillon indépendant $\{\mathbf{x}_i\}$ de taille L sont des estimateurs

- *asymptotiquement non biaisés,*

$$E(\mathbf{w}_L{}^{*mle}) \underset{L\to\infty}{\to} \mathbf{w}^*$$

- *consistants,*

$$\forall \varepsilon>0,\ P\left\{ |\mathbf{w}_L{}^{*mle}-\mathbf{w}^*| > \varepsilon \right\} \underset{L\to\infty}{\to} 0$$

- *asymptotiquement efficaces,*

$$Det\left(Cov(\mathbf{w}_L{}^{*mle}-\mathbf{w}^*)\right) I_\Phi \underset{L\to\infty}{\to} 1$$

où "l'information de Fisher" I_Φ *vaut* $Det\left(-L\int \nabla^2\left(\log p_k(\mathbf{x},\mathbf{w}^*)\right) p(\mathbf{x})d\mathbf{x}\right)$.

La consistance nous assure que lorsque L tend vers l'infini, $\mathbf{w}_L{}^{*mle}$ tend en probabilité vers $\mathbf{w}^*$. L'efficacité asymptotique mesure la vitesse de cette convergence.

Si on n'admet plus l'hypothèse forte que la vraisemblance conditionnelle appartient exactement à la famille $p_k(\mathbf{X},\mathbf{w})$, on retombe dans les problèmes soulignés dans la section 4.2. Seule alors la formule (4.20) reste valable, avec une probabilité $1-\eta$,

---

1 **Wilks S. S.**: *Mathematical Statistics*, New-York: Wiley (1962)

$$C^{risk}(\mathbf{w}_L{}^{*mle}) < \tilde{C}^{risk}(\mathbf{w}_L{}^{*mle}) + 2\tau \sqrt{\frac{h(\ln\frac{2L}{h}+1)-\ln\frac{\eta}{9}}{L}} \qquad (4.27)$$

où h est toujours la capacité de la classe d'événements $\vartheta_{kq}(\mathbf{w})$ exprimée en (4.19). Cela fournit encore une fois une majoration du risque moyen pour tout $\mathbf{w}$, et en particulier pour $\mathbf{w}_L{}^{*mle}$.

4.5.4 Ordres de grandeur

On pourrait s'attendre à ce que les résultats quantitatifs fournis par ces théorèmes de convergence uniforme soient inutilisables. En fait, cela pourrait être bien pire.

Prenons le cas de la classification de formes de dimension n en deux classes, à l'aide d'un séparateur linéaire, par exemple un perceptron. On l'a vu, la capacité d'un tel classificateur est $n+1$. On entraîne le perceptron avec L exemples, et on constate une proportion $\tilde{e}$ de mauvaises classifications.

La probabilité de mauvaise classification (taux d'erreur) s'écrit

$$e = 1 - \int \mathbb{1}_{\vartheta_{11}(\mathbf{x})\cup\vartheta_{22}(\mathbf{x})}\, p(\mathbf{x})d\mathbf{x}$$

D'après (4.20), avec une probabilité $1-\eta$, l'écart entre e et $\tilde{e}$ est inférieur à

$$|e - \tilde{e}| < 2\sqrt{\frac{n(\ln\frac{2L}{n}+1)-\ln\frac{\eta}{9}}{L}}$$

On peut de même déterminer un nombre d'exemples suffisant pour que cet écart soit inférieur à un seuil fixé.

Prenons maintenant des ordres de grandeur concrets: $\tilde{e} = 5\%$, $n=9$ $(h=10)$, $\eta=0.1$. Le tableau ci après contient les majorants de l'écart et du taux d'erreur pour quelques valeurs de L.

L	Majorant de $\lvert \tilde{e} - e \rvert$	Majorant de e
1 000	0,52	0,57
10 000	0,20	0,25
50 000	0,10	0,15
100 000	0,07	0,12
500 000	0,03	0,08

Fig 4.2 - Quelques valeurs numériques pour $\tilde{e} = 5\%$, $n=10$, $\eta=0.1$

Or, il est fréquent, pour *des problèmes concrets*, d'obtenir des écarts inférieurs à 0.1 avec une centaine d'exemples seulement. Notre condition suffisante est en effet *indépendante de la distribution* $p(\mathbf{x})$. Dans le cas de problèmes concrets, cette distribution est rarement quelconque.

Quoique surévalués, ces ordres de grandeur restent à la portée des calculateurs modernes. Cela représente une considérable avancée quand on les compare aux hypothèses asymptotiques pour lesquelles L est arbitrairement grand.

4.6 Procédures d'apprentissage.

L'utilisation d'un ensemble de test permet de comparer plusieurs systèmes et de sélectionner le meilleur. Elle permet également de définir un critère d'arrêt de l'algorithme d'apprentissage.

Les théorèmes de convergence uniforme permettent une fois de plus d'obtenir un intervalle de confiance pour le coût réel lorsque l'on applique de telles procédures, manuellement ou automatiquement.

En pratique, on dispose d'un ensemble d'exemples de taille L toujours trop faible, et on souhaite obtenir le meilleur résultat possible. On procède heuristiquement en deux étapes:

- La première étape consiste à identifier plusieurs systèmes et leurs algorithmes d'apprentissage.
- La seconde étape, dite de *sélection*, consiste à déterminer le meilleur, c'est à dire celui qui réalise le mieux l'optimisation de notre critère.

Cette technique est celle que connaissent tous les praticiens des algorithmes statistiques ou connexionnistes. On compare des régressions polynomiales de degrés variés, ou bien des perceptrons multi-couches avec plus ou moins d'unités cachées.

Cela se produit aussi de façon moins explicite: Lorsque l'on utilise un algorithme itératif d'apprentissage, on peut comparer le système à différents moments de l'apprentissage, et sélectionner le meilleur. Cela permet de définir *un critère d'arrêt*.

4.6.1 Mesure de la qualité d'un système.

Le premier problème posé par cet heuristique est la détermination du meilleur système, c'est à dire de la mesure de la qualité d'un système. Cela passe par la détermination d'un critère de qualité, que nous supposerons de la forme (3.2)

$$C(\mathbf{w}^*) = \int J(\mathbf{x},\mathbf{w}^*)\ p(\mathbf{x})d\mathbf{x} \tag{4.28}$$

Idéalement, le critère de qualité est le critère de coût qu'optimise notre algorithme d'apprentissage. L'exemple de la reconnaissance des formes montre cependant que ce n'est pas toujours possible: On ne sait pas minimiser le risque moyen, on procède indirectement, soit sur une approximation continue du risque moyen, soit sur des critères statistiques variés.

Plusieurs méthodes permettent de mesurer C, sans connaître la densité $p(\mathbf{x})$. Elles consistent toutes à mesurer une qualité empirique

$$\tilde{C}(\mathbf{w}^*) = \int J(\mathbf{x},\mathbf{w}^*)\ \tilde{p}(\mathbf{x})d\mathbf{x} = \sum_{k=1}^{L} C(\mathbf{x},\mathbf{w}^*) \tag{4.29}$$

sur des échantillons $\{\mathbf{x}_k\}$ bien choisis. Deux sont particulièrement générales:

i Utilisation des exemples d'apprentissage.

La plus simple consiste à mesurer $\tilde{C}(\mathbf{w}^*)$ à l'aide des exemples utilisés pour l'apprentissage. Le théorème (4.18) permet alors de déterminer un intervalle de confiance pour la qualité réelle.

$$\tilde{C}(\mathbf{w}^*) - 2\tau\sqrt{\frac{h(\ln\frac{2L}{h}+1)-\ln\frac{\eta}{9}}{L}} \ \le\ C(\mathbf{w}^*)\ \le \tilde{C}(\mathbf{w}^*) + 2\tau\sqrt{\frac{h(\ln\frac{2L}{h}+1)-\ln\frac{\eta}{9}}{L}} \tag{4.30}$$

où τ est un majorant de $(\sup J(\mathbf{x},\mathbf{w}^*) - \inf J(\mathbf{x},\mathbf{w}^*))$.

Cependant, les ordres de grandeur obtenus dans la section 4.5.4 montrent que cette *inégalité est trop grossière* pour effectuer une mesure, en particulier lorsque h est grand.

i Utilisation d'exemples de test.

Si l'on dispose d'un échantillon aléatoire $\{\mathbf{x}_k\}$, de taille L, *différent des exemples d'apprentissage*, on peut évaluer $\tilde{C}(\mathbf{w}^*)$ sur cet ensemble. On peut alors utiliser l'inégalité de Hoeffding (4.8). On a alors, avec une probabilité $1-\eta$,

$$\tilde{C}(\mathbf{w}^*) - \tau\sqrt{\frac{-\ln(\eta/2)}{2L}} \le C(\mathbf{w}^*) \le \tilde{C}(\mathbf{w}^*) + \tau\sqrt{\frac{-\ln(\eta/2)}{2L}} \qquad (4.31)$$

où τ est un majorant de $(\sup J(\mathbf{x},\mathbf{w}^*) - \inf J(\mathbf{x},\mathbf{w}^*))$. On obtient ainsi un intervalle de confiance pour la qualité réelle $C(\mathbf{w}^*)$ du système.

Environ 15000 exemples sont nécessaires, par exemple, pour mesurer un taux d'erreur à ±1%, avec une probabilité de 0.9. Il en suffit de 600 pour mesurer un taux d'erreur à ±5%, dans les mêmes conditions.

iii Comparaison

L'inégalité (4.30) majore l'écart entre la moyenne empirique et l'espérance, uniformément pour toutes les valeurs de $\mathbf{w}$. En particulier, si l'échantillon et $\mathbf{w}$ ne sont pas indépendants, c'est à dire si les exemples ont été utilisés pour l'apprentissage, la majoration reste valable. Elle permet donc d'obtenir un résultat, sans exemples supplémentaires, au prix d'une faible précision.

L'inégalité (4.31) majore l'écart entre la moyenne empirique et l'espérance, pour une seule valeur de $\mathbf{w}$, indépendante de l'échantillon. C'est à dire que les exemples n'ont pas été utilisés pour l'apprentissage. Elle permet, au prix d'exemples supplémentaires, d'obtenir une bonne précision.

4.6.2 Minimisation structurelle.

L'encadrement (4.30) est d'autant plus grossier que l'espace décrit par les paramètres $\mathbf{w}$ possède une capacité grande. Or, rien ne sert d'avoir un meilleur optimum empirique, si cela n'améliore pas le coût réel.

4.6.2.1 Structures sur l'espace des solutions.

L'idée de la minimisation structurelle [1] repose sur la supposition suivante. On suppose l'espace Ω des paramètres $\mathbf{w}$ doté, a priori, d'une *structure*, c'est à dire une suite (Ω_i) de N sous ensembles imbriqués

$$\Omega=\Omega_N \supset \Omega_{N-1} \supset \ldots \supset \Omega_2 \supset \Omega_1 \tag{4.32}$$

dont les capacités h_i+1 sont ordonnées

$$h_N \geq h_{N-1} \geq \ldots \geq h_2 \geq h_1 \tag{4.33}$$

On note $\mathbf{w}^*_i$ le minimum, à l'intérieur de l'espace Ω_i, d'un coût empirique défini sur L exemples. Alors, de façon évidente,

$$\tilde{C}(\mathbf{w}^*_N) \leq \tilde{C}(\mathbf{w}^*_{N-1}) \leq \ldots \leq \tilde{C}(\mathbf{w}^*_2) \leq \tilde{C}(\mathbf{w}^*_1) \tag{4.34}$$

Le coût réel ne suit pas en général cette règle. L'encadrement (4.20) du coût réel devient de plus en plus grossier lorsque l'on considère un espace Ω_i plus grand.

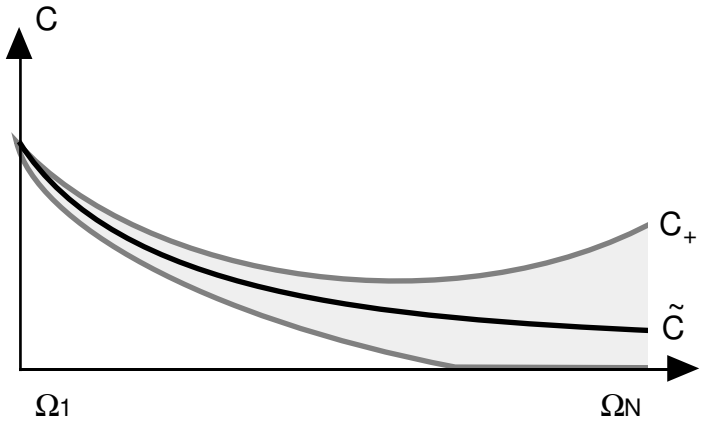


Fig 4.3- Evolutions du coût empirique, et de l'intervalle garanti du coût réel.

On peut alors définir le coût garanti $C_+(\mathbf{w}^*_N)$ avec une probabilité $1-\eta$ par

$$C(\mathbf{w}^*_i) \leq C_+(\mathbf{w}^*_i) = \tilde{C}(\mathbf{w}^*_i) + 2\tau \sqrt{\frac{h_i(\ln\frac{2L}{h_i}+1)-\ln\frac{\eta}{9}}{L}} \tag{4.35}$$

On obtient donc la situation résumée dans la figure 4.3.

---

1 **Vapnik V.N., Chervonenkis A. Ya:** *The Theory of Pattern Recognition* - Nauka, Moscou (1974)

4.6.2.2 Exemple de structures.

i Régressions polynomiales.

Dans le cas de régressions polynomiales, on cherche à minimiser un coût de la forme

$$C = \int (Y - Q[\mathbf{X}])^2 \, p\{(\mathbf{X}, Y)\} \, d\mathbf{X} dY \tag{4.36}$$

où $Q[\mathbf{X}]$ est un polynôme de degré maximum fixé. On peut alors décomposer ce polynôme sur une base $(\varphi_k[\mathbf{X}])$ de ces polynômes.

La capacité des sous-espaces vectoriels Ω_i engendrés par le i premiers éléments de cette base vaut alors $i+1$. Les Ω_i vérifient donc les propriétés (4.31) et (4.32).

La définition des Ω_i dépend en fait de la façon dont on a ordonné la base $(\varphi_k[\mathbf{X}])$. Cela définit en fait plusieurs structures sur notre espace de polynômes. On les ordonne en général selon leur degrés, mais la forme du problème peut en suggérer un autre ordre.

ii Perceptrons multi-couches.

Dans le cas d'un réseau multi-couches d'unités à seuil composé de N unités, et W, on montre [1] que la capacité est majorée par $2W.\ln(eN)$ où e est la base du logarithme népérien.

Cela suggère de structurer l'espace des perceptrons multi-couches en fonction du nombre de paramètres qu'il contiennent. Cette idée est confirmée par le résultat de convergence uniforme [2] qui montre, pour les perceptrons multi-couches, que la propriété

$$P\left\{ \sup_{\mathbf{w}} \left| \frac{C(\mathbf{w}) - \tilde{C}_L(\mathbf{w})}{\nu + C(\mathbf{w}) + \tilde{C}_L(\mathbf{w})} \right| > \varepsilon \right\} < \eta$$

est vraie pour un nombre d'exemples supérieur à

$$O\left(\frac{1}{\varepsilon^2 \nu} \left(W \left(k.\ln(\beta s u) - \ln(\varepsilon \nu) \right) - \ln(\eta) \right) \right) \tag{4.37}$$

---

1 **Baum E., Haussler D.:** *What size net gives valid generalization* - Neural Computation, vol 1, n° 1, pp 151-160, (1989)

2 **Haussler D.:** *Generalizing the PAC Model for Neural Net and other Learning Application* - Technical Report UCSC-CRL-89-30, University of Calif., Computer Research Laboratory, Santa Cruz, CA (1989)

où W est le nombre de poids, k est le nombre de couches, u le nombre maximum d'unités par couches, s la pente maximale des sigmoïdes, et β un *majorant des poids*. Ces proportionnalités sont assez bien vérifiées par l'expérience [1].

iii Algorithmes itératifs d'apprentissage.

Lorsque l'on utilise un algorithme itératif, les poids peuvent d'autant plus s'éloigner de leur valeurs initiales que l'on laisse l'algorithme tourner longtemps.

Or la relation (4.37) montre une dépendance avec la valeur maximale des poids. Cela s'interprète de la façon suivante:

- Lorsque les poids sont faibles, on travaille dans une portion linéaire de la sigmoïde. Le réseau fonctionne en gros comme une régression linéaire, de faible capacité.

- Lorsque les poids augmentent, on atteint les parties non linéaires de la sigmoïde. On retrouve alors une capacité plus importante, qui permet d'obtenir un coût empirique bien inférieur, mais au prix d'une plus grande incertitude sur le coût réel.

On peut envisager une structure sur les solutions, fondée sur la valeur maximale des poids. *Cela montre qu'il n'est pas avisé de rechercher systématiquement l'optimum du critère empirique.*

Le système possède à chaque instant une performance instantanée, et sélectionner le meilleur revient à définir *un critère d'arrêt sur l'apprentissage*. Ce critère d'arrêt ne dépend pas de la qualité avec laquelle on atteint l'optimum du critère empirique, mais de la qualité de la généralisation.

4.6.2.3 Méthodes de sélection.

La sélection de la meilleure solution $\mathbf{w}^*_i$ est évidemment fondée sur les méthodes de mesure du coût réel $C(\mathbf{w}^*_i)$, c'est à dire les encadrements (4.30) et (4.31).

i Minimisation du coût garanti

Une première méthode s'appuie sur (4.30). La borne supérieure de cet encadrement représente un coût garanti (4.35). On peut donc sélectionner la solution qui offre un coût garanti minimal.

---

1 **Haffner P., Waibel A., Shikano K.:** *Fast Back-Propagation Methods for Neural Networks in Speech*. Procs of European Conf. on Speech Communication and Technology, Paris, Vol 2, pp 553-556 (1989)

Cette méthode nécessite la connaissance de la capacité h associée à chaque espace de solutions Ω_i , ce qui est en fait assez rare. La faible précision de l'encadrement (4.30) n'est pas en général un obstacle insurmontable, le coût réel suivant assez bien les variations du coût garanti. (Vapnik dixit)

ii Validation croisée sur un ensemble de test.

Une autre solution s'appuie sur (4.31). Elle consiste à évaluer le coût à l'aide d'un échantillon de test, jamais utilisé pour l'apprentissage. On sélectionne donc la solution dont le coût, évalué selon cette méthode, est le plus faible. C'est la solution la plus fréquemment employée. Son utilisation comme critère d'arrêt est également très fréquente, et a été bien décrite dans [1].

Un problème se pose alors: Comment mesurer la qualité de notre système final? Faut-il disposer d'un troisième échantillon de test ?

C'est à nouveau un problème de convergence uniforme: On dispose de N fonctions $J(\mathbf{x},\mathbf{w}^*_n)$, et on cherche une majoration de l'écart de mesure. L'encadrement (4.30) donne alors un intervalle de confiance assez grossier.

Cependant, comme le nombre de fonctions $J(\mathbf{x},\mathbf{w}^*_n)$ est fini, il est possible de remonter à l'expression (4.17) et remarquer que $m_S(L)$ est évidemment majoré par N. On obtient alors

$$P\left\{ \sup_n |\tilde{C}_L(\mathbf{w}^*_n) - C(\mathbf{w}^*_n)| > \tau.\varepsilon \right\} < \eta_{L,\varepsilon} = 6\, N\, e^{-\varepsilon^2 L/4} \qquad (4.38)$$

La convergence uniforme est en effet une propriété naturelle lorsque le nombre de solutions possibles $\mathbf{w}^*_n$ est fini. On peut trouver simplement une bien meilleure borne en remontant à l'inégalité de Hoeffding (4.8). On peut alors majorer la probabilité que l'écart associé à l'une des N solutions dépasse un seuil $\tau.\varepsilon$, ce qui donne

$$\forall \varepsilon>0,\quad P\left\{ \sup_n |\tilde{C}_L(\mathbf{w}^*_n) - C(\mathbf{w}^*_n)| > \tau.\varepsilon \right\} \le 2Ne^{-2\varepsilon^2 L} \qquad (4.39)$$

La parenté entre les formules (4.38) issue de théorèmes de Vapnik, et (4.39) issue de l'inégalité de Hoeffding n'est évidement pas fortuite. On déduit de (4.39) que l'encadrement suivant est vrai avec une probabilité $1-\eta$,

$$\tilde{C}(\mathbf{w}^*) - \tau\sqrt{\frac{\ln(2N) - \ln(\eta)}{2L}} \le C(\mathbf{w}^*) \le \tilde{C}(\mathbf{w}^*) + \tau\sqrt{\frac{\ln(2N) - \ln(\eta)}{2L}} \qquad (4.40)$$

---

1 **Morgan N., Bourlard H.:** *Generalization and Parameter Estimation in Feedforward Nets: Some Experiments* - Advances in Neural Information Processing Systems II, Morgan Kaufmann (1990)

Cet encadrement, où $\mathbf{w}^*$ représente la solution choisie, offre une précision à peine inférieure à celle offerte par (4.31), lorsque N est assez faible. Cela rend légitime la *procédure d'essais et comparaisons* fréquemment utilisée pour rechercher un système acceptable.

5 Aspects techniques.

Les deux chapitre précédents ont décrit et formalisé mathématiquement une collection d'algorithmes fondés sur un processus d'optimisation stochastique. Cette étude permet de connaître leur fonctionnement, d'étudier leur convergence, d'évaluer les causes de mauvais fonctionnement, et d'en majorer la probabilité. De plus, des critères de comparaison, de validation croisée et d'arrêt ont été identifiés ou ébauchés.

Ces caractéristiques ont cependant été définies indépendamment de certains facteurs, dont:

- Le codage des conditions extérieures $\mathbf{X}$,
- La forme des fonctions $J(\mathbf{X},\mathbf{W})$,
- Le choix des paramètres $\mathbf{W}$ initiaux,
- La façon de faire évoluer les gains ε_t,

Or, ceux ci ont une influence considérable sur les aspects technologiques de l'apprentissage: la performance finale et le coût, en temps ou en matériel. Cela consiste en fait à définir une technologie: un ensemble de matériels, programmes, et procédures permettant d'approcher à moindre coût les espoirs suscités par la théorie.

Cette intrusion des aspects pratiques induit un changement de méthodologie. La réduction constatée de temps ou de moyens nécessaires importe plus que la rigueur ou la beauté formelle du raisonnement. On préfère disposer d'un système meilleur, sans avoir la certitude qu'il est optimal.

Cela entraîne deux conséquences:

- La première est qu'une technologie s'apparente à un livre de cuisine. Telle épice améliore le plat, pour des raisons connues, supposées, ou inconnues.

- La seconde est qu'une technologie n'est pas unique. Par hasard ou intuition, deux chercheurs ou deux équipes construiront deux technologies différentes. Chacune constitue un tout. Adopter dans l'une les recettes de l'autre n'a de sens que si l'on constate une amélioration, ce que rien ne prouve a priori.

Les sections 5.1 et 5.2 décrivent les quelques idées, souvent imprécises hélas, qui fondent la technologie qui sert de support à cette thèse. Les principes les plus anciens en ont été établis par Yann Le Cun a partir de 1986 dans le programme HLM. Nous les avons ensuite affinés et systématisés dans le programme SN. Ces programmes, brièvement décrits dans la section 5.3, ont joué un rôle important pour l'utilisation et la diffusion de cette technologie.

Il s'agissait, à l'origine, d'obtenir une variante efficace de l'algorithme de rétro-propagation. Les remarques de ce chapitre s'appliquent cependant à tout algorithme de gradient, mais prennent des importances relatives différentes. Certains algorithmes, comme k-means (cf §3.2.2), fonctionnent aisément, d'autres, comme la rétro-propagation du gradient, ont demandé beaucoup de tâtonnements.

5.1 Algorithmes stochastiques.

Le choix d'utiliser systématiquement des algorithmes de gradient stochastiques est fondé sur les arguments suivants, déjà développés dans ce qui précède:

- Les algorithmes stochastiques sont en général plus rapides que les algorithmes équivalents opérant sur l'ensemble des exemples.

  Ces exemples sont souvent très redondants. En reconnaissance de mots isolés, par exemple, il y a redondance entre les dizaines d'exemples que l'on utilise pour chaque mot; il y a également redondance entre exemples de mots différents qui partagent des syllabes, ou des phonèmes.

  Un algorithme stochastique ne nécessite que le passage d'une faible partie des exemples pour faire en moyenne ce qui exige le passage de tous les exemples pour un algorithme non stochastique.

  On constate ainsi une réduction du temps de convergence d'ordre de grandeur comparable au nombre d'exemples par classe . Cela représente un facteur 10 à 200 en pratique!.

- Les algorithmes stochastiques, comme on l'a développé dans la section 3.3.2.2, permettent d'atteindre avec une probabilité plus grande les bons optima du coût empirique.

En particulier, un minimum local de la fonction de coût $C(\mathbf{w})$ n'est un point stable de l'algorithme que si ce point est un minimum local pour toute fonction $J(\mathbf{x},\mathbf{w})$. On constate un phénomène comparable à celui du recuit simulé: L'algorithme a tendance à converger vers de bons minima.

Comme tous les algorithmes de gradient, une utilisation trop simpliste d'une descente de gradient stochastique peut facilement entraîner des temps d'apprentissage démesurés. En revanche, un algorithme stochastique bien conditionné rivalise en rapidité avec un algorithme exact: Il est souvent plus rapide (cf. §2.3.2), lorsque le nombre d'exemples est grand, d'utiliser la méthode de Widrow Hoff (2.21) que la méthode de Greville pour calculer une pseudo-inverse.

Il faut également noter que certaines méthodes efficaces pour des algorithmes de gradient déterministes, comme l'introduction de termes α d'inertie, (*momentum*)

$$\mathbf{w}_{t+1} - \mathbf{w}_t + (1-\alpha)(-\varepsilon\nabla C) + \alpha(\mathbf{w}_t-\mathbf{w}_{t-1}) \qquad (5.1)$$

ou δ de décroissance exponentielle (*decay*)

$$\mathbf{w}_{t+1} = (1- \varepsilon\delta)\mathbf{w}_t - \varepsilon\nabla C \qquad (5.2)$$

sont inefficaces, voire nuisibles, dans le cas du gradient stochastique.

5.2 Aspects numériques de l'optimisation.

Le problème de conditionnement constitue la principale cause de lenteur des algorithmes de gradient. On expose ici plusieurs façons d'améliorer la situation, soit au moyen de règles de bon sens, soit en évaluant de façon approchée la matrice hessienne.

Il est malheureusement très difficile d'étudier les aspects numériques de l'optimisation stochastique (3.6). Ceux-ci dépendent formellement de la distribution $p(\mathbf{x})$. De plus, le formalisme de quasi-martingales utilisé dans la section 3.3 est mal adapté à l'étude des aspects numériques de la convergence.

On se résout alors à étudier certains aspects de l'algorithme déterministe (3.5). L'expérience montre que les informations que cela apporte restent utilisables et profitables même dans le cas d'un algorithme stochastique.

5.2.1 Bon et mauvais conditionnement.

5.2.1.1 Définition.

La faiblesse de l'algorithme de descente de gradient réside dans *les problèmes de conditionnement*. La plus grande pente indique une trajectoire possible pour atteindre un minimum, mais cette trajectoire n'est pas nécessairement la plus courte.

La figure 5.1 montre deux exemples:

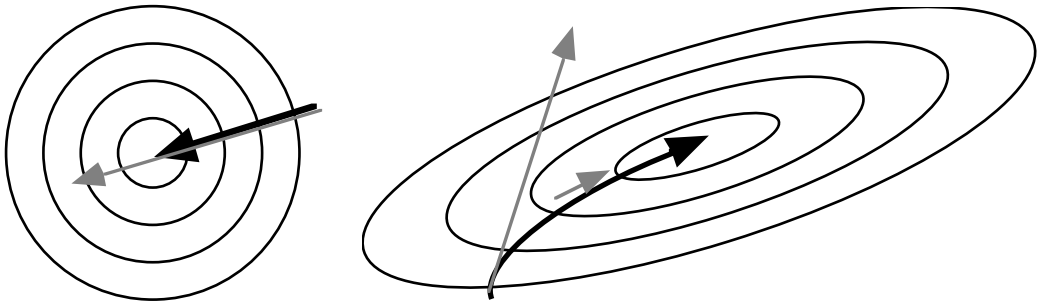


Fig 5.1- Exemples de bon et mauvais conditionnements.

- A gauche, les courbes de niveau de la fonction de coût C sont concentriques. La trajectoire de plus grande pente est une ligne droite dirigée vers le minimum, les gradients sont importants, la convergence sera rapide. On dit que le *problème d'optimisation est bien conditionné*.

- A droite, les courbes de niveau sont étirées. La trajectoire tombe rapidement au fond du ravin situé le long du grand axe. Or les gradients y sont faibles, et ne nous emmènent que lentement vers le minimum. On dit alors que le *problème d'optimisation est mal conditionné*.

Ce phénomène peut être modélisé à l'aide d'une approximation quadratique. Considérons un système de coordonnées centré sur le minimum $\mathbf{w}^*$, dans lequel le hessien $\mathbf{H}$ est diagonal. Un développement limité au voisinage du minimum donne

$$C(\mathbf{w}) = C^{opt} + \mathbf{w}^t \mathbf{H} \mathbf{w} + o(|\mathbf{w}|^2) \qquad (5.3)$$

Autour d'un minimum, le hessien $\mathbf{H}$ est défini positif. Toutes ses valeurs propres λ_k sont donc positives. Dans le cas d'un gradient déterministe, on a

$$\mathbf{w}_{t+1} \approx \mathbf{w}_t - \varepsilon \mathbf{H} \mathbf{w}_t = (\mathbf{I} - \varepsilon \mathbf{H}) \mathbf{w}_t \qquad (5.4)$$

On suppose le gain ε constant. On peut alors identifier trois cas, selon la valeur du gain comparée aux valeurs propres λ_i du hessien $\mathbf{H}$.

- Si $\varepsilon < 1/\max_k\{\lambda_k\}$, la matrice $\mathbf{I} - \varepsilon \mathbf{H}$ est positive, de norme inférieure à 1. Les poids $\mathbf{w}$ converge donc vers $\mathbf{w}^* = \mathbf{0}$.

- Si $1/\max_k\{\lambda_k\} < \varepsilon < 2/\max_k\{\lambda_k\}$, la norme de $\mathbf{I}-\varepsilon\mathbf{H}$ est encore inférieure à 1. Les poids $\mathbf{w}$ converge donc encore vers $\mathbf{w}^*$, mais en oscillant autour des axes pour lesquels le hessien possède une valeur propre $\lambda_k > 1/\varepsilon$.

- Si $2/\max_k\{\lambda_k\} < \varepsilon$, la valeur absolue des coefficients de $\mathbf{w}$ pour lequel le hessien possède une valeur propre $\lambda_k > 2/\varepsilon$ augmentent d'une proportion $\lambda_k\varepsilon - 1$, supérieure à 1. S'ils sont non nuls, l'algorithme diverge: c'est une situation instable.

On voit donc que le gain ε est limité par la valeur de la plus grande valeur propre, c'est à dire par la courbure transversale d'un ravin. De plus, la norme de $\mathbf{w}$ est dominée par sa composante qui converge le plus lentement, celle associée à la valeur propre non nulle la plus faible du hessien, c'est à dire par la courbure longitudinale du ravin. *Plus l'écart entre les valeurs propres extrêmes du hessien est important, plus le ravin est encaissé, plus la convergence est lente*, même en sélectionnant un gain ε optimal.

Comme le montrent les deux exemples suivants, la qualité du conditionnement est extrêmement variable d'un algorithme à l'autre.

5.2.1.2 Exemple 1: k-means.

L'algorithme k-means, (cf §3.2.2) est un algorithme de minimisation de l'erreur de quantification. Cette erreur s'écrit

$$C = \int \min_{k=1}^{K} (\mathbf{x}-\mathbf{w}_k)^2 \, p(\mathbf{x}) d\mathbf{x} \qquad (5.5)$$

En notant $\Omega(k)$ la partition du diagramme de Voronoï associé au point $\mathbf{w}_k$,son gradient s'écrit

$$\nabla_{\mathbf{w}_k} C = -2 \int \mathbb{1}_{\Omega_k}(\mathbf{x}) \, (\mathbf{x}-\mathbf{w}_{k(\mathbf{x})}) \, p(\mathbf{x}) d\mathbf{x} = -2 \int_{\Omega_k} (\mathbf{x}-\mathbf{w}_{k(\mathbf{x})}) \, p(\mathbf{x}) d\mathbf{x} \qquad (5.6)$$

L'intégrande de ce gradient est malheureusement discontinu sur un ensemble de mesure nulle. Calculer le hessien requiert donc de connaître les variations des frontières de la partition de Voronoï, ce qui est un problème difficile.

Supposons, pour simplifier, que la distribution de $\mathbf{x}$ est discrète, c'est à dire qu'il y a un nombre fini d'exemples $\mathbf{x}_k$. La probabilité qu'un des $\mathbf{x}$ se trouve sur une frontière de la partition de Voronoï est alors nulle; les variations de Ω_k n'ont plus d'influence sur C. On a alors:

$$\nabla^2_{\mathbf{w}_k,\mathbf{w}_q} C = \int 2\mathbf{I} \, \mathbb{1}_{\Omega_k}(\mathbf{x}) \, \mathbb{1}_{\Omega_q}(\mathbf{x}) \, p(\mathbf{x}) d\mathbf{x} = 2\mathbf{I} \, P\{\mathbf{x} \in \Omega_k \ \& \ k=q\} \qquad (5.7)$$

où $\mathbf{I}$ est la matrice identité. Le hessien est donc diagonal. Les coordonnées de chaque point $\mathbf{w}_k$ constituent un espace propre de valeur propre $2P\{\mathbf{x}\in\Omega_k\}$. Cela signifie que la convergence de chaque point est bien conditionnée. De plus, si les points initiaux ont été bien choisi, les probabilités $P\{\mathbf{x}\in\Omega_k\}$ sont très voisines.

On peut encore améliorer le conditionnement en décidant d'utiliser pour chaque point $\mathbf{w}_k$ un gain ε_k séparé, inversement proportionnel à une estimation empirique des probabilités $P\{\mathbf{x}\in\Omega_k\}$. En pratique, cela n'est même pas nécessaire…

5.2.1.3 Exemple 2: Un perceptron multi-couches.

A l'opposé, les perceptrons multi-couches sont des exemples remarquables de systèmes mal conditionnés.

Considérons, par exemple, un perceptron multi-couches contenant une entrée, une unité cachée, une sortie. On l'entraîne à reproduire en sortie son entrée. La fonction C possède alors la forme

$$C(\mathbf{w}) = \int \Big(x - f(w_2\, f(w_1 x))\Big)^2\; p(x)\,dx \qquad (5.8)$$

où f représente la sigmoïde, et $p(x)$ la distribution des entrées, que l'on supposera localisée au voisinage de 1.

Il est facile d'avoir une idée de la forme de cette fonction de coût si f est la fonction identité. On a alors

$$C(\mathbf{w}) = (1 - w_2\, w_1)^2 \int x^2 p(x)\,dx \qquad (5.9)$$

Le coût est alors identiquement nul sur l'hyperbole $w_1 w_2 = 1$. Cela signifie que les deux branches de l'hyperbole constituent deux ravins qui s'étirent le long des axes, et sont d'autant plus encaissés que l'on s'éloigne de l'origine.

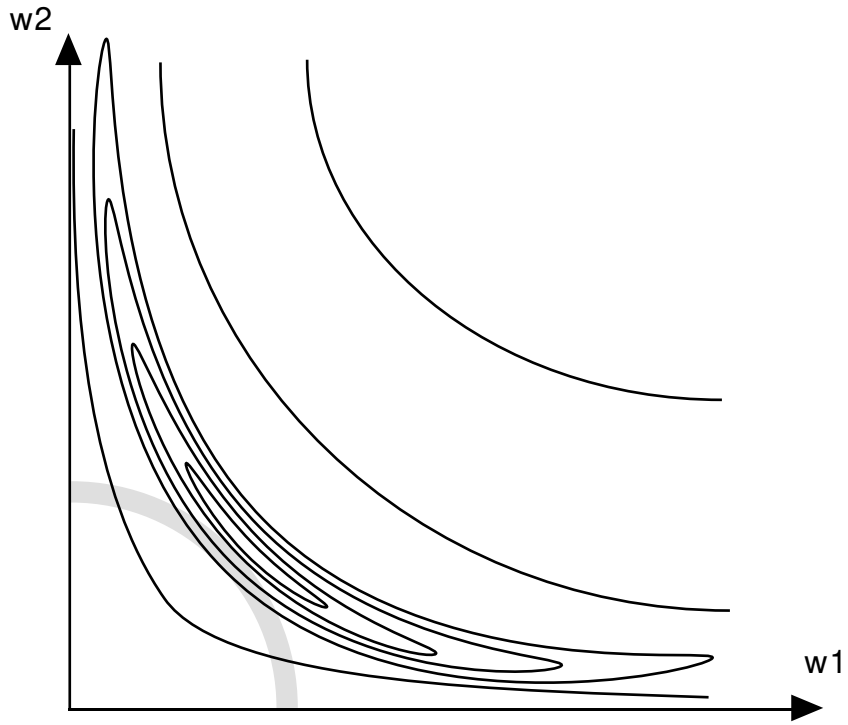


Fig 5.2 - Forme de la fonction de coût dans un réseau 1-1-1 pour l'auto-association. En grisé, la zone des poids initiaux suggérée par l'heuristique de la section 5.2.2.1

En présence d'une sigmoïde ces deux ravins existent toujours. Leur fond n'est plus de coût constant, mais présentent deux minima aux environs des points (1,1) et (-1,-1).

De plus, ce qui aggrave les choses, le coût C, comme la fonction sigmoïde, est borné. La fonction de coût possède donc des plateaux lorsque w_1 et w_2 sont élevés, c'est à dire des régions où $C(\mathbf{w})$ varie peu et où le gradient est presque nul (fig 5.2). Il faudrait alors un gain important pour se diriger rapidement vers le minimum. Mais un tel gain ne permet plus d'atteindre le minimum, au fond de sa crevasse.

5.2.2 Heuristiques pour le perceptron multi-couches.

Quelques règles heuristiques simples permettent d'améliorer sensiblement la situation. On considère dans cette section un perceptron multi-couches. Chaque unité effectue le calcul suivant

$$s = f\Big(\sum w_k x_k \Big) \qquad (5.10)$$

où f représente la fonction sigmoïde.

5.2.2.1 Initialisation des poids.

Les conséquences d'un mauvais conditionnement sont sensiblement aggravées par la présence de "plateaux", induits par la non linéarité de la sigmoïde.

On peut alors [1], choisir des poids initiaux de façon à utiliser surtout la partie linéaire à fort gradient de f. Il est également souhaitable de rester assez près des régions de forte courbure, afin de pouvoir, le cas échéant, exploiter la capacité additionnelle que fournissent ces non linéarités (cf §4.6.2.2 iii).

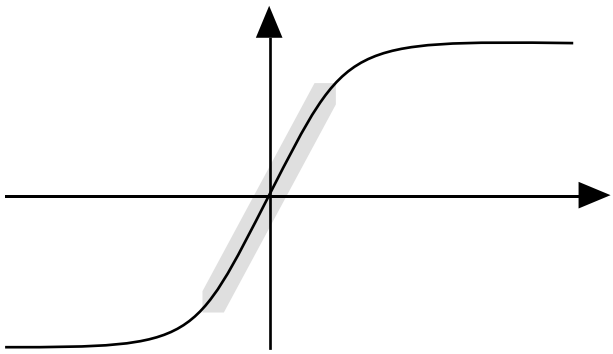


Fig 5.3- La sigmoïde, et la zone cible pour les sommes pondérées.

Cela signifie que les moyennes des sommes pondérées doivent être nulles, que leurs écart-types doivent correspondre à peu près au point de courbure maximum de la sigmoïde (fig 5.3).

Les statistiques des sommes pondérées dépendent à la fois de celles des entrées x_k et de celles des poids w_k. De plus, il est plus simple de considérer des *entrées et des poids de moyenne nulle*, c'est à dire

$$\langle x \rangle = 0, \quad \langle w \rangle = 0, \quad \mathrm{var}(x) = \langle x^2 \rangle, \quad \mathrm{var}(w) = \langle w^2 \rangle$$

Si les entrées sont bien représentées par des variables indépendantes, de moyenne nulle, de même variance, la variance de la somme pondérée est donnée par

$$\mathrm{var}(\sum x_k w_k) = n.\mathrm{var}(xw) = \langle x^2 w^2 \rangle = n \langle x^2 \rangle \langle w^2 \rangle = n \, \mathrm{var}(x) \, \mathrm{var}(w)$$

en se rappelant que $\langle xy \rangle = \langle x \rangle \langle y \rangle$, et $\mathrm{var}(x+y) = \mathrm{var}(x) + \mathrm{var}(y)$ lorsque x et y sont indépendants. L'écart-type des sommes pondérées est donc proportionnel à l'écart-type des poids et la racine du nombre n d'entrées pour chaque unité.

On peut donc initialiser les poids selon une *loi uniforme* sur un intervalle

$$\left[-\frac{K}{\sqrt{n}} , \quad \frac{K}{\sqrt{n}} \right] \qquad (5.11)$$

La constante K dépend en fait de la forme exacte de la fonction sigmoïde. Sa détermination est en fait imprécise, car l'écart type idéal des sommes pondérées n'est lui même pas déterminé de façon précise.

---

1 **Bottou L.:** *Reconnaissance de la parole par réseaux multi-couches* - Neuro-Nîmes 88 - pp 197-218, EC2 (1988)

De plus, la grossièreté des suppositions statistiques sur les entrées rend inutile toute évaluation théorique de cette zone.

On obtient de bons résultats avec $0.5<K<2$ avec des entrées prises dans l'intervalle $[-1,1]$ et en utilisant la sigmoïde suivante:

$$1.71 \tanh(\frac{2}{3}x) \qquad (5.12)$$

Une approche efficace pour déterminer K consiste à utiliser $K=1$ et à mesurer les écarts-types des sommes pondérées. Une bonne valeur de K est alors le quotient de l'écart-type mesuré et de l'écart-type optimal.

5.2.2.2 Stratégies pour les gains.

L'expérience montre que, tout au long de l'apprentissage, l'écart-type des sommes pondérées reste comparable à l'abcisse du point de courbure maximum de la sigmoïde. Cela signifie donc que l'écart-type des poids reste inversement proportionnel à la racine du nombre d'entrées de l'unité en aval. On peut alors disposer d'un *gain spécifique pour chaque unité,* lui aussi inversement proportionnel à la racine du nombre d'entrées.

Cette opération constitue une transformation de l'espace des paramètres: L'utilisation de gains différents consiste en fait à effectuer des étirements ou des écrasements sélectifs de l'ellipse de la figure 5.1. On peut ainsi améliorer sensiblement le conditionnement…

Ces gains doivent-ils rester constant? Dans la section 5.1, en effet, on a identifié des cas simples de convergence à gain constant, et des limites supérieures pour les gains ont été identifiées. Les théorèmes de la section 3 s'appliquent cependant au cas des gains décroissants. C'est en effet la meilleure façon de s'assurer que les gains finiront par être assez faibles pour permettre la convergence.

Il est possible d'utiliser une loi fixée de décroissance des gains. Il vaut mieux cependant décider de la décroissance des gains en se fondant sur les évolutions du coût. Lorsque l'on constate une stabilisation du coût, on divise par exemple les gains par deux. Dans le cas d'un algorithme stochastique, le coût n'est pas toujours décroissant. On assiste en effet à des phénomènes de changement de bassin d'attraction, (cf. §3.3.2.2), qui sont d'ailleurs souhaitables. Il faut alors raisonner sur des mesures lissées du coût.

Une autre solution consiste à calculer explicitement le hessien, et à utiliser ces information pour contrôler les gains.

5.2.3 Utilisations du hessien pour contrôler les gains.

Dans cette section, on généralise l'idée qui consistait à utiliser des gains différents pour chaque unité, où pour chaque paramètre, en adoptant un gain matriciel ε. L'équation d'ajustement des paramètre (3.5) s'écrit alors

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \varepsilon \nabla C(\mathbf{w}_t) \qquad (5.13)$$

Il est aisé d'étendre les démonstrations de convergence de la section 3.3 au cas des gains matriciels, lorsque ceux-ci sont des matrices symétriques, définies positives.

5.2.3.1 Transformation de l'espace des paramètres.

Il est possible de modifier la structure du paramétrage de la fonction de coût au moyen d'une transformation de l'espace des paramètres [1]. On suppose pour cela que les paramètres $\mathbf{w}$ sont déterminés par une fonction Φ de nouveaux paramètres $\mathbf{v}$.

$$\mathbf{v} \xrightarrow{\Phi} \mathbf{w} \xrightarrow{C} C(\mathbf{w}) = C \circ \Phi(\mathbf{v}) = C(\Phi(\mathbf{v}))$$

On optimise alors la fonction $C(\Phi(\mathbf{v}))$ des paramètres $\mathbf{v}$. L'équation (2.32) montre comment un tel formalisme permet de concevoir des systèmes dits à poids partagés.

Cette notion permet également d'appliquer une transformation de l'espace des poids qui améliore le conditionnement [2]. Il suffit pour cela de rendre circulaire les ellipses de la figure 5.1.

Le hessien est une matrice symétrique. De plus, il est défini positif au voisinage d'un minimum. Il s'écrit donc

$$\mathbf{H} = \mathbf{P}^t \mathbf{D} \mathbf{P} = \mathbf{P}^t \Lambda \Lambda \mathbf{P} \qquad (5.14)$$

où $\mathbf{P}$ est une matrice orthogonale, c'est à dire que $\mathbf{P}\mathbf{P}^t = \mathbf{I}$, et où $\mathbf{D}$ et Λ sont deux matrices diagonales à coefficients positifs. Les termes diagonaux de Λ sont les racines carrées de ceux de $\mathbf{D}$. On considère alors la transformation de l'espace des poids suivante

$$\mathbf{w} = \Phi(\mathbf{v}) = \mathbf{P}^t \Lambda^{-1} \mathbf{v} \qquad (5.15)$$

---

1 **Le Cun Y.:** *A theoretical framework for back-propagation* - Proceedings of the 1988 Connectionist Models Summer School, pp 21-28, CMU, Pittsburgh, PA, Morgan Kaufmann (1988)

2 **Le Cun Y.:** *Generalization and Network Design Strategies - (Expanded version)* - Technical report CRG-TR-89-4, University of Toronto (1989)

En reprenant (5.3), le coût s'écrit alors

$$\begin{aligned} C &= C^{opt} + \mathbf{v}^t\Lambda^{-1}\mathbf{P}\mathbf{P}^t \Lambda \Lambda \mathbf{P}\mathbf{P}^t\Lambda^{-1}\mathbf{v} + o(|\mathbf{v}|^2) \\ &= C^{opt} + \mathbf{v}^t\mathbf{v} + o(|\mathbf{v}|^2) \end{aligned} \qquad (5.16)$$

Le nouveau hessien est donc la matrice identité. Toutes ses valeurs propres sont bien sûr égales, le problème est donc bien conditionné. L'équation de modification des poids s'écrit alors

$$\mathbf{v}_{t+1} = \mathbf{v}_t - \varepsilon\nabla C o\Phi(\mathbf{v}_t) = \mathbf{v}_t - \varepsilon\,(\mathbf{P}^t\Lambda^{-1})^t\nabla C(\mathbf{w}_t) \qquad (5.17)$$

5.2.3.2 Méthodes de Newton.

i Algorithme de Newton.

On peut exprimer cette modification directement dans l'espace des $\mathbf{w}$,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \varepsilon\,\mathbf{P}^t\Lambda^{-1}(\mathbf{P}^t\Lambda^{-1})^t\nabla C(\mathbf{w}_t) = \mathbf{w}_t - \varepsilon\,\mathbf{H}^{-1}\nabla C(\mathbf{w}_t) \qquad (5.18)$$

Cela signifie simplement que l'on utilise un gain matriciel $\varepsilon\mathbf{H}^{-1}$.

On remarque de plus que si $\varepsilon=1$ et si $C(\mathbf{w})$ est exactement quadratique, l'équation (5.18) permet d'atteindre directement le minimum. On appelle cela la méthode de *Newton*.

Elle possède deux inconvénients:

- L'inverse du hessien n'est une matrice symétrique, définie positive que si le hessien l'est. Cela signifie que la méthode de Newton n'est applicable que dans une zone de courbure positive.

- Lorsque l'on utilise quelques milliers de paramètres, comme cela est fréquent dans les dispositifs connexionnistes, le hessien possède quelques millions d'éléments. Il n'est question, ni de calculer, ni de stocker, ni d'inverser une telle matrice.

Lorsque la taille du hessien est assez limitée pour permettre de le stocker, l'inversion reste d'un coût prohibitif. Il existe cependant des méthodes, comme BFGS (Broyden-Fletcher-Goldfarb-Shanno, voir [1], [2]) qui permettent d'évaluer progressivement une l'inverse du hessien, en mesurant l'écart entre $C(\mathbf{w}_{t+1})$ et $C(\mathbf{w}_t)+(\mathbf{w}_{t+1}-\mathbf{w}_t)\nabla C(\mathbf{w}_t)$. Cette méthode n'est cependant pas applicable au cas du gradient stochastique, car on ne connaît pas exactement $C(\mathbf{w}_t)$, ni même $\nabla C(\mathbf{w}_t)$. Or l'accélération

---

1 **Jacobs, David A.H.:** *The State of the Art in Numerical Analysis*, Chapter III.1, §3.6, Academic Press, (edition 1977)

2 **Watrous R.L.:** *Learning Algorithms for Connectionist Networks: Applied Gradient Methods for Nonlinear Optimisation*, Procs of 1st Int. Conf. on Neural Networks, San Diego, CA, vol IV, pp 619-627, (1987)

procurée par l'aspect stochastique de l'algorithme est comparable [1] à celle que procure BFGS: On ne gagne rien à utiliser BFGS en abandonnant l'approche stochastique.

ii Algorithme Quasi-Newton.

En général, le hessien est trop grand pour que l'on puisse même se poser cette question. On est alors réduit à travailler avec une approximation diagonale du hessien, ce qui limite également les problèmes d'inversion.

En notant σ_k la dérivée seconde $\frac{\partial^2 C}{\partial w_k^2}$, on obtient une méthode, dite, *quasi-Newton*, qui s'écrit

$$\Delta w_k = -\frac{\varepsilon}{\sigma_k}\frac{\partial C}{\partial w_k} \tag{5.19}$$

Le gain matriciel ε est alors la matrice diagonale des ε/σ_k. Cet algorithme ne fonctionne toujours pas lorsque l'approximation diagonale du hessien n'est pas définie positive, c'est à dire lorsque un σ_k est négatif ou nul.

ii Algorithme de Levenberg-Marquardt.

La solution consiste à rechercher une *approximation définie positive du hessien*. La plus connue est l'approximation de *Levenberg-Marquardt* [2], qui s'applique aux algorithmes de régression, pour lesquels le coût possède la forme

$$C = \int (\mathbf{Y}-f(\mathbf{X},\mathbf{w}))^2\, p(\mathbf{x})d\mathbf{x} \tag{5.20}$$

La diagonale du hessien s'écrit alors

$$\mathbf{H} = 2\int \nabla f(\mathbf{X},\mathbf{w})\nabla f(\mathbf{X},\mathbf{w})^t \; - \nabla^2 f(\mathbf{X},\mathbf{w})^t(\mathbf{Y}-f(\mathbf{X},\mathbf{w}))\, p(\mathbf{x})d\mathbf{x} \tag{5.21}$$

On décide alors de négliger les non-linéarités qui proviennent de la forme de f, et de ne corriger que le problème de conditionnement issus de la distance quadratique que l'on minimise. C'est à dire

$$\mathbf{H} \approx 2\int \nabla f(\mathbf{X},\mathbf{w})\nabla f(\mathbf{X},\mathbf{w})^t\, p(\mathbf{x})d\mathbf{x} \tag{5.22}$$

---

1 **Becker S., Le Cun Y.:** *Improving the Convergence of Back-Propagation Learning with Second Order Methods.*- In Procs. of the 1988 Connectionist Models Summer School, pp 29-37, San Mateo, Morgan Kaufmann, (1989)

2 **Press W.H., Flannery B.P., & al.:** *Numerical Recipes* - Cambridge University Press, Cambridge (1988)

Dans le cas d'un perceptron multi-couches, on calcule une approximation diagonale, définie positive du hessien proche de l'approximation de Levenberg-Marquardt. On calcule aisément la formule de récurrence suivante (cf. §2.3.4.1 pour les notations):

$$\frac{\partial^2 C_{\mathbf{x}}}{\partial a_j^2} = f'(a_j)^2 \sum_{k \in Aval(j)} W_{jk}^2 \frac{\partial^2 C_{\mathbf{x}}}{\partial a_k^2} - f''(a_j) b_j$$

$$\frac{\partial^2 C}{\partial W_{kj} \partial W_{qj}} = \int \frac{\partial^2 C_{\mathbf{x}}}{\partial a_j^2} s_k s_q \, p(\mathbf{x}) d\mathbf{x} \qquad (5.23)$$

dans laquelle on peut négliger la dérivée seconde $f''(a_j)$ de la sigmoïde. On peut alors applique la règle de mise à jour des poids:

$$\Delta W_{jk} = - \frac{\varepsilon}{\mu + \sigma'_{jk}} \frac{\partial C}{\partial W_{jk}} \qquad (5.24)$$

où σ'_{jk} représente le terme diagonal de l'approximation positive du hessien. Le terme $\mu > 0$ sert à garantir que le dénominateur ci dessus est non nul.

iv Evaluation de la diagonale du hessien dans le cas stochastique.

Lorsque l'on désire utiliser des gains proportionnels à ceux suggérés ci-dessus dans un algorithme stochastique, il faut pouvoir évaluer la diagonale du hessien. On a, comme en (5.23),

$$\mathbf{H}(\mathbf{w}) = \int \nabla^2_{\mathbf{w}} J(\mathbf{x}, \mathbf{w}) \, p(\mathbf{x}) d\mathbf{x} \qquad (5.25)$$

Or, on ne connaît pas $p(\mathbf{x})$. De plus, le hessien dépend de $\mathbf{w}$, et donc évolue dans le temps. On peut en approcher les termes au moyen d'une *intégration avec perte:*

$$(s_k)_{t+1} = (1-\gamma)(s_k)_t + \gamma \frac{\partial J}{\partial w_k}(x,w) \qquad (5.26)$$

avec $0 < \gamma < 1$, une constante de l'ordre de l'inverse du nombre de types d'exemples.

v Conclusion.

Appliquer de tels algorithmes requiert de calculer non seulement les dérivées $\partial J / \partial w_k$, mais aussi les dérivées secondes $\partial^2 J / \partial w_k^2$. Cela représente une perte de temps, plus ou moins bien compensée par l'accélération apportée par un meilleur contrôle du gain. En fait, on ne constate pas d'accélérations significatives dans des cas concrets. En revanche, il n'est plus nécessaire de surveiller le système pour ajuster les gains au fur et à mesure que décroît le coût.

Ce meilleur contrôle des gains permet également d'atteindre de meilleurs optima de la fonctionnelle empirique. Cela ne garantit cependant pas que ces algorithmes ont une influence positive sur la qualité réelle du système (cf. chp. 4).

Les dérivées secondes peuvent avoir d'autres utilisations: Elles fournissent des indices [1] sur les importances respectives des diverses parties de notre système d'apprentissage. Cela permet éventuellement de le simplifier, en réduisant sa capacité (cf. chp. 4), et en augmentant ainsi les facultés de généralisation.

5.3 Outils de simulation.

Ces quelques principes sont matérialisés par divers programmes. Dans le jargon connexionniste, on les appelle fréquemment *simulateurs*. Divers facteurs, scientifiques ou économiques, ont contribué à donner à ces programmes une place de plus en plus importante.

5.3.1 Intérêt.

i Aspects techniques.

Comme toujours le besoin de programmer des algorithmes d'adaptation et d'apprentissage d'une manière réutilisable apparaît bien vite. Bien qu'il reste malsain d'utiliser de tels algorithmes sans avoir la moindre idée de leur fonctionnement, chacun peut alors les utiliser en se dispensant au moins d'une pénible tâche de mise au point.

- Dans le cas d'un problème nécessitant un algorithme d'adaptation, on souhaite comparer divers algorithmes et sélectionner le meilleur, comme on l'a suggéré dans la section 4.6. Mais il est souvent couteux, voire dangereux, de mettre en œuvre un système expérimental, simplement pour en mesurer la qualité.

  C'est pourquoi on souhaite disposer d'*outils de simulation*, qui permettent de mettre au point notre système adaptatif dans un contexte proche des conditions d'exploitation.

- Dans le cas d'algorithmes d'apprentissage, on utilise deux programmes: L'un est le programme d'exploitation, qui utilise en fait un jeu de paramètres $\mathbf{W}$ figés. Ces paramètres sont déterminés par un programme d'apprentissage, à l'aide d'un ensemble d'exemples.

---

1 **Le Cun Y., Denker J.S., Solla S.:** *Optimal Brain Damage* - Advances in Neural Information Processing Systems II, Morgan Kaufmann Denver (1990)

Là encore, on désire comparer plusieurs façons de paramétrer notre système: Le programme d'apprentissage doit donc permettre d'une part d'entraîner des systèmes différents, d'autre part de reproduire les conditions d'exploitation afin de pouvoir comparer la qualité de ces systèmes. C'est également un programme de *simulation*.

Des outils logiciels généraux pour de tels algorithmes doivent offrir d'une part des algorithmes d'apprentissage offrant un bon compromis entre généralité et efficacité. Toutes les astuces nécessaires pour améliorer la vitesse d'apprentissage sont souhaitables. Cela concerne en particulier le contrôle du gain, les paramètres initiaux, la possibilité de calculer des approximations du hessien…

Ils doivent en outre présenter une interface souple, permettant certes de visualiser et d'altérer les diverses variables de l'algorithme, mais permettant aussi de reproduire les *conditions de mesure de la qualité du système*. La plus grande partie du programme est alors consacrée à ces problèmes d'interface.

ii Aspects économiques.

On constate depuis deux décennies une chute vertigineuse du coût des ordinateurs. La relative cherté des logiciels, et la rareté du personnel qualifié pour le créer constituent alors l'obstacle majeur à un développement encore accéléré des techniques informatiques.

Chaque fois qu'un espoir apparaît de réduire les coûts de développement du logiciel, systèmes experts ou réseaux connexionnistes, la pression économique suscite une euphorie collective. Des chœurs d'analystes économiques promettent immédiatement un avenir radieux à ces nouvelles techniques. Ces promesses sont fondées en fait sur un recensement réel des besoins de l'industrie en logiciels "intelligents", et c'est en fait une erreur: On oublie d'évaluer *l'adéquation de ces nouvelles techniques aux besoins* que l'on recense.

Des myriades de sociétés, anciennes ou nouvelles, désirent alors acquérir une place de choix dans cet avenir radieux. Afin d'acquérir une réputation, beaucoup cherchent alors à aborder un palier intermédiaire: le marché des outils de simulation, destinés aux centres de recherche et développement.

Le niveau de cette bataille commerciale est assez décevant: On perdrait presque de vue qu'il vaut bien mieux disposer d'un bon algorithme et d'une interface difficile d'accès que du contraire. On oublie déjà que le simulateur le plus rapide n'est pas celui qui effectue le plus d'opérations en une seconde, mais au contraire celui qui requiert le moins d'opérations pour apprendre un problème donné à un taux de performance donné.

On peut se demander si cette indigence est le fait d'une pauvreté de l'offre ou d'une pauvreté de la demande…

5.3.2 Exemples.

Voici, en guise d'exemple, une brève description des trois principaux outils utilisés dans notre laboratoire.

i Un précurseur: HLM.

HLM [1], pour Hierarchical Learning Machine, fut écrit par Yann Le Cun dès 1985. Ce simulateur contenait essentiellement une version stochastique de l'algorithme de rétro-propagation qui s'est avérée très efficace, et qui est à l'origine de la plupart des travaux du laboratoire.

Son interface textuelle était très simple, et se limitait à une vingtaine de commandes essentielle. Elles permettaient d'effectuer des itérations d'apprentissage, de mesurer la performance sur quelques exemples, et de lire des fichiers d'exemples, des fichiers décrivant l'architecture d'un perceptron multi-couches, et de sauvegarder les poids déterminés par l'apprentissage.

Il a cessé d'être utilisé dans notre équipe après trois ans de bons et loyaux services, mais conserve encore, je crois, quelques fans en France.

ii SN et ses dérivés.

L'idée du simulateur SN [2] est relativement simple: C'est un interprète Lisp, qui contient des primitives spécialisées pour toutes les opérations coûteuses que doivent effectuer nos algorithmes. Les boucles internes sont écrites en C; les boucles externes sont exécutées par l'interprète Lisp.

Au démarrage, environ 500 fonctions Lisp sont chargées, définissant de multiples protocoles d'apprentissage, de test, d'initialisation des poids, de définition des réseaux et des exemples, des routines permettant d'effectuer des représentations graphiques, et même parfois des routines permettant de lire et d'écrire des fichiers aux formats des simulateurs HLM et Galatée…

De plus, il est toujours possible de modifier ces fonctions au cours de l'apprentissage. Cette souplesse nous a valu une bonne surprise: Initialement conçu pour des perceptrons multi-couches, SN simule de bonne grâce quelques autres algorithmes classiques. Cette surprenante propriété n'est pas étrangères aux tentatives d'unification que constituent les chapitres 3 et 4.

---

1 **Le Cun Y.:** *Manuel du simulateur HLM* - dans "Modèles Connexionnistes de l'apprentissage": Thèse de Doctorat de l'Université Paris 6, Paris (1987)

2 **Bottou L., Le Cun Y.:** *SN: Un simulateur pour réseaux connexionnistes* - Neuro Nîmes 88, pp 371-382, EC2 ed. (1988)

La rapidité et la souplesse d'un tel système repose bien sûr sur le choix judicieux des parties de l'algorithme écrites en C, et des parties écrites en Lisp. Aujourd'hui, le temps d'interprétation représente environ 10% du temps de calcul.

C'est également un ensemble de programmes bien plus complexe que HLM. Après avoir terminé la première version en juin 1987, je l'ai confié à Yann qui l'a solidement amélioré dans les six mois suivants. Fin 1988, après une réécriture totale, la version SN2 comblait quelques lacunes et corrigeait quelques erreurs. Elle représente maintenant plus d'un mégaoctet de C et de Lisp.

Depuis 1988, SN est devenu un banc d'essai apprécié dans plusieurs laboratoires, en France et à l'étranger. Il a servi à explorer les réseaux à poids partagés et à utiliser des approximations du hessien [1]. Il a été utilisé, entre autres, pour entraîner des systèmes de reconnaissance de code postaux [2], de détection sonar [3], de prévisions de séries temporelles [4] , et bien sûr de reconnaissance de la parole.

iii La librairie "Galatée" d'algorithmes connexionnistes.

Le développement d'outils de simulation a été planifié dans le cadre du programme européen Esprit. Toute l'ambition du projet "Pygmalion" consiste à définir ainsi un futur standard européen d'outils de simulation pour réseaux connexionnistes.

Diverses tâches ont été confiées à notre laboratoire, dont l'écriture, fin 1989, d'une librairie C, regroupant un grand nombre d'algorithmes connexionnistes [5]. Cette tâche était destinée d'une part à évaluer les difficultés d'implémentation de chaque algorithme, en prévision des outils futurs, d'autre part à être diffusée largement, afin de profiter d'une base installée aussi large que possible.

Cette librairie a été baptisée "Galatée", en souvenir de la statue qu'aima vainement son créateur, Pygmalion. Dans le but de satisfaire un grand nombre de chercheurs, il importait d'aborder chaque algorithme avec sa terminologie spécifique, en accord avec une bibliographie de référence.

---

1 **Le Cun Y.:** *Generalization and Network Design Strategies - (Expanded version)* - Technical report CRG-TR-89-4, University of Toronto (1989)

2 **Le Cun Y., Boser B., & al.,** (Bell Laboratories)**:** *Handwritten Digit Recognition with a Back-Propagation Network-* Advances in Neural Information Processing Systems II, Morgan Kaufmann, (1990)

3 **Bollivier, M. de., Lemer A., Tanguy J.:** *Reconnaissance de bruits sous-marins par réseaux multi-couches* - Neuro-Nîmes 88, EC2, pp 423-430, (1988)

4 **Varfis A.:** *Univariate Economic Time Series Forecasting by Connectionnist Methods*, Proc. of INNC'90, vol 1, pp 342-345, Paris (1990)

5 **Mejia C., Bottou L., Fogelman Soulié F.:** *Galatea: a C library for connectionist applications* - Procs. of INNC'90, Paris, 9-13, (1990)

C'est pourquoi Galatée est composée de multiples modules distincts, au détriment il est vrai de toute perspective d'unification des algorithmes correspondants. Chaque module comporte un ensemble de fonctions C permettant de définir un réseau et de l'exploiter, et est accompagné d'une petite interface textuelle qui accepte une vingtaine de commandes.

La plupart de ces modules ont été écrits séparément par différents membres du laboratoire, à l'aide de routines communes nommées AIP (Algorithm Independent Part), inspirées de la mécanique d'un tableur: Les données sont rangées dans des vecteurs et des matrices, et on peu associer à chaque élément de ces vecteurs ou matrices une règle de calcul.

Le développement de ce mégaoctet de programmes a été supervisé par Carlos Mejia, que j'ai assisté de mon mieux pendant l'étape de définition. Cela a été une expérience unique: il fallait pour chaque module se plier à la façon de penser du créateur de chaque algorithme, tout en conservant une vision globale.

5.4 Conclusion.

La mise au point d'une technologie fiable et rapide pour des algorithmes d'adaptation et d'apprentissage est un travail assez pénible. Cependant quelques ordres de grandeur suffiront à en montrer l'intérêt:

- Une programmation plus efficace d'un même algorithme permet de diviser le temps d'apprentissage par un facteur 2 à 5.

- Une carte accélératrice spécialisée permet de gagner un facteur 10 en rapidité. Un an plus tard, cette performance est dépassée par la dernière station de travail de chaque constructeur.

- Améliorer un algorithme d'apprentissage à bon escient permet parfois de diviser le temps d'apprentissage par 1000!

  Nous en avons fait l'expérience sur deux tâches comparables de reconnaissance de chiffres manuscrits par perceptron multi-couches: Le temps d'apprentissage a été réduit d'environ une centaine d'heures en 1986 [1] sur un Vax8650 à cinq minutes en 1989 [2] sur une station de travail Sun4.

---

1 **Bourrely J., Bottou L.:** *Neur3: un réseau de neurones pour la reconnaissance de caractères* - Manuscrit non publié (décembre 1986).

2 **Le Cun Y.:** *Generalization and Network Design Strategies* - Connectionism in Perspective, Zurich, Switzerland, Elsiever (1989)

Plusieurs auteurs ([1] par exemple) ont récemment rapporté des résultat comparables.

1 **Haffner P., Waibel A., Shikano K.:** *Fast Back-Propagation Methods for Neural Networks in Speech*. Proc. from the Fall meeting of the Acoustical Society of Japan. (1988)

6 Reconnaissance de la parole.

6.1 Le signal de parole.

Cette section s'appuie principalement sur les ouvrages [1] et [2] et le rapport [3], qui sont à la fois bien documentés, et fort précis. Certaines parties exploitent des informations d'autres sources, qui sont alors citées au cours du développement. Sont décrits ici, dans un ordre thématique, les divers problèmes qu'aurait à résoudre un système achevé de reconnaissance de la parole, ainsi que quelques méthodes actuellement utilisées.

6.1.1 Physique du signal de parole.

Le signal de parole est d'abord un signal sonore. C'est donc une onde matérielle de faible puissance se déplaçant dans l'air. Avant même que nous puissions le capter, le signal est sujet à des perturbations importantes: bruit ambiant, résonances, phénomène d'écho, ...

---

1 **Liénard J.S.**: *Les processus de la communication parlée* - Masson (1977).

2 **O'Shaugnessy D.:** *Speech Communication, human and machine* -Addison Wesley. (1987).

3 **Singer H.:** *Utilisation de dissylabes pour la reconnaissance de la parole* - Rapport LIMSI, 88-4, (1988)

Le signal de parole couvre quasiment toute l'entendue du spectre audible. En pratique on peut se limiter à la bande 50-5000Hz. Il est remarquable que des personnes agées n'entendent quasiment plus les fréquences supérieures à 1800Hz et comprennent fort bien des conversations, ce qui représente une performance remarquable à mettre au compte du système de reconnaissance humain. Les modes de propagation du signal sonore dépendent aussi de la fréquence. Les plus hautes (> 2000Hz pour un locuteur ordinaire) se déplacent souvent de façon privilégiée dans une ou plusieurs directions. Les fréquences plus faibles se déplacent de façon plus isotrope. On parle d'*ondes planes* et d'*ondes sphériques* bien que ces modèles représentent des cas limites. Ces problèmes sont bien connus des preneurs de son.

6.1.2 Caractéristiques spectrales.

L'outil de base de traitement et de représentation de la parole est le *spectrogramme*. Un spectrogramme est la transcription des données bidimensionnelles (amplitude/temps) en un motif tridimensionnel (énergie/fréquence/temps). Il est usuellement représenté avec le temps en abscisse, la fréquence en ordonnée, alors que les énergies sont notées en niveaux de gris.

Un spectrogramme est le module de l'intégrale de Fourier sur une fenêtre temporelle:

$$sp(t, f) = \left| \int_{t-T}^{t+T} W(t-\tau)\ X(\tau)\ e^{2i\pi f\tau}\, d\tau \right| \qquad (6.1)$$

dans laquelle $X(\tau)$ est le signal, T est la demi longueur de la fenêtre temporelle, $W(\tau)$ est une fonction fenêtre douce, par exemple:

$$W(\tau) = \cos\left(\frac{\pi\tau}{T}\right) \qquad (6.2)$$

Sur un tel diagramme, on peut distinguer plus ou moins distinctement plusieurs informations pertinentes:

- La *tonalité* ou pitch, fréquence de la première harmonique observable. Elle est de l'ordre de 120Hz pour des hommes et 250Hz pour des femmes. Cela représente la hauteur sur laquelle est “chantée” la parole.

- Les séparations voyelles-consonnes sont en général assez visibles: les voyelles sont beaucoup plus énergétiques que les consonnes. Les silences caractéristiques des consonnes occlusives sont par exemple toujours bien marqués.

- Les *formants* sont des zones quasi horizontale du spectrogramme dans lesquelles les fréquences harmoniques sont renforcées. Ces structures, principalement visibles dans les voyelles apportent

des informations précieuses. Les mouvements des formants semblent aujourd'hui être l'une des clefs de la parole. On s'intéresse habituellement aux trois formants de fréquences les plus basses.

- Le *voisement*: C'est à dire la présence ou non de l'ensemble d'harmoniques qui caractérisent les sons *voisés* par opposition aux sons *chuchotés* .

Quelques dispositions peuvent être prises pour améliorer la qualité des spectrogrammes:

- Augmenter l'énergie des hautes fréquences par une première étape de filtrage. Cela imite en fait les résonances du pavillon de l'oreille.

- Utiliser une échelle linéaire pour les basses fréquences (<1800Hz) et logarithmique pour les hautes fréquences. L'une de ces échelles mixtes est appelée *échelle de Bark* .

6.1.3 Du signal au message.

Il est facile de définir abstraitement le problème de la reconnaissance de la parole. Il consiste à passer du signal au message. On sait bien ce qu'est physiquement un signal sonore. Un siècle de téléphonie nous ont appris a en domestiquer bien des aspects, à l'enregistrer, le coder, le reproduire, le transporter.

La notion de message est bien plus ambiguë. On s'intéresse en fait à ce qu'il signifie. Dans un programme d'ordinateur, il faut alors savoir représenter "ce qu'il signifie". Cela n'est possible que dans des cas restreints, comme reconnaître les réponses affirmatives des réponses négatives. On impose donc au locuteur un vocabulaire restreint et une grammaire réduite. Dans beaucoup de cas, d'ailleurs, cette contrainte est naturelle: dans le cas d'un dispositif de numérotation vocal dans une cabine téléphonique, par exemple.

On se limite donc à la reconnaissance de mots ou de phonèmes. Cela pose dans les deux cas un double problème de *classification* et de *segmentation*: il faut identifier quel mot ou phonème est prononcé durant quel intervalle de temps. Or, on ne sait pas délimiter un mot ou un phonème dans le temps de façon satisfaisante. A cause de la lenteur de nos mouvements musculaires, chaque son émis est précédé d'une étape préparatoire, et possède une influence sur les sons suivants.

En pratique, le signal est plus composé de transitions que de signaux stables, correspondant, par exemple, à chaque phonème. On appelle cela le phénomène de *coarticulation*.

Caractériser ces transitions est donc une tâche critique pour identifier une séquence de phonèmes. Il suffit, pour s'en persuader d'essayer de couper une bande magnétique de façon à isoler un seul phonème. Entendu isolément, il ne correspond pas à ce que l'on croyait entendre lorsqu'on l'écoutait dans son contexte.

Pour ne rien gâter, un même locuteur peut *articuler* avec soin ou non. Il peut *parler rapidement*: cela ne compresse pas du tout le signal dans le temps, mais augmente le nombre d'élisions, et fait disparaître les parties stables des formants qui devaient caractériser les voyelles, Il peut *chuchoter ou crier*, et l'on comprend que cela ne modifie pas seulement le niveau d'énergie. Deux locuteurs peuvent posséder une *hauteur mélodique moyenne* différente: de moins de 100Hz pour la voix grave d'un homme jusqu'à 400Hz pour un enfant. Ils peuvent avoir des *fréquences formantiques* différentes dans un rapport de 1 à 1.3, des *étendues mélodiques* différentes (voix chantante), des *vitesses d'élocution* très variables, ou des timbres différents, expression imprécise qui désigne tout le reste.

6.1.4 Contraintes.

Tout ceci explique que l'on ne sache reconnaître la parole que dans des situations bien précises. Les systèmes de reconnaissance de la parole commettent rarement plus de 5% d'erreurs. Ils ne se distinguent que par les contraintes qu'il requièrent pour atteindre ce taux de performance.

On peut résumer les contraintes les plus courantes par les oppositions suivantes:

- Est il *mono-locuteur*, *multi-locuteurs*, ou *indépendant du locuteur* ? Cela signifie respectivement, qu'il est capable de ne reconnaître la parole que prononcée par un unique locuteur, par un ensemble bien précis de locuteurs, ou bien que tout locuteur convient.

- Reconnait-il des *mots isolés* ou de la *parole continue* ? Restreint-il les phrases reconnues à une grammaire contraignante ou non ?

- Quelle est la taille du vocabulaire utilisé ?

- Quelle est la qualité de la parole qu'il utilise ?

6.2 Chaînes de traitement.

Un système de reconnaissance de la parole est habituellement constitué de trois composants indépendants:

- Un dispositif d'acquisition du signal, usuellement un microphone ou une bande magnétique, associés à un système d'amplification et de filtrage. Le signal est ensuite numérisé afin d'être traité.

- On pratique ensuite des prétraitements, dont le but est d'une part de réduire le flot de données dans le système à un niveau acceptable, d'autre part de présenter les données sous un format compatible avec l'étape de reconnaissance.

- La troisième étape consiste à extraire de ces données transformées une séquence de symboles (des mots ou des phonèmes, par exemple). C'est le point le plus délicat des systèmes de reconnaissance.

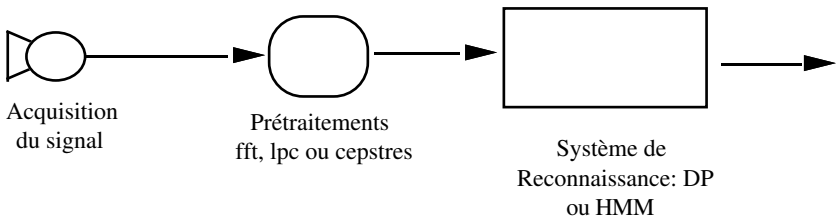


Fig 6.1-Une chaîne de reconnaissance de la parole.

Les séquences de symboles ainsi générées sont ensuite exploitées, par exemple par un programme d'analyse sémantique, dans le cas d'un dispositif de commande vocale, ou encore par un programme de transcription orthographique dans le cas d'une machine à écrire vocale. Ces programmes ne seront pas abordés ici; il faut savoir tout de même qu'ils représentent un problème en général extrêmement difficile.

6.3 Acquisition et prétraitements.

Le traitement informatique de la parole nécessite un signal numérique. On l'échantillonne usuellement à une dizaine de kHz, afin de conserver l'information spectrale jusqu'à environ 5 kHz. Comme les hautes fréquences sont souvent atténuées, on les renforce artificiellement à l'aide d'un filtre numérique. Cela ressemble en fait à l'opération accomplie par le pavillon de l'oreille.

L'étape suivante consiste à effectuer sur le signal numérisé quelques prétraitement en vue de réduire le bruit à un niveau acceptable. Il est parfois possible, lorsque l'on connait la nature des bruits qui peuvent altérer le signal, d'essayer d'extraire le signal utile, ou d'en améliorer la qualité [1]. De sensibles améliorations des performances des systèmes opérant en milieu bruité peuvent être attendues d'un bon traitement à ce niveau.

Beaucoup de systèmes de reconnaissance de la parole, cependant, opèrent sur du signal "propre", enregistré avec un microphone de haute qualité dans une salle sourde. De tels traitements sont alors peu utiles.

---

1 **Picone J., Johnson M.A., Hartwell W.T.:** *Enhancing the Performance of Speech Recognition with Echo Cancellation* - Procedings IEEE ICASSP 1988, S-Vol.1, pp 529-532, (1988)

A ce stade, l'information présente dans le signal représente plusieurs centaines de milliers de bits par seconde. De telles quantités de données sont fort difficiles à manipuler. Il importe alors d'extraire un jeu réduit de paramètres significatifs, en nombre limité (1000 à 2000 par seconde). Cela peut être fait de plusieurs manières:

6.3.1 Codage spectral, banc de filtres.

Le codage le plus ancien est le codage spectral. Il semble en effet acquis qu'un spectrogramme de bonne définition contient toute l'information nécessaire à l'identification de la parole ainsi codée. Avec un apprentissage important, un phonéticien est capable de "lire" un spectrogramme. Il fait néanmoins appel pour cela à ses connaissances linguistiques.

En diminuant à la fois la résolution fréquentielle et temporelle d'un spectrogramme, par moyennage, on peut compresser le signal de parole en environ cent vecteurs spectraux par seconde d'une quinzaine d'éléments chacun.

Un banc de filtres est une autre façon d'arriver au même résultat. Il suffit de disposer en parallèle d'une quinzaine de filtres numériques "passe-bande" recouvrant la plage de fréquence du signal, et de mesurer en sortie, l'énergie moyenne pendant de courts intervalles de temps.

Quelques aménagements ont prouvé leur utilité dans les systèmes de reconnaissance classiques. On s'attache souvent à répartir les bancs de filtres selon une échelle physiologique, dite *échelle de Bark* (ou de Mel), linéaire jusqu'à 1.8kHz, logarithmique au delà. D'autre part, on code fréquemment les *énergies sur une échelle logarithmique*.

Toute la difficulté de telles manipulations est de conserver un jeu de paramètres significatifs. Dans le cas de l'analyse spectrale, la perte d'information est concentrée dans l'opération de moyennage destinée à réduire les résolutions temporelles et fréquentielles du spectrogramme. On peut s'en convaincre en effectuant une transformée de Fourier inverse pour générer des signaux susceptibles d'être écoutés; ceux ci sont rarement intelligibles.

6.3.2 Codage prédictif linéaire.

Le codage prédictif linéaire [1] [2] consiste à représenter des tranches de 10ms à 30ms du signal par les coefficients $a_1 \ldots a_n$ d'un modèle autorégressif.

---

1 **Itakura F., Saito F.:** *Speech Analysis-Synthesis System Based on the Partial Autocorrelation Coefficients*. Acoust Soc of Japan Meeting (1969)

2 **Atal B.S., Hanauer S.:** *Speech analysis and synthesis by linear prediction of the speech wave*. Journal of the Acoustical Society of America. n° 50, pp 637-655, (1971)

Une méthode pour évaluer ces coefficients consiste à minimiser l'énergie du signal résiduel, dans la tranche de temps considérée, c'est à dire de la différence entre le signal modélisé $a_1X_{t-1}+a_2X_{t-2}+\ldots\ldots+a_nX_{t-n}$ et le signal réel X_t.

$$\sum_t \left(X_t - \sum_{i=1}^{n} a_i \, X_{t-i} \right)^2 \qquad (6.3)$$

En annulant les dérivées de l'énergie résiduelle, on obtient le système (6.4),

$$\forall\, i, \quad V_{i0} = \sum_{k=1}^{N} V_{ik} \, a_k \qquad (6.4)$$

dans lequel V_{ik} est la matrice d'autocorrélation du signal dans chaque tranche, définie par:

$$\forall\, i,k, \quad V_{ik} = \sum_t X_{t-i} \, X_{t-k} \qquad (6.5)$$

Cette matrice est une matrice de Toeplitz, i.e. $V_{ik} = \varphi(i-k)$, et le système (6.4) peut être résolu rapidement à l'aide de la méthode de Levinson-Durbin [1].

Le codage prédictif linéaire est actuellement la technique la plus répandue de compression du signal de parole, et est devenu un outil important en reconnaissance de la parole.

6.3.3 Codage homomorphique (cepstres).

La notion de *cepstre*, anagramme de spectre, repose sur un modèle simple de production de la parole: Un système linéaire dont les paramètres varient dans le temps, représentant le conduit vocal, excité, soit par des pulsations périodiques dans le cas d'un son voisé, soit par du bruit blanc. Le son produit est donc le résultat de la convolution de l'excitation et du système linéaire.

On considère donc que le spectrogramme est le produit du spectre de l'excitation, et du spectre de transfert du conduit vocal (i.e. position de la langue, des dents, ouverture de la bouche, etc…). Or les variations dans le domaine fréquentiel de ces deux termes sont fort différentes. L'excitation présente une périodicité fréquentielle de l'ordre de la fréquence fondamentale de la voix (80 à 150 Hz). La fonction de transfert représente les résonances du conduit vocal, appelées *formants*, que l'on voit assez nettement sur un spectrogramme, à des intervalles de fréquences plus grands (500 à 1000 Hz).

On peut donc séparer ces deux informations par une transformée de Fourier partielle en fréquence, sur le spectre codé logarithmiquement. Si $c(t,n)$ est le cepstre, $sp(t,f)$ l'énergie spectrale, on a

---

1 **O'Shaugnessy D.:** *Speech Communication, human and machine* -Addison Wesley. (1987).

$$c(t,n) = \left| \int_0^{+\infty} \log(sp(t,f)) \, e^{-2i\pi fn} \, df \right| \qquad (6.6)$$

Le calcul du cepstre est en général assez couteux. Il nécessite en effet deux transformées de Fourier. Cependant les codage obtenus semblent être de bonne qualité et particulièrement intéressants pour les problèmes de reconnaissance indépendante du locuteur.

6.3.4 Quantification vectorielle.

Lorsque l'on a extrait un jeu assez réduit de paramètres, il peut être intéressant d'utiliser un algorithme de quantification vectorielle. L'idée consiste à stocker des valeurs typiques des vecteurs de paramètres du signal dans un dictionnaire, et de considérer ensuite le signal comme une suite de mots de ce dictionnaire.

On utilise donc un algorithme de clustering (par exemple “k-means”, cf §3.2.2) sur des données d'apprentissage pour déterminer une partition de l'espace vectoriel considéré. Les descriptions de chaque cluster sont alors stockées dans le dictionnaire. Pour coder une nouvelle suite de vecteurs, on remplace chaque vecteur par le numéro du cluster auquel il appartient.

Utiliser un tel codage apporte des simplifications sensibles des algorithmes de reconnaissance. Elle permettent en particulier de transformer des modèles de Markov continus en modèles de Markov discrets (cf §6.5).

6.4 La reconnaissance par alignement temporel.

Une première méthode de reconnaissance consiste à comparer le signal avec des signaux de référence stockés dans un dictionnaire. La programmation dynamique permet de calculer une distance résistant aux distorsions temporelles du signal.

Nous supposons maintenant que l'on dispose d'un jeu de paramètres réduit, mais significatif. Il faut donc transformer une suite de vecteurs, représentant le signal, en une suite de symboles, représentant des entités phonétiques.

Il faut donc choisir une *unité de segmentation*. L'idée la plus naturelle consiste à reconnaître la séquence de phonèmes prononcée. Il suffit de reconnaître une trentaine de phonèmes pour reconnaître l’ensemble des mots du français!. Cependant les phénomènes de coarticulation limitent les performances de tels systèmes: ils commettent tous près de 30% d’erreurs!

On choisit alors des unités phonétiques plus grandes, mais il faut alors en reconnaître un beaucoup plus grand nombre: quelques dizaines de milliers de mots, ou quelques milliers de syllabes ou dissyllabes [1].

6.4.1 Métriques.

Une première approche pour reconnaître un mot consiste à définir une distance permettant de comparer deux séquences de vecteurs de paramètres. Ces séquences ne sont pas nécessairement de même longueur. On utilise donc alors un algorithme qui permet de calculer une distance globale entre séquences, à partir des *distances locales* entre vecteurs de paramètres.

Lorsque l'on souhaite réaliser un dispositif rapide, on adopte souvent la *distance taxi* (distance L_1) comme distance locale, c'est à dire la somme des écarts entre composantes. Il est fréquent, également d'utiliser la *distance euclidienne* (distance L_2).

On peut en outre normaliser la distance euclidienne en tenant compte des corrélations entre composantes. On utilise alors la *distance de Mahalanobis*, définie ainsi:

$$d(X,Y) = (X-Y)^T M^{-1} (X-Y) \qquad (6.7)$$

Dans laquelle X et Y sont deux vecteurs, et M^{-1} l'inverse de la matrice de corrélations entre les vecteurs de référence. Cette distance présente l'avantage de pondérer automatiquement chaque terme. Cependant la matrice M^{-1} est difficile à évaluer correctement à partir d'un ensemble d'exemples restreint.

Dans le cas de données obtenues par *codage prédictif linéaire,* on utilise souvent une pseudo-distance dite *log-likehood-ratio* qui tire parti des mathématiques des LPC.

$$d(R,T) = \log \left(\frac{R_{\bullet}\,{}^tV R_{\bullet}}{T_{\bullet}\,{}^tV T_{\bullet}} \right) \qquad (6.8)$$

Dans laquelle V est la matrice d'autocorrélation du signal de test, définie plus haut, et la notation $R_{\bullet}$ représente un vecteur R augmenté d'un premier terme -1.

Si l'on considère que chaque référence R est en fait un système de prédiction linéaire du signal, le terme $R_{\bullet}\,{}^tV R_{\bullet}$ représente l'énergie du signal résiduel lorsque l'on essaie de prédire le signal de test avec le système R. On normalise ensuite cette erreur par l'énergie du signal résiduel obtenu avec le système de prédiction linéaire T optimal pour ce signal.

---

1 **Decker M., Gauvain J.L., Mariani J.:** *Reconnaissance de mots isolés par diphonèmes enchaînés* - 14èmes J.E.P. Paris, (1985)

6.4.2 Programmation dynamique.

Une fois définie une distance locale, il reste à calculer une distance globale, entre séquences de longueurs souvent différentes. L'algorithme appelé *programmation dynamique* (DP) ou *dynamic time warping* (DTW) est fréquemment utilisé: il permet de comparer efficacement des mots de tailles différentes en cherchant un chemin d'appariement optimal [1] [2] .

Un mot conserve son sens même après certaines distorsions temporelles. La programmation dynamique appliquée à la reconnaissance de la parole consiste à distordre dans le temps les mots de façon à les faire coincider au mieux. Soit deux mots

$$R = (r_1 \ldots r_J) \text{ et } T = (t_1 \ldots t_I) \text{ avec } r_i \text{ et } t_j \text{ vecteurs de } R^p. \qquad (6.9)$$

L'ajustement optimal entre ces deux mots est représenté par un chemin croissant dans la matrice $[1,I]x[1,J]$. A chaque point (i,j) du chemin C est associée une distance locale $d(t_i, r_j)$. Le coût du chemin C est donc la somme de toutes ces distances locales sur tous les points du chemin. On cherche donc un chemin de coût optimal, tel que

$$C_1 = (1, 1) , \; C_L = (I,J), \qquad (L \text{ est la longueur du chemin }) \qquad (6.10)$$

ne comportant que des transitions bien définies, dites *équations locales*, par exemple:

$$\{ (i-1,j), (i,j-1), (i-1,j-1) \} \rightarrow (i,j) \qquad (6.11)$$

Le calcul du chemin optimal peut se faire récursivement en constatant que tout les sous-chemins du chemin optimal sont aussi optimaux. Soit $g(i,j)$ le coût du chemin optimal reliant $(1,1)$ à (i,j). On a alors, avec les équations locales (6.11):

$$g(i,j) = d(t_i, r_j) + \min \{ g(i-1,j), g(i,j-1), g(i-1,j-1) \} \qquad (6.12)$$

On peut alors calculer récursivement tous les $g(i,j)$ donc $g(I,J)$ qui constitue une mesure du coût de l'appariement de R et T. Pour des raisons d'efficacité, on ne calcule en général les $g(i,j)$ que colonne par colonne, et en se limitant aux portions de la matrice $[1,I]x[1,J]$ situées dans le voisinage de la diagonale principale.

---

1 **Gauvain J.L., Mariani J., Liénard J.S.:** *On the use of time compression for word-based speech recognition* - IEEE - ICASSP - Boston, (1983)

2 **Mariani J., Prouts B., Gauvain J.L., Gangolf J.T.:** *Man Machine Speech Communication Systems, including word-based recognition and text to speech synthesis* - IFIP World Computer Congress, Paris, (1983).

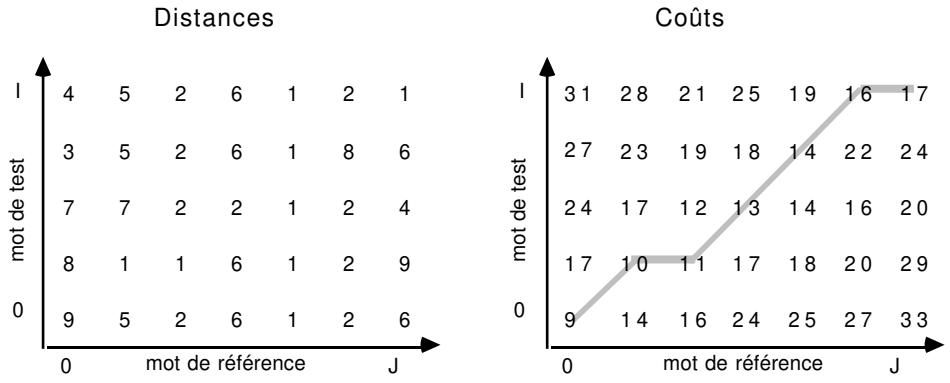


Fig 6.2-Les coûts $g(i,j)$, à droite sont calculés à partir du tableau des distances, à l'aide des équations locales décrites plus haut. En gris, le chemin optimal d'appariement.

6.4.3 Parole continue.

La programmation dynamique s'applique également au cas de la parole continue. Les équations locales, c'est à dire les transitions autorisées le long du chemin, peuvent en effet être différentes en divers points de la matrice de programmation dynamique. Il suffit alors de concaténer toutes les séquences de référence, et d'autoriser des transitions spéciales reliant la fin d'un mot au début d'un autre.

Etant donné une séquence de test de longueur quelconque, l'algorithme de programmation dynamique va chercher un chemin optimal d'appariement entre la séquence de test, et l'ensemble des mots de référence. Chaque transition spéciale marquera en fait une limite de mot.

On peut également, ce qui améliore sensiblement les performances, n'autoriser que les transitions spéciales qui correspondent à une grammaire bien définie.

Il faut cependant remarquer qu'il est nécessaire de connaître la segmentation exacte de tous les mots de référence pour pouvoir appliquer cette procédure. Il n'existe actuellement aucun moyen automatique de segmentation de la parole, autre que la reconnaissance des mots.

Le plus souvent, les données de référence sont segmentées à la main, ce qui requiert une certaine expérience. Cette opération est donc extrêmement coûteuse; il est donc souhaitable de disposer d'un système capable d'apprendre à partir de la parole continue.

6.4.4 Indépendance par rapport au locuteur.

L'indépendance par rapport au locuteur semble au premier abord n'être qu'une question de définition des tests de performances d'un système. Or la variabilité du signal de parole entre deux locuteurs peut être considérable. Il s'avère presque impossible de réaliser un système indépendant du locuteur, si l'on ne dispose pour l'entraîner que de données provenant de moins d'une quinzaine de locuteurs. Il est souvent

nécessaire de disposer de données *segmentées* concernant une soixantaine de locuteurs, prononçant chacun plusieurs fois l'ensemble des mots à reconnaître.

D'autre part, stocker une référence par mot et par locuteur prendrait un espace mémoire fort important. Des techniques spécifiques au problème d'indépendance par rapport au locuteur ont donc été développées. Un système typique indépendant du locuteur utilisera une douzaine de références, obtenues à partir des prononciations d'une cinquantaine de locuteurs. Il a donc été nécessaire d'une part de regrouper les locuteurs en une douzaine de groupes, d'autre part d'extraire une référence moyenne pour chaque mot, prononcé par les locuteurs d'un même groupe.

Nous avons vu que les techniques de reconnaissance permettent usuellement de mesurer la similitude entre deux prononciations d'un même mot. En effectuant une moyenne sur tous les mots, on peut obtenir une mesure de la distance qui sépare deux locuteurs. Cette topologie permet à des algorithmes classiques de clustering de déterminer un nombre arbitraire de groupes de locuteurs similaires.

Dans le cas de la DP, une technique simple permet de calculer des références moyennes. Elle consiste à prendre les prononciations du même mot deux par deux, et à effectuer un moyennage le long de leur chemin d'appariement optimal. Les mots étant en général de longueur différente, la référence moyenne aura la longueur du plus long d'entre eux. Il est possible, au prix de quelques complexités additionnelles, d'obtenir une référence de longueur intermédiaire.

L'expérience montre qu'un système indépendant du locuteur commet 3 à 5 fois plus d'erreurs que le même système fonctionnant en monolocuteur.

6.5 Modèles statistiques: HMM.

Une seconde approche consiste à utiliser des modèles statistiques paramétriques de production du signal, les modèles de Markov cachés. Le principe du maximum de vraisemblance (cf §3.4.) permet alors d'établir les formules de Baum, qui décrivent un algorithme itératif de réestimation des paramètres.

6.5.1 Une approche paramétrique.

Une autre approche pour la reconnaissance de la parole, les modèles de Markov cachés (Hidden Markov Models, HMM), connaît un grand succès depuis quelques années.

Les modèles de Markov Cachés entrent dans le cadre des approches paramétriques de la reconnaissance des formes, (cf §3.4.3). Il s'agit ici de restaurer la densité de probabilité de séquences de symboles (mots, phonèmes, etc…) sur l'espace des séquences de vecteurs de paramètres.

On se donne donc un *modèle paramétrique de production de séquences* aléatoires, dont la forme constitue une *hypothèse* sur la nature du signal. Ayant un certain nombre d'exemples de séquences, on se propose d'évaluer les paramètres de ce modèle.

6.5.2 Source de Markov.

Le point clef de l'approche décrite ci dessus est évidemment la nature du modèle paramétrique de production de séquences. On utilise en général des *sources de Markov*, dont il existe maintes variantes. Dans cette section, nous suivrons la présentation de Liporace [1, 2].

On se donne une chaîne de Markov, c'est à dire un ensemble d'états et une matrice de probabilités de transitions entre ces états: A chaque instant, cette "machine" saute de l'état courant vers un nouvel état, avec une probabilité déduite de la matrice de transitions. On suppose en outre qu'à chaque instant également, un signal élémentaire (on dit une *observation*) est émis par l'état courant, selon une loi de probabilité paramètrée qui lui est propre.

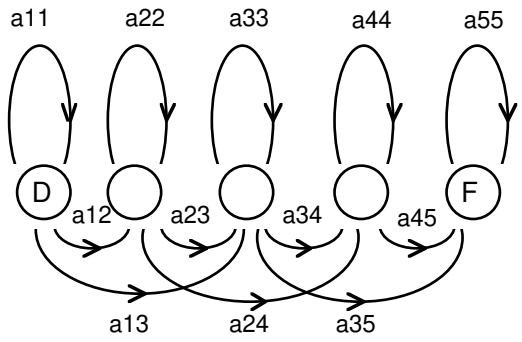


Fig 6.3 - Représentation habituelle d'une source de Markov. Les cinq cercles représentent les états, les flèches, les transitions de probabilité a_{ij} non nulle.

Une telle machine a pour effet de produire une séquence aléatoire d'observations, et constitue de fait un modèle de production de séquence, paramètrée par *la matrice de probabilités de transitions* d'une part, et *les paramètres des lois d'émission d'observations* d'autre part.

---

1 **Liporace Louis.A.**. *Maximum Likelihood Estimation for Multivariate Observations of Markov Sources*, IEEE Trans. on Information Theory, vol IT-28, n°5 (1982)

2 **Liporace Louis A.** *PTAH on continuous multivariate functions of Markov chains* , IDA-CRD report N° 80193 (1976)

6.5.2.1 Définitions.

Soient:

Ω L'*espace des observations*, c'est à dire l'espace mesurable dans lequel les séquences à modéliser prennent leur valeurs. Dans le cas de la parole, cela peut être, par exemple, l'espace $\mathbf{R}^n$ des vecteurs de paramètres issus d'un banc de filtres.

O Une *séquence d'observations*, c'est à dire un T-uplet $(O_1 \ldots O_T) \in \Omega^T$.

On appelle *source de Markov* la donnée de

$S=\{s_t\}$ Un ensemble d'états,

$\mathbf{A_0}= [a_i]$ Le vecteur des probabilités initiales des états, que l'on suppose a priori connues. Leur somme doit être 1. Signalons un cas particulier important: Définir $a_{i_0}=1$, $a_i=0 \ \forall i \neq i_0$ est équivalent à définir un état initial i_0.

$\mathbf{A}=[a_{ij}]$ La matrice des probabilités de transition entre états. a_{ij} est la probabilité de transition de l'état $i \in S$ vers l'état $j \in S$. On peut bien sûr imposer des contraintes sur cette matrice, en imposant une valeur nulle à certains de ses termes. Cela revient à autoriser ou interdire certaines transitions.

L'égalité suivante est toujours vérifiée: $\sum_{j \in S} a_{ij} = 1$

b_i Un ensemble de lois sur Ω, associées aux états $i \in S$. Dans la suite, la notation $b_i(O_t)$ représentera, soit dans le cas d'un espace Ω discret, la probabilité de O_t, soit dans le cas d'un espace Ω continu la densité de la loi b_i au point O_t.

6.5.2.2 Remarques sur la loi d'émission.

Dans certains cas, la loi b_i est associée à chaque transition (transition emitter model vs state emitter model). Cela conduit à des notations un peu plus plus lourdes, sans changer significativement le sens des démonstrations.

Un autre point délicat concerne la nature de ces lois. On souhaite un modèle paramétré: Dans le cas discret, cette loi peut être donnée exhaustivement. Dans le cas continu, on considère généralement des lois normales ou des sommes de lois normales, comme ci dessous, où $\mathbf{m}_i$ et Σ_i sont respectivement le vecteur moyen et la matrice définie positive des covariances de la loi normale.

$$b_i(O_t) = \frac{1}{2\ (2\Pi)^{\frac{n}{2}}\ \sqrt{Det(\Sigma_i)}}\ e^{-\frac{(O_t-\mathbf{m}_i)^t\ \Sigma_i^{-1}\ (O_t-\mathbf{m}_i)}{2}} \qquad (6.13)$$

6.5.2.3 Hypothèses implicites.

La définition des sources de Markov constitue une hypothèse sur la nature du signal. Outre l'arbitraire absolu de la définition des probabilités d'emission, on peut identifier deux conséquences de cette définition:

- La probabilité a priori d'un état ne dépend que de l'état précédent: c'est l'hypothèse de Markov: $P(\ s_t \mid s_{t-1}\ ) = a_{s_t,s_{t-1}}$

- Les observations ne dépendent que de l'état courant: Toute la dépendance entre deux observations à deux instants donnés est exprimée au travers de la dépendance entre les états correspondants.

Ces hypothèses sont évidemment critiquables. Il semble très difficile (cf. §4.2) d'estimer une densité sans effectuer d'hypothèses a priori, mais il reste possible d'utiliser des hypothèses moins contraignantes. Il faut alors s'attendre (cf. §4.4) à devoir utiliser plus d'exemples de séquences pour déterminer les paramètres avec une même qualité.

6.5.3 Détermination des paramètres.

On dispose maintenant de séquences d'observations aléatoires O^k, et l'on cherche à estimer les meilleurs paramètres $\lambda = \{\lambda_i\}$ de la source de Markov, c'est à dire ceux qui conduisent à la détermination conjointe des meilleures densités dans les Ω^T des séquence d'observations de longueur T.

Pour appliquer le *principe du maximum de vraisemblance*, il faut écrire la vraisemblance d'une observation dans le modèle décrit ci-dessus.

6.5.3.1 Vraisemblance d'une observation.

Grâce aux hypothèses d'indépendance entre les observations lorsque la séquence d'états est fixée, on peut écrire la vraisemblance *dans notre modèle* d'une séquence d'observation $(O_1 \ldots O_T)$, *sachant la séquence d'états* $(s_1 \ldots s_T)$:

$$L_\lambda\ (\ O_1 \ldots O_T \mid s_1 \ldots s_T\ ) = \prod_{t=1}^{T} b_{s_t}(O_t) \qquad (6.14)$$

donc, en définissant pour plus de commodité $a_{s_0 s_t} = a_{s_t}$

$$L\lambda\,(\,O_1\ldots O_T\,,\,s_1\ldots s_T\,) = P(s_1\ldots s_T\,)\,L\lambda(\,O_1\ldots O_T \mid s_1\ldots s_T\,)$$

$$L\lambda\,(\,O_1\ldots O_T\,,\,s_1\ldots s_T\,) = \prod_{t=1}^{T} a_{s_{t-1}s_t}\; b_{s_t}(O_t) \qquad (6.15)$$

On peut alors écrire la vraisemblance d'une observation comme étant la somme sur toutes les séquences d'états des $L\lambda\,(\,O_1\ldots O_T\,,\,s_1\ldots s_T\,)$:

$$L\lambda\,(\,O_1\ldots O_T\,) = \sum_{(s_1\ldots s_T)\in\,\Gamma} L\lambda(\,O_1\ldots O_T\,,\,s_1\ldots s_T\,)$$

$$L\lambda\,(\,O_1\ldots O_T\,) = \sum_{(s_1\ldots s_T)\in\,\Gamma}\;\prod_{t=1}^{T} a_{s_{t-1}s_t}\; b_{s_t}(O_t) \qquad (6.16)$$

en notant Γ l'ensemble de toutes les séquences d'états.

Il nous faut donc déterminer des paramètres λ qui maximisent la vraisemblance des séquences d'observation de référence, tout en respectant les contraintes imposées par la nature probabiliste de ces mêmes paramètres. Mais avant tout, il nous faut pouvoir calculer cette vraisemblance.

6.5.3.2 Calcul récursif de la vraisemblance.

Les hypothèses régissant les sources de Markov donnent aux vraisemblances ont une propriété remarquable: Il en effet possible de calculer très simplement $L\lambda\,(\,O_1\ldots O_T\,)$ de façon récursive.

$$\begin{aligned} L\lambda\,(\,O_1\ldots O_T\,) &= \sum_{j\in S} L\lambda(\,O_1\ldots O_T \mid s_t=j\,) \\ &= \sum_{j\in S} L\lambda(\,O_1\ldots O_t,\, s_t=j\,)\; L\lambda(O_{t+1}\ldots O_T \mid O_1\ldots O_t,\, s_t=j\,) \end{aligned}$$

Les hypothèses sur les sources de Markov spécifient que $(O_{t+1}\ldots O_T)$ ne dépend de $(O_1\ldots O_t)$ qu'au travers de l'état s_t. On peut donc écrire:

$$L\lambda\,(\,O_1\ldots O_T\,) = \sum_{j\in S} L\lambda(\,O_1\ldots O_t,\, s_t=j\,)\; L\lambda(\,O_{t+1}\ldots O_T \mid s_t=j\,)$$

$$\boxed{L\lambda\,(\,O_1\ldots O_T\,) \stackrel{\Delta}{=} \sum_{j\in S}\alpha_t(j)\;\beta_t(j)\;,\quad \forall\; t\in[1\ldots T]} \qquad (6.17)$$

On peut alors calculer $\alpha_t(j)$ et $\beta_t(j)$ de la façon suivante:

$$\begin{aligned} \alpha_t(j) &= L\lambda(\,O_1\ldots O_t,\, s_t=j\,) \\ &= \sum_{i\in S} L\lambda(\,O_1\ldots O_t,\, s_t=j,\, s_{t-1}=i) \end{aligned}$$

$$= \sum_{i \in S} L_\lambda(\, O_1 \ldots O_{t-1},\, s_{t-1}=i)\; L_\lambda(\, O_t,\, s_t=j \mid O_1 \ldots O_{t-1},\, s_{t-1}=i)$$

$$\begin{aligned} \alpha_t(j) &= \sum_{i \in S} \alpha_{t-1}(i)\; L_\lambda(\, O_t,\, s_t=j \mid s_{t-1}=i) = \sum_{i \in S} \alpha_{t-1}(i)\; b_j(O_t)\; a_{ij} \\ \alpha_0(j) &= a_j \quad \textit{par définition}. \end{aligned} \tag{6.18}$$

de même, on peut établir la formule:

$$\begin{aligned} \beta_t(j) &= \sum_{k \in S} \beta_{t+1}(i)\; L_\lambda(\, O_{t+1},\, s_{t+1}=k \mid s_t=j) = \sum_{k \in S} \beta_{t+1}(i)\; b_k(O_{t+1})\; a_{jk} \\ \beta_T(j) &= 1 \quad \textit{par définition}. \end{aligned} \tag{6.19}$$

Les formules (6.17) (6.18) et (6.19) nous permettent donc de calculer de façon algorithmiquement efficace la vraisemblance d'une observation.

6.5.3.3 Réestimation des paramètres

L'expression exacte de $L_\lambda\,(\, O_1 \ldots O_T\,)$ est malheureusement bien trop compliquée pour pouvoir envisager une résolution analytique du problème de maximisation de la vraisemblance.

Connaissant un jeu de paramètres λ, on va chercher un nouveau jeu de paramètres λ', tel que $L_{\lambda'}(O_1 \ldots O_T) > L_\lambda(O_1 \ldots O_T)$, à l'aide d'une astuce classique, qui fonctionne sur tous les polynômes à coefficients positifs.

Pour ce faire, on définit la fonction auxiliaire:

$$Q(\lambda,\lambda') = \sum_{(s_1 \ldots s_T) \in \Gamma} L_\lambda(O_1 \ldots O_T,\, s_1 \ldots s_T)\, \log\{L_{\lambda'}(O_1 \ldots O_T,\, s_1 \ldots s_T)\} \tag{6.20}$$

Grâce à l'inégalité $\log(x) < x - 1$ et à la positivité des probabilités dans (3), on a:

$$Q(\lambda,\lambda') - Q(\lambda,\lambda) \le L_{\lambda'}(O_1 \ldots O_T) - L_\lambda(O_1 \ldots O_T) \tag{6.21}$$

Ce qui implique qu'optimiser $Q(\lambda,\bullet)$ garantit que $L_{\lambda'}(O_1 \ldots O_T) > L_\lambda(O_1 \ldots O_T)$. Si on remplace (6.15) dans (6.20), les paramètres $\log(a'_{ij})$, $\log(a'_i)$ et $\log(b'_i(O_t))$ apparaissent de façon linéaire:

$$Q(\lambda,\lambda') = \sum_{(s_1 \ldots s_T) \in \Gamma} L_\lambda(O_1 \ldots O_T, s_1 \ldots s_T) \quad \Big(\sum_{t=1}^{T} \log(a'_{s_{t-1} s_t}) + \log(b'_{s_t}(O_t)) \Big)$$

On peut alors *calculer analytiquement l'optimum sous contraintes* de $Q(\lambda,\bullet)$ en écrivant la dérivée du Lagrangien:

$$\nabla_{\lambda',\mu} \left\{ Q(\lambda,\lambda') - \sum \mu_i C_i(\lambda') \right\} = 0 \qquad (6.22)$$

dans laquelle les $C_i(\lambda')$ représentent les contraintes sur les paramètres. La résolution de l'équation conduit aux *formules de Baum* [1], ou *algorithme de Baum-Welch*. Ces calculs assez lourds ne seront pas plus détaillés maintenant. (cf §9.1, pour une autre dérivation, plus générale, des formules de Baum).

i Formules de Baum dans le cas discret.

Dans le *cas discret*, les paramètres sont les probabilités de transition a_{jk} et les probabilités d'émission $b_j(\xi_n)$, où les ξ_n sont les observations quantifiées. Les formules de réestimation correspondantes sont:

$$a'_{jk} = \frac{\sum_{t=1}^{T} \alpha_{t-1}(j)\, a_{jk}\, b_k(O_t)\, \beta_t(k)}{\sum_{t=1}^{T} \alpha_{t-1}(j)\, \beta_{t-1}(j)} \qquad b'_i(\xi_n) = \frac{\sum_{O_t=\xi_n} \alpha_t(i)\, \beta_t(i)}{\sum_{j \in S} \alpha_t(j)\, \beta_t(j)} \qquad (6.23)$$

ii Formules de Baum dans le cas continu.

Dans le cas gaussien, les probabilités d'émission sont modélisées par des lois normales (6.13). On obtient alors aux formules ci dessous. Les a'_{jk} sont les probabilités de transition, les M'_i les centres des gaussiennes, et les Σ'_i les matrices de covariance des gaussiennes. On note X^T le vecteur transposé de X.

$$a'_{jk} = \frac{\sum_{t=1}^{T} \alpha_{t-1}(j)\, a_{jk}\, b_k(O_t)\, \beta_t(k)}{\sum_{t=1}^{T} \alpha_{t-1}(j)\, \beta_{t-1}(j)} \qquad \mathbf{m}'_i = \frac{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)\, O_t}{\sum_{t=1}^{T} \alpha_t(i)\, \beta_t(i)}$$

$$\Sigma'_i = \frac{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)\, O_t\, O_t^T}{\sum_{t=1}^{T} \alpha_t(i)\, \beta_t(i)} - \mathbf{m}'_i\, \mathbf{m}'^T_i \qquad (6.24)$$

---

1 **Baum L. E., Petrie T., Soules G., Weiss N.**: *A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov chains*, Annals of mathematic statistics, vol41, n°1, pp 164-171 (1970)

6.5.4 Convergence.

6.5.4.1 Propriétés de convergence.

On ne peut assurer que la *convergence vers un maximum local* de la vraisemblance: Aucune garantie n'est donnée de trouver un maximum global. La pratique montre qu'en général, l'algorithme converge vers une solution acceptable.

Cette convergence appelle deux remarques:

- On dispose en pratique de plusieurs séquences d'observations $(O^k_1 \dots O^k_T)$ pour entraîner notre modèle. Le principe du maximum de vraisemblance consiste donc (cf. §3.4.3) à rechercher le maximum de

$$\prod_k L_\lambda(O^k_1 \dots O^k_T) = \prod_k \sum_{(s_1 \dots s_T) \in \Gamma} L_\lambda(\, O_1 \dots O_T \,,\, s_1 \dots s_T \,)$$

  Il faut alors développer ce polynôme, formuler la fonction auxiliaire, et annuler les dérivées de son Lagrangien. Les réestimations correspondantes (cf §9.1) sont comparables aux formules de Baum, mais une sommation supplémentaire sur les exemples, pondérée par les inverses de leurs vraisemblances, se glisse aux numérateurs et dénominateurs.

- Jusqu'à présent, rien ne nous assure que l'optimum de la fonction auxiliaire apporte, à chaque réestimation, une amélioration importante de la vraisemblance. La minoration des accroissements de la vraisemblance,

$$Q(\lambda,\lambda') - Q(\lambda,\lambda) \leq L_{\lambda'}(O_1 \dots O_T) - L_\lambda(O_1 \dots O_T),$$

  pourrait être extrêmement grossière. Il n'en est rien, comme on le montrera dans la section 9.1.

6.5.4.2 Problèmes numériques.

Cet algorithme pose également d'important problèmes numériques. Le calcul des $\alpha_t(i)$ et des $\beta_t(i)$ consiste essentiellement à multiplier entre elles des probabilités inférieures à 1.

Toutes ces valeurs tendent donc vers zéro très vite. Les formules de Baum deviennent alors des quotients de la forme $0/0$. Une méthode pour s'affranchir de ces problèmes numériques est proposée par Levinson dans [1]. On pose, la suite (N_i) étant donnée:

---

1 **Levinson S.E., Rabiner L.R. Sondhi M.M.:** *An introduction to the application of the theory of probabilistic functions of a Markov process to Automatic Speech Recognition* - Bell Systems Technical Journal Vol62, n°4, pp 1035-1074 (april 1983)

$$\alpha'_t(j) = C_t\,\alpha_t(j) \quad \text{et} \quad \beta'_t(j) = D_t\,\beta_t(j) \quad \text{avec} \quad C_t = \prod_{i=1}^{t} N_i \ \text{et} \ D_t = \prod_{i=t+1}^{T} N_i$$

En reprenant les équations (6.18) et (6.19), on obtient:

$$\alpha'_t(j) = \frac{C_t}{C_{t-1}} \sum_{i \in S} \alpha'_{t-1}(i) b_j(O_t) a_{ij} = N_t \sum_{i \in S} \alpha'_{t-1}(i) b_j(O_t) a_{ij}$$

$$\beta'_t(j) = \frac{D_t}{D_{t+1}} \sum_{k \in S} \beta'_{t+1}(i) b_k(O_{t+1}) a_{jk} = N_t \sum_{k \in S} \beta'_{t+1}(i) b_k(O_{t+1}) a_{jk} \qquad (6.25)$$

Ces formules de récurrence permettent de calculer les $\alpha'_t(j)$ et les $\beta'_t(j)$. On détermine alors les N_t au cours de la passe avant, de telle sorte que: $\sum_{j \in S} \alpha'_t(j) = 1$.

De plus, on peut faire apparaître les coefficients C_t et D_t dans les formules de Baum, en multipliant numérateur et dénominateur par $C_t.D_t=C_T=D_1$. On obtient ainsi des formules de réestimation équivalentes aux formules de Baum, dans lesquelles les $\alpha'_t(j)$ et $\beta'_t(j)$ remplacent formellement les $\alpha_t(j)$ et $\beta_t(j)$.

6.6 HMMs pour la reconnaissance de la parole.

Une abondante littérature traite de l'application des modèles de Markov aux problèmes de reconnaissance de la parole. Citons [1,2,3] aux Etats-Unis, et [4,5] en Europe. De nombreux développements ont été effectués. En aucun cas ces quelques lignes peuvent prétendre à l'exhaustivité. La seule ambition de cette section est de donner au lecteur les quelques informations nécessaires pour la suite de l'exposé.

---

1 **Jelinek F.:** *Continuous Speech Recognition by Statistical Methods* - Proc. IEEE vol64, n°4, pp 532-556 (april 1976)

2 **Bahl L.R., Jelinek F., Mercer R.L.:** *A maximum likelihood Approach to Continuous Speech Recognition* - IEEE Trans. PAMI, vol5, n°2, pp 179-190 (march 1983)

3 **Bahl L.R., Brown P.F., de Souza P.V., Mercer R.L.:** *Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition* -Proc. IEEE ICASSP 86, pp 49-52, Tokyo, (1986)

4 **Derouault A.M.,** *Context Dependent Phonetic Markov Models for Large Vocabulary Speech Recognition* - Proc. IEEE ICASSP 1987, pp 360-363 (1987)

5 **Jouvet D.:** *Reconnaissance de mots connectés indépendamment du locuteur par des méthodes statistiques* - Thèse de Doctorat ENST-88E006 (juin 1988)

6.6.1 Règle Bayesienne.

Posons maintenant un problème de reconnaissance, par exemple, de mots isolés $\{\omega_1 \ldots \omega_N\}$. Soient $W_1 \ldots W_N$ des modèles de Markov censés représenter chacun des mots ω_i. Dans chacun de ces modèles, on souhaite définir un état initial $s_I(W_i)$ et un état final $s_F(W_i)$.

Pour définir un état initial et un état final, il suffit, dans l'équation (6.16), de limiter Γ, ensemble de toutes les séquences d'états, aux seules séquences commençant par $s_I(W_i)$, et finissant par $s_F(W_i)$.

On remarque en effet que limiter l'ensemble Γ ne change pas la forme des équations développées dans la section précédente! Il suffit de veiller à effectuer le calcul récursif des $\alpha_t(i)$ et des $\beta_t(i)$ le long seulement des transitions autorisées dans Γ. La définition d'un état initial impose en outre que les probabilités initiales aient la forme $a_i(W_k) = \delta(i, s_I(W_k))$.

i Utilisation du théorème de Bayes.

On a vu qu'une source de Markov est un modèle de production de signal. Ils peuvent cependant facilement être utilisés pour la reconnaissance, en utilisant le théorème de Bayes.

Si l'on dispose d'une séquence d'observations $(O_1 \ldots O_T)$, correspondant à un mots inconnu de l'ensemble $\{\omega_1 \ldots \omega_N\}$. On peut approcher la probabilité conditionnelle de chaque mot ω_i par $P(W_i \mid O_1 \ldots O_T)$. D'après la formule de Bayes (sur les densités!):

$$P(\omega_i \mid O_1 \ldots O_T) \approx P(W_i \mid O_1 \ldots O_T) = \frac{L(O_1 \ldots O_T \mid W_i)\, P(W_i)}{L(O_1 \ldots O_T)} \qquad (6.26)$$

Pour une séquence d'observation fixée, le dénominateur de cette expression est constant. Le mot reconnu est donc:

$$\omega^* = \arg\max_{\{\omega_i\}} \left(L(O_1 \ldots O_T \mid W_i)\, P(W_i) \right) \qquad (6.27)$$

ii L'algorithme Forward.

Dans un premier temps, il faut estimer la grandeur $L(O_1 \ldots O_T \mid W_i)$. En appliquant (6.17) et (6.18) pour notre modèle W_i, et en tenant compte des contraintes sur les chemins admissibles, on obtient:

$$L(O_1 \ldots O_T \mid W_k) = \alpha_T(s_F(W_k))$$

$$\alpha_0(s_I(W_k)) = 1$$

$$\alpha_t(j) = \sum_{i \in S} \alpha_{t-1}(i)\, L(O_t, s_t = j \mid s_{t-1} = i) = \sum_{i \in S} \alpha_{t-1}(i)\, b_j(O_t)\, a_{ij} \qquad (6.28)$$

Cet algorithme est appelé *algorithme forward*. En effet, le calcul récursif de $\alpha_t(j)$ se fait à l'aide de ces mêmes valeurs à l'instant précédent, c'est à dire, dans le sens du temps.

iii Introduction d'un modèle de langage.

Reste à estimer $P(W_i)$. Le modèle W_i représentant le mot ω_i, on évalue cette probabilité en se donnant un *modèle de langage*, c'est à dire en déterminant empiriquement la probabilité d'occurence de chaque mot.

$$P(W_i) \approx P(\omega_i)$$

ce qui donne:

$$\omega^* \approx \arg\max_{\{\omega_i\}} (L(O_1 \ldots O_T \mid W_i) P(\omega_i)) \qquad (6.29)$$

Notons toutefois que la confusion entre la *probabilité du modèle* et la *probabilité du mot* n'est pas à l'abri de toute critique. Cette approximation n'est valide que si les modèles représentent les mots avec précision.

Si, par exemple, un modèle comportant peu d'état est utilisé pour représenter un mot long, comme notre fameux "anticonstitutionnellement", ce modèle va se révéler peu sélectif, et fournira un valeur identique de $L(O_1 \ldots O_T \mid W_k)$ pour à peu près tous les mots longs. $P(W_k)$ sera alors bien supérieur à $P(\omega_k)$. De toute façon, il y a bien d'autres raisons pour qu'un tel système ne fonctionne pas!

6.6.2 Cas de la parole continue.

6.6.2.1 Super modèle.

Afin d'aborder le traitement de la parole continue, on construit un grand modèle de Markov, en ajoutant aux diagrammes quelques transitions liant tous les états finaux à tous les états initiaux.

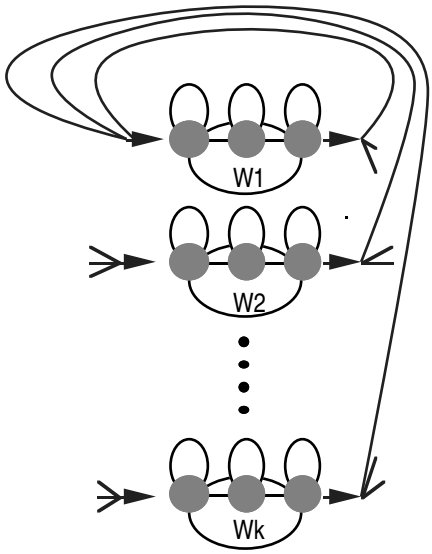


Fig 6.4 - Etablissement de transitions reliant les états finaux aux états initiaux. (Toutes les transitions n'on pas été représentées)

Notons que ce modèle a alors plusieurs états initiaux, et plusieurs états finaux. Cela ne pose pas de problèmes pratiques, il suffit comme précédemment d'inclure ces contraintes d'une part dans l'ensemble Γ des chemins admissibles, d'autre part dans la distribution de probabilités initiales $[a_i]$ des états.

Dans un premier temps, nous supposerons que les paramètres des divers modèles ont déjà été déterminés en appliquant l'algorithme de Baum. Reste à choisir les probabilités des nouvelles transitions liant entre-eux les mots. On peut soit rendre ces transitions équiprobables, soit en fixer les probabilités en s'appuyant sur des connaissances grammaticales. On définit ainsi une *grammaire probabiliste*.

6.6.2.2 Algorithme de Viterbi.

i Définition.

L'algorithme de Viterbi consiste à calculer la *vraisemblance du signal le long du chemin le plus probable*. C'est à dire:

$$L^{Viterbi}(O_1 \dots O_T) = \underset{(s_1 \dots s_T) \in \Gamma_T}{Max} \{ L(O_1 \dots O_T, s_1 \dots s_T) \} \qquad (6.30)$$

Comme dans le cas de l'algorithme Forward, on peut effectuer ce calcul récursivement, dans le sens du temps. Il suffit en fait de remplacer tous les opérateurs de sommation (Σ) en opérateurs de maximum (Max). Cela découle de l'associativité de ces opérateurs, et de la distributivité de la multiplication à leur égard.

$$L^{Viterbi}(O_1 \ldots O_T \mid W_k) = \alpha^v_T(s_F(W_k))$$

$$\alpha^v_0(s_I(W_k)) = 1$$

$$\alpha^v_t(j) = \max_{i \in S} \{\alpha^v_{t-1}(i)\, L(O_t, s_t=j \mid s_{t-1}=i)\} = \max_{i \in S} \{\alpha^v_{t-1}(i)\, b_j(O_t)\, a_{ij}\} \qquad (6.31)$$

Comparé avec l'algorithme Forward, l'algorithme de Viterbi possède deux qualités majeures, et un défaut important.

- On peut retrouver explicitement la meilleure séquence d'états. Il suffit de se rappeler des arguments de l'opérateur Max qui ont été sélectionnés. Ce point est crucial pour le traitement de la parole continue.

- L'algorithme de Viterbi se prête à de multiples simplifications. On peut pratiquer par exemple *l'élagage* de séquences d'états déjà fort improbables à un instant donné. On peut également pratiquer un codage logarithmique des probabilités, et changer les produits en somme.

- Mais il faut se rappeler que *l'algorithme Forward et l'algorithme de Viterbi ne sont pas équivalents*. En toute rigueur, il faudrait appliquer Forward. Rien ne prouve en effet que la séquence d'états la plus probable fournit une bonne indication sur la séquence de mots la plus probable.

ii Utilisation de l'algorithme de Viterbi.

Soit un signal $(O_1 \ldots O_T)$. L'algorithme de Viterbi permet de trouver la *séquence d'état* qui modélise le plus vraisemblablement le signal. Cette *séquence d'états traverse une suite de modèles* $(W_1 \ldots W_\kappa)$. On dit alors que la séquence de mots $(\omega_1 \ldots \omega_\kappa)$ est reconnue.

On peut raffiner ce procédé à l'aide de *modèles de silences*. Notre système devient alors capable de distinguer parole et silences, et de fonctionner en temps continu.

iii Utilisation d'unités phonétiques plus petites que le mot.

Un modèle de Markov représentant un mot contient usuellement de 10 à 30 états. Si on désire toutefois reconnaître un grand nombre de mots (20000 par exemple), le modèle de Markov global serait énorme: Brut, l'algorithme de Viterbi requiert un nombre d'opérations proportionnel au produit du nombre d'état et de la longueur du signal. C'est en tout état de cause beaucoup trop!

D'où l'idée de ne pas reconnaître des mots, mais des unités phonétiques plus courtes, et moins nombreuses. Par exemple: une cinquantaine de phonèmes. Cela toutefois rend plus sensibles les

problèmes de coarticulation. Plus les unités phonétiques sont courtes, plus les effets de voisinage sont proportionnellement important.

Comme dans le cas de l'alignement temporel, il y a un compromis à faire sur le nombre des unités phonétiques à reconnaître, et on peut se demander si les progrès réalisés ces dernières années ne sont pas essentiellement dus à l'augmentation considérable de la puissance de calcul disponible pour un même prix.

6.6.2.3 Apprentissage "en ligne".

Ces méthodes sont aujourd'hui utilisées dans un grand nombre de systèmes [1,2,3].

Nous avons supposé tout à l'heure que les paramètres de nos modèles étaient déjà déterminés. Cela signifie que l'on disposait d'exemples bien délimités de chaque mot, ou unité phonétiques. Une telle base de données peut être coûteuse; on désirerait en fait entraîner nos modèles à l'aide de signal non segmenté.

On dispose alors de signaux d'entraînement, composés d'une séquence de plusieurs mots ou unités phonétiques. La séquence de mots est connue, mais on ne connait pas la position des frontières entre mots.

- On peut utiliser un analogue de l'algorithme forward [4]. On constitue un modèle de Markov en concaténant dans l'ordre tous les modèles de la séquence de mots (fig 6.5). On peut alors appliquer une itération de l'algorithme de Baum-Welch, sans avoir besoin de segmentation explicite.

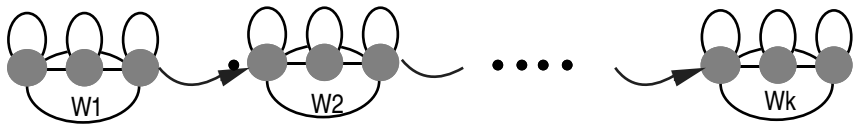


fig 6.5 - Concaténation des modèles de mots W_1 à W_k

---

1 **Kai Fu Lee:** *Large Vocabulary Speaker Independent Continuous Speech Recognition: The SPHINX system* - CMU PhD Thesis - CMU-CS -88-148 (1988)

2 **Bourlard H., Kamp Y., Wellekens C.J.:** *Speaker Dependent Connected Speech Recognition via Phonemic Markov Models* - Proc IEEE ICASSP 85, 31.5.1-4, (1985)

3 **Rabiner L.R., Wilpon J.G., Quinn A.M., Terrace S.G.:** *On the applications of embedded digit training to speaker independent connected digit recognition* - IEEE Trans. Acoustics, Speech and Signal Processing, vol32, n°2, pp 272-279 (april 1984)

4 **Jouvet D., Monné J., Dubois D.:** *A new network-based speaker independent connected word recognition system.* - IEEE ICASSP 86, pp 1109-1112, Tokyo (1986)

Il ne faut alors plus chercher à s'approcher de l'optimum global! Il est en effet vraisemblable que certains mots seraient mieux représentés par des modèles ayant plus d'états. Ils risquent de déborder sur les modèles voisins. Il faudrait toutefois franchir l'unique transition autorisée d'un mot à l'autre, qui représente un 'col' dans la fonction de coût.

- On peut également utiliser la procédure d'apprentissage de Viterbi: On effectue une segmentation grossière, par exemple linéaire, que l'on utilise pour déterminer les signaux d'entraînement de nos modèles de Markov, qui eux même peuvent fournir une nouvelle segmentation, grâce à l'algorithme de Viterbi. Cette nouvelle segmentation est alors utilisée pour réentrainer les modèles de Markov, etc…

Dans les deux cas, l'apprentissage est plus facile si les conditions initiales sont meilleures. Une bonne stratégie consiste à entraîner les modèles séparément, soit sur une petite base de données segmentées, soit sur une base de données grossièrement segmentée. L'apprentissage "en ligne" peut ensuite être utilisé pour affiner la sélection des paramètres, sur des données abondantes, mais moins coûteuses.

6.6.3 Remarques.

6.6.3.1 Capacité discriminante.

L'une des particularité de l'apprentissage de ces modèles est son aspect *peu discriminant: Chaque modèle de mot peut être entraîné séparément*. C'est extrêmement utile d'un point de vue pratique.

Cela montre aussi les limites de cet apprentissage: L'algorithme de Baum-Welch n'extrait pas des *caractéristiques permettant de discriminer deux mots*, mais essaie de faire une *approximation de la densité de probabilité du mot* dans l'espace de toutes séquences d'observation.

Si cette densité était approchée avec une grande précision, il suffirait d'appliquer la règle de décision de Bayes comme exposé ci dessus. Cependant, on ne dispose de garanties (cf. §4.2 et §4.5.3) que lorsque la densité appartient à la famille paramétrique utilisée pour la modélisation.

La modélisation Markovienne est cependant assez *robuste* pour fournir des résultats satisfaisants, tant que n'apparaissent pas des phénomènes de coarticulation. Dans le cas contraire, le signal devient ambigu, on s'approche des frontières de discrimination, et les performances se dégradent de façon importante.

6.6.3.2 Approximation de Viterbi.

Lorsque l'on utilise l'algorithme de Viterbi, on calcule la séquence d'états la plus vraisemblable (6.30), et on en déduit une séquence de mots.

$$L^{\text{Viterbi}}(O_1 \ldots O_T) = \max_{(s_1 \ldots s_T) \in \Gamma_T} L(O_1 \ldots O_T, s_1 \ldots s_T)$$

Or, rien ne prouve que cette séquence d'états traverse la séquence de modèles la plus vraisemblable. En effet, la vraisemblance d'une séquence de modèles est la somme des vraisemblances des séquences d'états traversant ces modèles.

$$L(O_1 \ldots O_T, W_1 \ldots W_N) = \sum_{\substack{(s_1 \ldots s_T) \in \Gamma_T \\ \text{traversant } W_1 \ldots W_N}} L(O_1 \ldots O_T, s_1 \ldots s_T) \qquad (6.32)$$

L'approximation de Viterbi est justifiée si on suppose que la vraisemblance d'une séquence d'état domine nettement les autres. Ce n'est pas le cas, par exemple, dans les périodes de coarticulation. Le signal y est ambigü; le chemin le plus vraisemblable est alors mal déterminé; les différences de vraisemblance entre les chemins possibles sont donc faibles.

Cependant, un algorithme exact ne fonctionnerait probablement pas mieux. En effet la faible capacité discriminante de l'algorithme de Baum lui permet rarement d'estimer les vraisemblances avec précision dans les portions ambigües du signal.

A titre d'illustration, considérons un problème de reconnaissance phonétique dans lequel chaque phonème est représenté par trois états correspondant au début, à la partie stable, et à la fin du phonème.

Plusieurs auteurs, dont [1] ont remarqué dans des problèmes similaires que les états de début et de fin possédaient une probabilité d'émission très peu sélective. Un même phonème peut en effet avoir des réalisation phonétiques très dissemblables dans des contexte différents.

La vraisemblance d'un chemin $s_1 \ldots s_T$ décrivant une séquence de deux phonèmes peut se décomposer sur trois parties: $s_1 \ldots s_{t1-1}$, correspondant à la partie stable du premier phonème, $s_{t1} \ldots s_{t2}$, correspondant à la partie coarticulée, et $s_{t2+1} \ldots s_T$ correspondant à la partie stable du second phonème.

$$\begin{aligned} L_\lambda(O_1 \ldots O_T, s_1 \ldots s_T) &= \prod_{t=1}^{T} a_{s_{t-1} s_t} \, b_{s_t}(O_t) \\ &= L(O_1 \ldots O_{t1-1}, s_1 \ldots s_{t1-1}) \, L(O_{t1} \ldots O_{t2}, s_{t1} \ldots s_{t2}) \, L(O_{t2+1} \ldots O_T, s_{t2+1} \ldots s_T) \\ &= L_{/1/} \; L_{/12/} \; L_{/2/} \end{aligned}$$

---

1 **Kai Fu Lee, Hsiao Wuen Hou:** *Speaker Independent Phone Recognition using Hidden Markov Models* - IEEE Trans. on Acoustics, Speech and Signal Processing, vol37, n°11, pp 1641-1648.

Or, nous savons que les probabilités d'émission des états initiaux et finaux composant la partie $s_{t1} \ldots s_{t2}$ sont très peu sélectives. $L_{/12/}$ est alors pratiquement indépendant du chemin $s_{t1} \ldots s_{t2}$. Ce terme peut alors être mis en facteur dans la somme de l'algorithme Forward ou dans le Max de l'algorithme de Viterbi. Seuls $L_{/1/}$ et $L_{/2/}$ contribuent donc à différencier deux séquences de phonèmes.

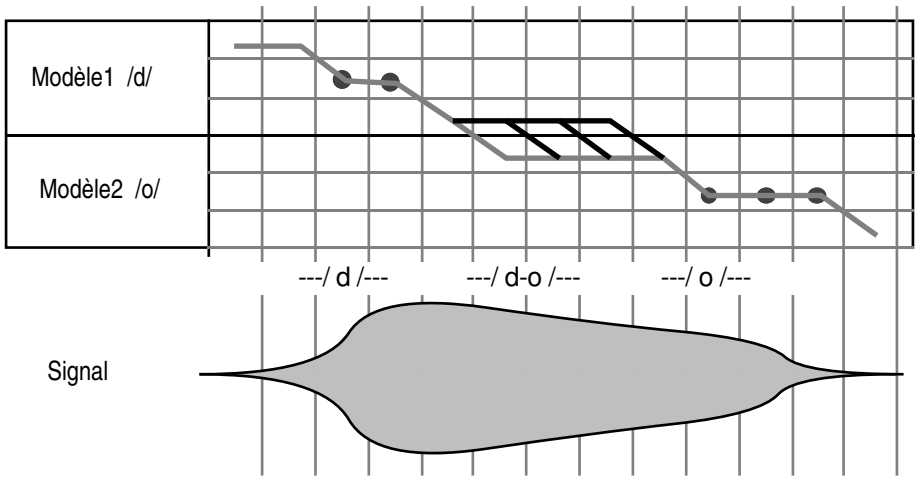


Fig 6.6 - Délocalisation des chemins pendant la reconnaissance du son /do/. Pendant la période de coarticulation, notée /d-o/, plusieurs chemins entre l'état final de /d/ et l'état initial de /o/ sont en concurrence. Mais ces états sont peu sélectifs…

Mais $L_{/1/}$ et $L_{/2/}$ correspondent à des états d'une part sélectifs, d'autre part correspondant à un signal peu ambigu. La vraisemblance des chemins autres que le plus vraisemblable peut alors y être négligée.

6.6.3.3 Algorithme de Viterbi, et Programmation Dynamique.

On a constaté maintes fois l'identité de principe entre les algorithmes d'alignement temporel (DP) et l'algorithme de Viterbi utilisé dans les modèles de Markov cachés.

- Dans le cas de l'alignement temporel, on cherche *l'appariement de coût minimal* entre des signaux de référence, et le signal. Les contraintes sur les appariements sont exprimées par les *équations locales*.

- Dans le cas de l'algorithme de Viterbi, on recherche la *séquence d'états dont la vraisemblance conditionnelle est maximale*. Les transitions d'un état à l'autre sont contraintes par *l'architecture du modèle de Markov et la valeur des probabilités de transitions*.

Ces deux problèmes se différencient surtout par le vocabulaire. Il s'agit toujours de rechercher un chemin optimal dans l'espace produit signal x références ou signal x états, en tenant compte de contraintes exprimées sur l'ensemble des chemins admissibles.

Il n'est donc pas étonnant que les algorithmes d'alignement temporel et de Viterbi ne soient que deux versions d'une même technique, la *programmation dynamique*. Elle permet en effet d'extraire un séquence optimale, soit sous la forme d'une liste de signaux de référence utilisés, soit sous la forme d'une séquence de groupes d'états.

7 Méthodes connexionnistes pour la reconnaissance de la parole.

Les qualités de résistance au bruit et d'adaptabilité des méthodes connexionnistes en font des candidats intéressants pour réduire certaines contraintes. De nombreux travaux ont été publiés dans ce domaine (cf. [1]) depuis quelques années

7.1 Architectures pour la reconnaissance de la parole.

Nous avons vu que tous les systèmes de reconnaissance de la parole reposaient sur un ensemble de contraintes plus ou moins sévères. Avec un peu d'optimisme, on peut espérer que des algorithmes d'adaptation et d'apprentissage permettront d'assouplir certaines de ces contraintes.

Les algorithmes connexionnistes, et plus particulièrement les perceptrons multi-couches, résistent remarquablement au bruit [2], et possèdent une forte capacité discriminante. Ces qualités ne suffisent

---

1 **Lippmann R.P.:** *Review of Neural Networks for Speech Recognition* - Neural Computation, vol 1, pp 1-38, (1989)

2 **Bollivier M. de, Gallinari P., Thiria S.:** *Cooperation of neural nets for robust classification*, IJCNN 90, San Diego, vol1, pp 113-120, (1990)

cependant pas: Il importe donc de ne pas sous-estimer l'acquis des méthodes "classiques", raffinées depuis une vingtaine d'années.

7.1.1 Une classification atypique.

Après les prétraitements, la tâche d'un système de reconnaissance consiste, dans un cadre très général, à extraire du signal codé une séquence de symboles représentant le message. C'est donc une tâche de reconnaissance des formes. Elle s'écarte cependant des hypothèses présentées dans la section 3.4.

- Le nombre de classes n'est pas fini. Chaque classe est en effet identifiée par une séquence de symboles virtuellement infinie, représentant par exemple une séquence de mots, syllabes ou phonèmes.

- Les formes ne sont pas éléments d'un espace vectoriel de dimension finie: Le signal est en effet codé sous la forme d'une séquence de vecteurs virtuellement infinie.

On cherche donc à identifier et localiser dans le signal les diverses unités phonétiques qui se succèdent. Les phénomènes de coarticulation compliquent singulièrement ce double problème de *classification* et de *segmentation*.

7.1.2 Reconnaissance statique.

Il existe une façon simple de s'affranchir de ces problèmes: contraindre le locuteur à insérer un bref silence entre chaque mots. Il est alors possible de déterminer les limites de ces *mots isolés*, en étudiant les variations de l'amplitude du signal.

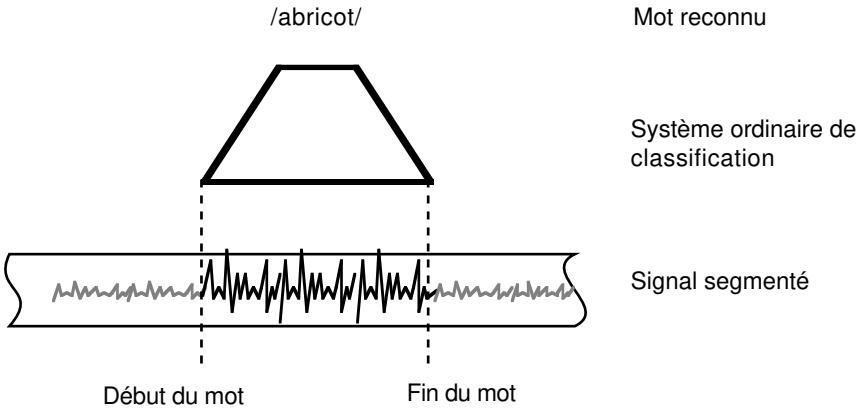


Fig 7.1 - On peut se ramener au cas standard de reconnaissance des formes si l'on dispose d'une segmentation en mots a priori.

On découpe alors des signaux durant environ une seconde, qu'il suffit de regrouper en autant de classes qu'il y a de mots dans un vocabulaire défini à l'avance (fig 7.1). On peut alors utiliser toutes les méthodes connues de classification, sans plus s'inquiéter des problèmes de distorsions temporelles.

Les premiers réseaux connexionnistes pour la reconnaissance de la parole fonctionnaient ainsi. Dans [1], Elman et Zipser utilisent un perceptron multi-couches pour discriminer six syllabes isolées: /ga/ /gi/ /gu/ /da/ /di/ /du/, mais d'autres algorithmes peuvent être utilisés (cf. [2], [3], [4]).

7.1.2 Reconnaissance dynamique.

7.1.2.1 Pour des mots isolés.

Pour un nombre d'exemples fixé (cf. chp. 4), un système d'apprentissage de faible capacité réduit l'écart entre la performance mesurée sur les exemples, et la performance réelle. Pour une même performance mesurée sur les exemples, un système de faible capacité est donc préférable.

Or, même un mot isolé présente des régularités remarquables. Deux prononciations du même mot diffèrent non seulement par la réalisation des phonèmes qui le composent, mais aussi par les instants auxquels ils apparaissent.

---

1 **J.L. Elman, D. Zipser:** *Learning the hidden structure of speech*. UCSD Institute for Cognitive Science Tech. Report 8701, San Diego, (1987).

2 **Prager R.W., Harrison T.D., Fallside F.:***Boltzmann machines for speech recognition* - Computer, Speech & Language - Vol 1, n°1, pp 3-27 (1986)

3 **Lippmann R.P, Gold B.:** *Neural Classifiers useful for Speech Recognition* - Procs of 1st International Conf. on Neural Networks, IEEE IV-417 (1987)

4 **Kohonen T.:** *The "Neural" phonetic typewriter*. IEEE Computer, March 1988, pp 11-22, (1988)

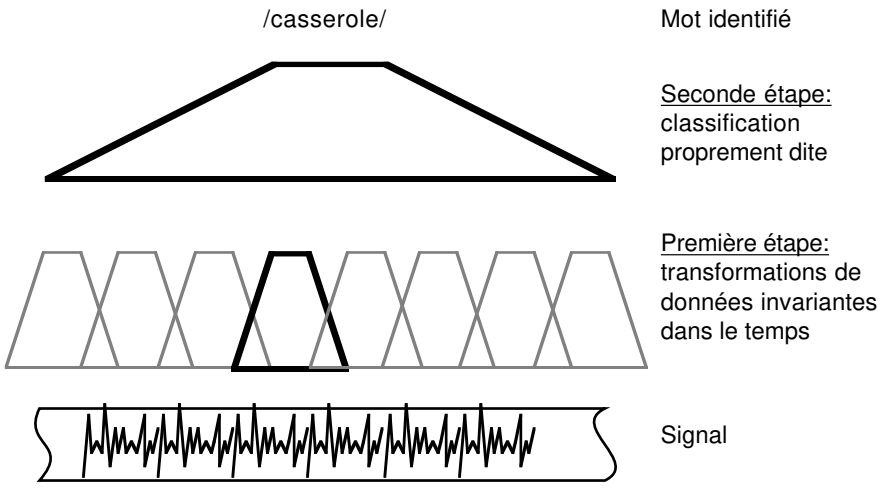


Fig 7.2 - Système invariant dans le temps pour la reconnaissance de mots isolés

On peut concevoir un système beaucoup moins sensible à cette dernière cause de variations organisé comme suit (Fig 7.2):

- La première étape consiste à transformer le signal de façon *invariante dans le temps*. Cette transformation vise à réduire l'information totale transportée par notre signal, sans toutefois perdre l'information phonétique que nous recherchons.

- Dans une seconde étape, un *système de classification* de capacité plus faible peut identifier les mots à partir de ce codage plus compact.

Un tel système, de capacité plus faible, peut offrir la même performance sur les exemples d'apprentissage, et donc une performance réelle supérieure.

Malheureusement, on ne connaît pas a priori un tel système de transformation du signal. Il devra donc être *déterminé par apprentissage*, *au travers du système de classification* dont il dépend.

7.1.2.2 Pour de la parole continue.

Certains systèmes de classification donnent des informations quantitatives sur la ressemblance entre la forme observée et les formes apprises. C'est en particulier le cas de ceux (cf. §3.4) qui fournissent des probabilités a posteriori ou des vraisemblances conditionnelles.

Dans [1], les auteurs suggèrent d'utiliser les indices statistiques fournis par de tels systèmes de classification comme *distance locale* dans un algorithme de programmation dynamique (Fig 7.3).

---

1 **Bourlard H., Wellekens C.J.:** *Speech Pattern Discrimination and Multilayer Perceptrons* - Computer, Speech and Language - vol 3, pp 1-19, (1989) & also Philips Research Lab. Brussels, Manuscript M211 (1987)

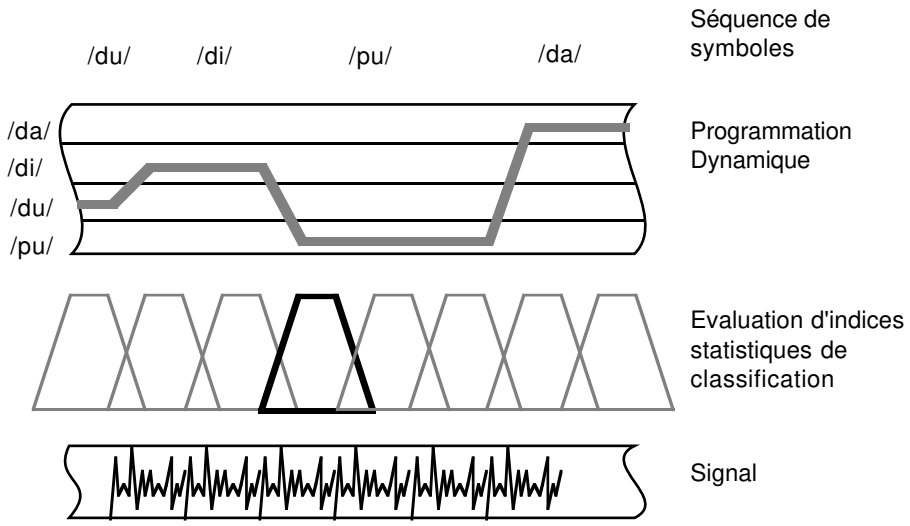


Fig 7.3 - Utilisation de système d'évaluation d'indices statistiques de classification comme générateur de distances locales pour une programmation dynamique.

L'apprentissage du premier étage, chargé d'évaluer des indices statistiques est difficile à mettre en œuvre. Les exemples d'apprentissage doivent être des unités phonétiques segmentées, le plus souvent à la main. Depuis quelques années, des efforts considérables visent à constituer de telles bases de données [1].

Il serait bon, cependant, de s'affranchir de cette contrainte. Il faudrait pour cela *entraîner ces systèmes au travers de l'étage de programmation dynamique*.

Les modèles de Markov cachés sont un représentant remarquable d'un tel système. Les vraisemblances locales $b_i(O_t)$ servent, au moyen de l'algorithme de Viterbi (6.31), à déterminer des séquences d'unités phonétiques. Or, cet algorithme n'est jamais qu'une variante de programmation dynamique.

7.1.2.3 Liens entre ces deux approches.

Il suffit de comparer les figures 7.2 et 7.3 pour se convaincre du lien étroit qui unit ces deux approches. Chacune est constituée de deux étapes de traitement.

- Une étape de *transformation* du signal, invariante dans le temps. Cette étape est explicite dans le cas de la reconnaissance de mots isolés. Dans le cas de la reconnaissance de séquences, l'extraction d'indices statistiques de classification n'est qu'un cas particulier de transformation invariante du signal.

---

1 **Lamel L.F., Kassel R.H., Seneff S.:** *Speech Database development: Design and analysis of the acoustic-phonetic corpus* - Proc. DARPA Speech Recogn. Workshop, L.S.Baumann Ed, pp 100-109 (1986)

- Dans une seconde étape, l'*extraction* soit de mots isolés, soit de séquences d'unités phonétiques est alors effectuée à partir du signal transformé.

Bien souvent, chacun de ces deux étages est un système défini par apprentissage. Il est alors important de savoir réaliser l'apprentissage de ces deux étages de façon *globalement optimale*. Une réponse générale est proposée au chapitre 8.

7.2 Réseaux à délais.

Cependant, un tel *apprentissage coopératif* peut être réalisé simplement au moyen de l'algorithme de rétro-propagation du gradient. Celui ci réalise en fait l'apprentissage simultané des couches successives d'un perceptron multi-couches. C'est l'idée fondamentale des *réseaux à délais*.

7.2.1 Structure et apprentissage des réseaux à délais.

Les réseaux à délais (Time Delay Neural Networks, TDNN) ont été largement utilisés pour la reconnaissance de phonèmes pour plusieurs langues, dont le japonais [1], l'anglais, [2], et le français [3]. Ils ont également été utilisés dans le cadre de la reconnaissance de mots isolés [4]. Un réseau à délais est en fait un cas particulier de *perceptron multi-couches à connexions contraintes*.

On essaie, dans un réseau à délais, de reconstituer l'ensemble de nos deux étages de traitement à l'intérieur d'un perceptron multi-couches.

- L'étape de transformation est alors effectuée par un ensemble de couches contraintes, c'est à dire reliées par des connexions à poids partagées, (cf. §2.2.3.*iv*).

  On regroupe les unités des couches contraintes en *extracteur de traits*. Chacun est constitué d'unités effectuant une même transformation du signal à des instants différents. Ces unités

---

1 **Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K.:** *Phoneme recognition: neural networks vs. Hidden Markov Models*. Proceedings ICASSP 88, S-Vol.1, 107-110, (1988).

2 **Lang K., Hinton G.:** *the development of the Time Delay Neural Network Architecture for Speech Recognition*, Carnegie Mellon University TR CMU-CS-88-152, (1988)

3 **Devillers L.:** *Reconnaissance monolocuteur des phonèmes français au moyen de réseaux à masques temporels* - Procs. des XVIIIèmes Journées d'Etudes sur la Parole - Montreal (1990)

4 **Bottou L.:** *Reconnaissance de la parole par réseaux multi-couches* - Rapport de stage, DEA d'Informatique du LRI, Univ. Paris XI, Orsay (1988)

partagent les même poids, mais sont reliées à des fenêtres successives de la couche précédente (Fig 7.4).

- Dans le cas de la reconnaissance de mots isolés, l'étape d'extraction est un simple classificateur linéaire, constitué d'une couche de sortie contenant une unité par classe, totalement connectée à la couche précédente. La classe reconnue correspond alors à l'unité dont la sortie est la plus forte.

  Le cas de la reconnaissance de séquences est plus difficile. Une solution simple, l'intégration temporelle, est souvent adoptée dans le cas de la reconnaissance de phonèmes. La dernière couche contrainte est alors constituée d'un extracteur de trait par phonème. Le phonème reconnu est alors identifié par l'extracteur de trait dont la somme des sorties est la plus grande.

L'apprentissage d'un tel réseau est effectué à l'aide de la procédure de rétro-propagation du gradient, modifiée pour tenir compte de la présence de poids partagés (cf. §2.3.4.2.*ii*).

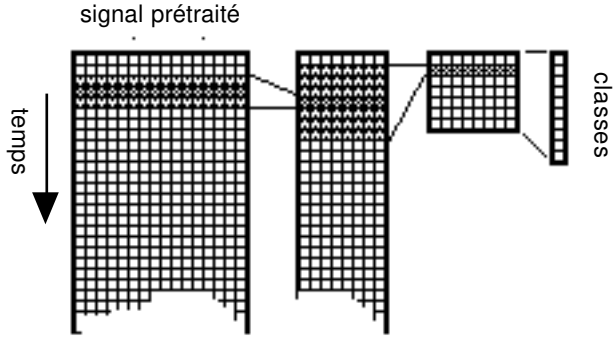


Fig 7.4 - Un réseau à délais. Chaque colonne de cellules dans une couche cachée constitue un extracteur de traits. Chaque couche cachée effectue une transformation du signal invariante dans le temps. Ici, l'extraction est assurée par une couche totalement connectée.

7.2.2 Réseaux récurrents et réseaux à délais.

Une autre classe de perceptrons multi-couches, les *réseaux récurrents*, permet également de réaliser des systèmes dynamiques pour la reconnaissance de la parole [1], [2], [3].

---

1 **Watrous R.L.:** *Learned Phonetic Discrimination Using Connectionists Networks*, in European Conference on Speech Technology, pp 377-380, Edinburgh (sept 1987)

2 **Kuhn G., Watrous R.L., Ladendorf B.:** *Connected recognition with a recurrent network*- Procs of NeuroSpeech 89, Edinburgh, Scotland (1989)

3 **Gori M., Bengio Y., De Mori R.:** *BPS, A learning algorithm for capturing the dynamic nature of speech*, Proc of Intl. Joint Conf. on Neural Networks, Washington DC, Vol II, pp 417-423, (1989)

Dans un réseau récurrent, les connexions sont autorisées à former des boucles. On adopte souvent un séquencement synchrone: Toutes les cellules sont mises à jour simultanément, leurs états à l'instant t sont utilisés pour calculer leurs états à l'instant $t+1$.

Un tel réseau est évidement invariant dans le temps, car le même réseau est utilisé à chaque instant. Les boucles lui permettent seulement d'utiliser des informations provenant du passé.

L'apprentissage *per se* d'un tel réseau requiert encore des aménagements de la règle de rétro-propagation. Pineda [1] a étudié un algorithme qui permet de stocker des états stables, c'est à dire des associations entrées-sorties laissées invariantes par le réseau.

Dans le cas de la parole, les entrées, c'est à dire le signal, ne sont jamais invariantes. Les sorties, c'est à dire les séquences d'unités phonétiques, évoluent également dans le temps, selon la nature du signal. L'algorithme d'apprentissage doit permettre au réseau d'apprendre non des états stables, mais une trajectoire contrôlée par les entrées. Cette formulation conduit à l'algorithme présenté dans [2].

Ce dernier algorithme n'est en fait qu'une adaptation de la rétro-propagation avec poids partagés. En effet, un réseau récurrent peut toujours être *déplié* (Fig 7.5).

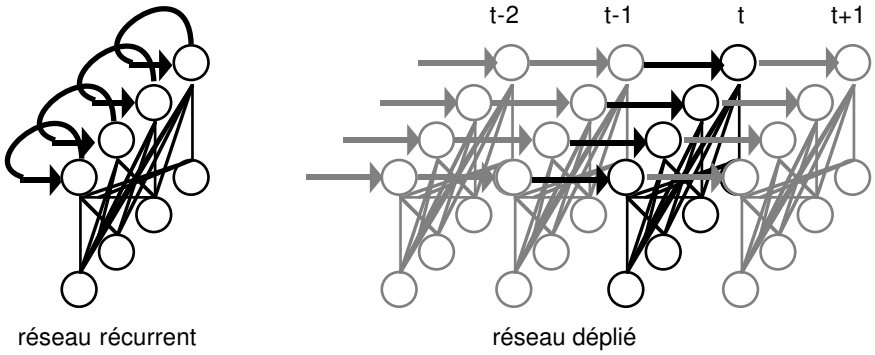


Fig 7.5 - Le dépliage temporel d'un réseau récurrent.

Il suffit de juxtaposer plusieurs copies de notre réseau, à des instants successifs. Aux boucles, correspondent alors des connexions liant les unités de deux réseaux adjacents. On obtient alors un gros réseau, dont toutes les connexions sont contraintes à rester invariantes dans le temps. Il suffit alors, comme pour un réseau à délais, d'appliquer l'algorithme de rétro-propagation avec poids partagés.

Un point délicat consiste à déterminer la longueur du dépliage temporel. En effet, les premières unités de notre gros réseau ne peuvent recevoir d'information des instants précédents. En pratique, les

---

1 **Pineda F.J.:** *Generalization of Back-Propagation to Recurrent Neural Networks,* Physical Review Letters, Vol 59,n°19, pp 2229-2232, (1987)

2 **Pearlmutter B.:** *Learning State Space trajectories in Recurrent Neural Networks* - CMU Tech. Report CMU-CS-88-191, (1988)

problèmes abordés recèlent toujours un horizon: En parole, par exemple, il est impossible de prononcer une phrase infinie sans y insérer de silences. On se limite alors à des portions de phrases comprises entre deux silences.

7.2.3 Exemples de réseaux à délais.

Ces exemples, issus de [1] et [2], sont destinés à montrer concrètement comment on peut mettre en œuvre un réseau à délais pour une tâche de reconnaissance de la parole et quels sont les problèmes rencontrés.

7.2.3.1 Données et prétraitements.

Deux ensembles de données ont été utilisés.

- L'un d'eux (B1) a été élaboré au LIMSI. La partie que nous avons utilisée est constituée des 10 chiffres (zéro à neuf) français, prononcés par 26 locuteurs, hommes et femmes.

  Chaque chiffre a été prononcé une fois par chaque locuteur, dans un environnement silencieux. Ces données ont été segmentées manuellement, avec précision.

- Une base de données en plusieurs langues (anglais, allemand, français, italien et espagnol) a été constituée dans le cadre du projet Esprit "Pygmalion". Seule la partie française (B2) était alors disponible. Elle est composée de trente mots, dont les dix chiffres et vingt mots de commande (ouvre, efface, arrête…), prononcés 10 fois par 10 locuteurs dont cinq sont des femmes.

  Le niveau de bruit est assez élevé, comparable à celui qui règne dans un bureau. De plus, les mots ont été segmentés automatiquement, à l'aide d'un programme de mesure du niveau d'énergie. Le résultat est très pauvre; des erreurs de 100 ms ne sont pas rares, alors que certains mots ne durent pas plus de 300 ms!

Dans les deux cas, le signal a été prétraité de façon très classique [3]. Après avoir franchi un filtre anti-repliement à 5 kHz le signal est échantillonné à 10 KHz. Après avoir amplifié numériquement les

---

1 **Bottou L., Fogelman Soulié F., Blanchet P., Liénard J.S.:** *Speaker independent isolated digit recognition: Multilayer perceptron vs Dynamic Time Warping*, Neural Networks, vol3, pp 453-465, (1990)

2 **Driancourt X.D., Bottou L.:** *TDNN-Extracted Features*, Procs. of Neuro-Nîmes 90, EC2, (1990)

3 **Gauvain J.L.:** *A syllable based isolated word recognition experiment* - Procs. IEEE Conf. on ASSP 1986 (1986)

hautes fréquences, on simule un banc de seize filtres répartis selon l'échelle de Bark, de la façon suivante.

Une transformée de Fourier rapide est appliquée sur une fenêtre de 25.6 ms du signal, décalée de 12.8 ms chaque fois (10 ms dans le cas de la seconde base de données). On obtient alors un spectre de 128 énergies, dans la bande 0-5 kHz. Ces énergies sont sommées sur 16 fenêtres fréquentielles triangulaires représentant chaque élément de notre banc de filtres. Les résultats sont alors codés logarithmiquement sur 8 bits.

On obtient alors des séquences d'un vecteur à 16 composantes toutes les 12.8 ms (ou 10 ms dans le cas de la seconde base de données).

7.2.3.2 Reconnaissance de chiffres indépendamment du locuteur.

i Apprentissage avec 16 locuteurs.

Pour une première expérience, les 16 derniers locuteurs de B1, dans l'ordre alphabétique, ont été utilisés pour entraîner un réseau à délais (Fig 7.6). Les 10 locuteurs restants constituent un ensemble de test pour mesurer grossièrement la performance de notre système.

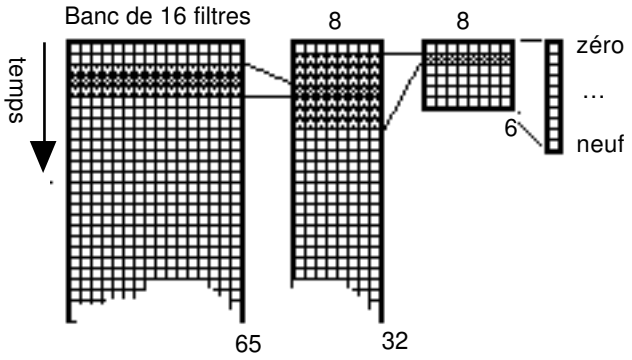


Fig 7.6: Un réseau pour la reconnaissance de chiffres. Les unités de la première couche cachée sont reliées à des fenêtre de trois trames sur la couche d'entrée se recouvrant d'une trame. Les unités de la seconde couche cachée sont reliées à des fenêtres de 7 unités de la couche précédente, se recouvrant de deux unités. La couche de sortie est totalement connectée.

L'entrée de ce réseau contient 65 tranches de 12.8 ms, c'est à dire assez pour contenir tous les chiffres dont nous disposons. Dans un premier temps, ceux ci sont cadrés à gauche, et complétés par des zéros, correspondant au silence, afin de remplir l'entrée du réseau. Après apprentissage, aucune erreur n'est relevée sur l'ensemble d'apprentissage, et le taux d'erreur relevé sur les mots prononcés par les 10 autres locuteurs est de 5%.

Malheureusement, le taux d'erreur sur cet ensemble de test grimpe à 11% lorsque l'on décale nos mots aléatoirement de 0 à 4 trames, et à 30% lorsque on décale nos mots de 0 à 8 trames. Les premières

couches de notre réseau sont en effet invariantes par translation, mais l'étage de classification, simple séparateur linéaire, ne l'est pas.

On décide alors d'entraîner et de tester le réseau avec les même échantillons d'apprentissage et de test, aléatoirement décalés de 0 à 8 trames dans l'entrée. Cela a pour effet d'augmenter artificiellement l'ensemble d'apprentissage: On obtient alors de meilleurs résultats sur l'ensemble de test.

On mesure alors 0.8% d'erreur sur l'ensemble d'apprentissage, et 1% sur l'ensemble de test (aléatoirement décalé lui aussi), lors du meilleur apprentissage. Les mêmes données ont été utilisées avec un programme d'alignement temporel du LIMSI. Le taux d'erreur était également de 1%.

Ce résultat ne *permet absolument pas de comparer* les deux méthodes: D'après (4.31) la demi largeur d'un *intervalle de confiance* à 90% est de 12% pour nos 100 mots de test !

On constate en revanche que la technique d'augmentation artificielle de l'ensemble d'apprentissage semble conduire à de meilleurs résultats (avec les réserves qu'impose notre ensemble de test trop faible), et à une intéressante résistance à une mauvaise segmentation.

ii Apprentissage avec quatre ensembles de 10 locuteurs.

Dans le but d'améliorer les intervalles de confiance, quatre ensembles d'apprentissages de 10 locuteurs ont été définis. Les quatres réseaux ainsi obtenus ont été testés sur les 16 locuteurs restants. La demi largeur d'un intervalle de confiance à 90% autour de la moyenne de ces quatres résultats est alors de 5% environ.

Le taux d'erreur pour l'alignement temporel est alors de 1,9%. Le taux d'erreur pour le TDNN est de 4,8% sur les données de test non décalées, et de 5,5% sur les données de test aléatoirement décalées de 0 à 8 trames.

Sous réserve, encore, de la validité statistique de ces résultats, la réduction de l'ensemble d'apprentissage a peu affecté le programme d'alignement temporel, et a beaucoup plus affecté le TDNN. En revanche, ses propriétés de résistance à une mauvaise segmentation restent intéressantes.

7.2.3.3 Reconnaissance multi-locuteurs de mots isolés.

Les données B2, plus nombreuses, autorisent de meilleures mesures de la performance de nos systèmes. Elle ne possède malheureusement que 10 locuteurs, ce qui restreint trop l'intérêt d'un test indépendant du locuteur.

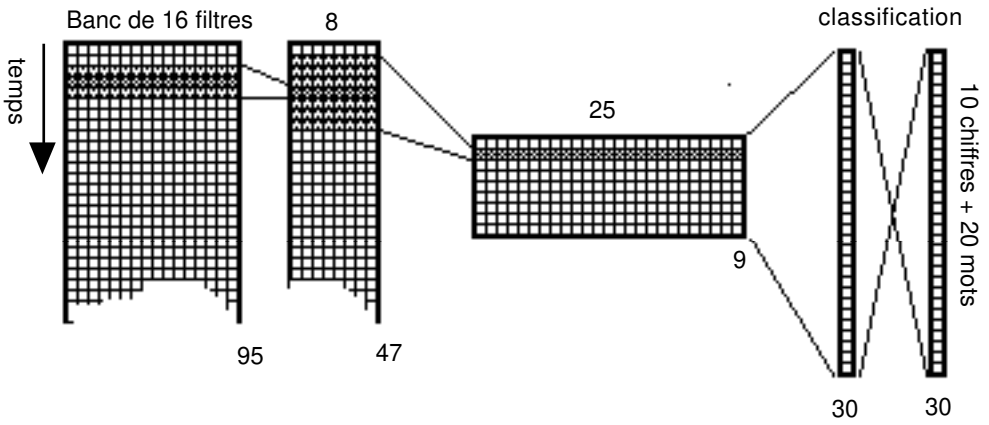


Fig 7.7 Un réseau pour la reconnaissance de 30 mots isolés. Ici, l'étape de classification est assurée par un perceptron multi-couches, avec 30 unités cachées, totalement connectées.

Nous avons donc décidé, pour chaque locuteur et chaque mot, d'utiliser cinq répétitions pour notre ensemble d'apprentissage, et cinq pour un ensemble de test. On dispose donc d'un échantillon de 1500 formes pour mesurer le taux d'erreur; la demi-largeur d'un intervalle de confiance à 90% est alors de 3% environ.

Il semblait difficile d'utiliser exactement l'architecture de réseau précédente. Plusieurs réseaux ont donc été essayés. Le meilleur est représenté dans la figure ci-après (Fig 7.7). Ce réseau possède 2181 unités, 39298 connections, seulement 10095 paramètres. Une couche cachée supplémentaire a été insérée entre la dernière couche contrainte et la sortie du réseau. En effet, un système linéaire de classification s'est avéré moins performant pour déterminer les frontières de nos 30 classes.

Le taux d'erreur mesuré sur les exemples d'apprentissage est de 0.5%. Le risque moyen, estimé sur les exemples de test est de 3,5%±3%. Le risque moyen associé à un réseau légèrement différent, sans couche cachée supplémentaire dans l'étage de classification, est de 5,5%±3%.

7.2.3.4 Utilisation des codages exhibés par un réseau.

On l'a vu, le système final de classification possède un impact non négligeable sur la qualité du système obtenu. On souhaite donc, comme le suggère la section 7.1, utiliser un classifieur bien adapté aux données de parole, comme l'alignement temporel.

Cela présente de multiples avantages:

- Comparé à un réseau à délai seul, ce système hybride permet d'aborder des problèmes de parole continue.

- Comparé à un alignement temporel seul, la compression des données effectuée par le réseau se traduit par une réduction quadratique du temps de calcul pendant la reconnaissance, et donc d'opérer en temps réel.

Le problème réside alors dans l'apprentissage coopératif des deux modules composant ce système. Il est cependant possible de contourner ce problème, en entraînant séparément nos deux modules. On *perd alors toute garantie d'optimalité*, mais on espère atteindre un *niveau de performances supérieur*.

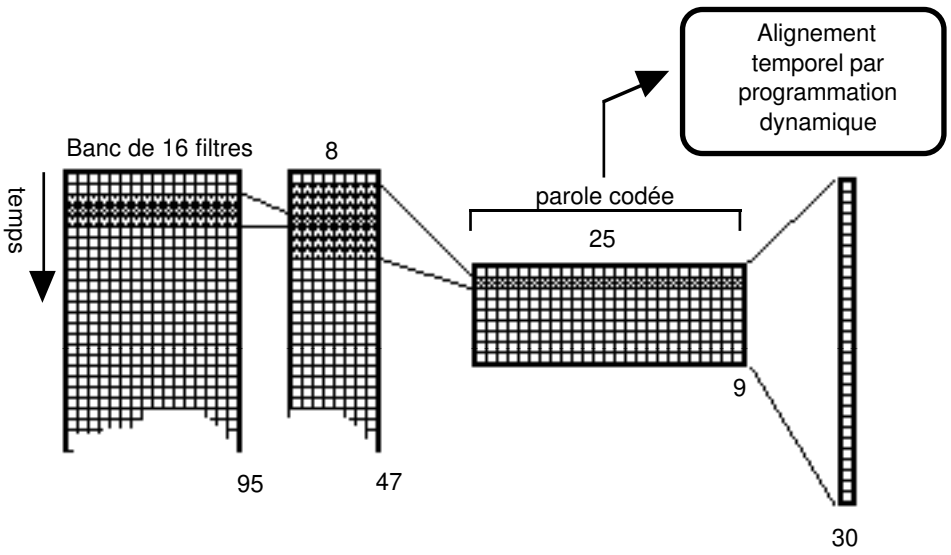


Fig 7.8 - Un système hybride: Le TDNN est entraîné au travers d'une couche de classification linéaire. On utilise, en phase de reconnaissance, un programme d'alignement temporel opérant sur les codages intermédiaires générés par les couches internes du réseau.

On procède alors de la façon suivante (fig 7.8):

- On entraîne un réseau à délais, au travers d'un classificateur linéaire, c'est à dire une couche de sortie totalement connectée.

  Le risque moyen, évalué sur l'ensemble de test, est, on l'a vu, de 5,5%±3%. Pour atteindre ce résultats, l'étage de classification se fonde uniquement sur les états de la dernière couche contrainte. Ces états constituent donc un *codage* du signal, sous la forme d'une séquence de 9 vecteurs à 25 composantes, assez significatif pour autoriser ce niveau de performances.

- Pendant la phase de reconnaissance, on utilise un programme d'alignement temporel pour comparer ces codages à ceux correspondant aux exemples d'apprentissage, préalablement stockées.

Nous disposions d'un programme d'alignement dynamique sans raffinement, qui commet plus de 10% d'erreurs sur l'ensemble de test lorsque l'on travaille sur les données spectrales. En utilisant ce programme comme étage de classification dans notre réseau à délai, le risque moyen, mesuré sur l'ensemble de test, chute à 1,5%±3%.

Sur cet exemple au moins, la combinaison d'un réseau à délais et d'un système d'alignement temporel améliore les résultats par rapport à chaque système. Il semble raisonnable d'espérer des résultats meilleurs encore, en procédant à un vrai apprentissage coopératif de nos deux modules…

7.3 Conclusion.

Le problème de reconnaissance de la parole illustre bien les trois points capitaux qui doivent être considérés, avant d'envisager tout système d'apprentissage:

- Atteindre un niveau raisonnable de validité statistique requiert de *collecter un échantillon important de données*, non seulement *pour l'apprentissage*, mais aussi *pour évaluer et comparer* la qualité des systèmes obtenus.

  Or la collecte d'un tel ensemble de données est une opération aussi coûteuse qu'ennuyeuse. Souvent, on utilise des données déjà collectées, qui réduisent cruellement, à la fois la variété des apprentissages réalisables et la signification des résultats.

- La connaissance a priori du problème suggère d'*utiliser des systèmes contraints*. Un tel système, c'est le cas des réseaux à délais, permet d'atteindre un même coût empirique avec une capacité moindre, et souvent, pour un même échantillon d'apprentissage, un risque moyen inférieur.

  Mais *tisser ces contraintes à l'intérieur d'un perceptron multi-couches n'est pas la panacée*: L'adoption, comme étage de classification, d'un programme assez grossier d'alignement temporel permet à la fois de réduire significativement le risque moyen, et d'aborder des problèmes de reconnaissance de parole continue.

- Il faut donc concevoir des *systèmes hybrides*, combinant à la fois des systèmes connexionnistes et des systèmes classiques éprouvés (comme l'alignement temporel). Il faut alors entraîner coopérativement les divers composants du système complet.

En fait, cette notion de coopération de systèmes d'apprentissage fournit un langage pratique pour décrire et concevoir un algorithme complexe. Le chapitre suivant est consacré à son étude.

8 Algorithmes modulaires d'apprentissage.

8.1 Systèmes modulaires.

De nombreux systèmes complexes d'apprentissage s'expriment aisément sous la forme de la coopération de modules plus simples. Cette expression porte l'accent sur les aspects structurels des systèmes d'apprentissage. Sa formalisation consiste simplement à écrire la fonction de coût local $J(x,w)$ *comme une composée de fonctions.*

8.1.1 Motivation.

Le chapitre précédent (cf §7.1) présente plusieurs systèmes adaptatifs pour la reconnaissance de la parole, qui peuvent être interprétés comme la coopération de deux étages, possédant chacun leur propre algorithme d'apprentissage.

Plus généralement, la pratique suggère d'introduire une *structure* appropriée dans les systèmes d'apprentissage. En effet, il est bien plus commode de concevoir un système complexe en assemblant des systèmes plus simples, que d'imaginer son fonctionnement global.

Une méthode unifiée pour la coopération de plusieurs algorithmes d'apprentissage présente donc de multiples intérêts.

- Elle permet d'aborder l'aspect structurel des systèmes d'apprentissage, totalement ignoré dans les chapitre 3 et 4.

- Elle permet, en pratique, de concevoir des systèmes *modulaires*, ce qui peut avoir un impact considérable sur le développement et l'implémentation de systèmes d'apprentissage complexes. Cet impact peut être comparé à celui des notions de modularité et d'encapsulation sur le développement de logiciels complexes.

Cette modularité permet alors de concevoir un algorithme adapté à chaque problème.

Dans le chapitre 7, on a trouvé préférable d'utiliser une programmation dynamique plutôt qu'un classifieur linéaire comme dernier étage d'un système de reconnaissance de la parole.

Une autre illustration est présentée dans [1]. Les algorithmes LVQ (cf §2.3.5) sont réputés efficaces pour des tâches de classification, mais sont peu adaptés aux données bruitées. Un système composé d'un perceptron multi-couches et d'un LVQ permet d'obtenir de meilleurs résultats que chacun de ces systèmes utilisés isolément. Ce résultat, observé sur des données synthétiques, est confirmé par [2] dans le cas de problèmes de reconnaissance de mots et de phonèmes isolés.

Dans ces exemples, chaque système était entraîné *séquentiellement*: Dans un premier temps, on entraîne un perceptron multi-couches pour notre tâche de classification. Ensuite, on utilise les premiers étages du perceptron, et on classe les sorties obtenues au moyen de l'algorithme LVQ (Fig. 8.1).

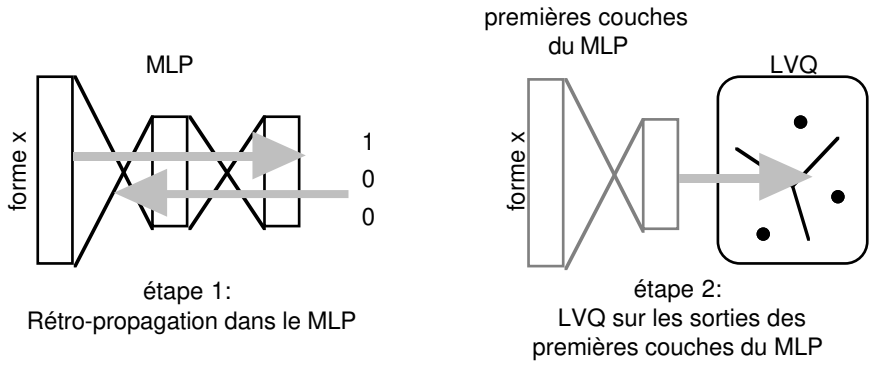


Fig 8.1 - Apprentissage séquentiel MLP+LVQ.
(1): On entraîne le MLP. (2): Le MLP est figé, on entraîne LVQ.

---

1 **Bollivier M. de, Gallinari P., Thiria S.:** *Cooperation of neural nets for robust classification*, IJCNN 90, San Diego, vol1, pp 113-120, (1990)

2 **Bennani Y., Charouar N., Gallinari P., Mellouk A.:** *Comparing Neural net models on speech recognition tasks*, Procs of Neuro-Nîmes 90, to appear (1990)

Dans la première étape, les premières couches du perceptron ont été entraînées à produire des représentations qu'un séparateur linéaire pouvait décoder. Ce séparateur linéaire n'est pas optimal: Utiliser sur ces données une méthode de plus proches voisins, comme LVQ, permet d'améliorer les performances. Or, ces données n'ont pas été optimisées pour être décodées à l'aide d'un tel algorithme.

On peut alors espérer de meilleurs résultats en entraînant *coopérativement* nos systèmes, et non séquentiellement. Dans ce cas, il est possible de mélanger ces deux méthodes [1], et d'obtenir ainsi un algorithme hybride.

8.1.2 Ecriture modulaire du coût local $J(\mathbf{X},\mathbf{W})$.

8.1.2.1 Une approche de la modularité.

Nous avons vu qu'une grande variété d'algorithmes d'apprentissage s'exprime comme la minimisation par descente stochastique de gradient (3.6) d'une fonction de coût de type (3.2):

$$C = \int J(\mathbf{x},\mathbf{w})\ p(\mathbf{x})\ d\mathbf{x} \qquad \textit{et} \qquad \mathbf{w}^{t+1} = \mathbf{w}^t - \varepsilon_t\ \nabla J(\mathbf{x}^t,\mathbf{w}^t)$$

Ces algorithmes ne diffèrent que par la forme de la fonction de coût local $J(\mathbf{X},\mathbf{W})$, et pourtant possèdent des propriétés bien distinctes.

Nos systèmes d'apprentissage sont avant tout des systèmes de traitement de l'information. Des informations fournies en entrée sont transformées, et restituées en sortie. Entrées et sorties représentent l'interaction du système avec l'extérieur, et sont représentées toutes deux par la variable $\mathbf{X}$ du coût local.

Ce coût local $J(\mathbf{X},\mathbf{W})$ exprime donc deux propriétés distinctes:

- D'une part, il décrit une dépendance paramétrique entre les entrées et les sorties,
- D'autre part, il mesure l'adéquation de cette dépendance aux spécifications de notre système, c'est à dire à ce que nous voulons apprendre.

Une formulation modulaire d'un dépendance paramétrique entre entrées et sorties est aisément concevable: Notre système peut être décrit comme une boite noire, elle même composée d'un enchaînement de boites noires. Ces boites noires, à leur tour, sont sujettes à une description modulaire.

---

1 **Bollivier M. de, Gallinari P., Thiria S.:** *Multi-Module Neural Networks for Classification* - Procs of INNC'90, vol 2, pp 777-780, (1990)

En revanche, notre système global est conçu dans un but déterminé. Une mesure de l'adéquation de notre système à ses spécifications est donc nécessairement globale: elle ne découle pas de mesures partielles associées aux boites noires qui composent un système modulaire. L'apprentissage d'un tel système consiste donc à *minimiser un coût unique*.

L'expression locale $J(\mathbf{x},\mathbf{w})$ de ce coût unique doit alors être elle-même considérée comme une transformation paramétrique des entrées, et non comme la résultante de coûts locaux associés à chaque boite noire.

8.1.2.2 Modélisation.

Nous décrirons un système modulaire comme l'association de modules élémentaires F_n dont les entrées seront notées x_k, les sorties y_j et les paramètres w_i. Les entrées de chaque module sont reliées à des sorties appartenant aux modules précédents (Fig 8.2).

Chaque module est défini par une relation dérivable, entre entrées, sorties et paramètres. Cette formulation est en effet assez générale pour représenter tous les algorithmes présentés ici.

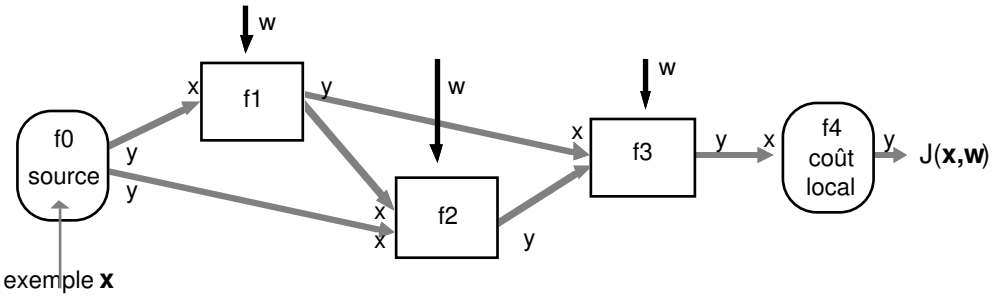


Fig 8.2 - Un système modulaire.

Pour simplifier les calculs, nous considérerons que le premier module f_0 ne possède que des sorties: Sa fonction consiste à alimenter le système global avec de nouvelles données. De plus, comme annoncé ci-avant, le dernier module calcule simplement le coût local $J(\mathbf{x},\mathbf{w})$.

La description mathématique des connexions entre modules requiert une certaine rigueur dans la façon d'indicer les différentes grandeurs considérées. Chaque module est défini par

$$\forall\, j \in Y^{-1}(n),\ \ y_j = f_j\left((x_k)_{k \in X^{-1}(n)} , (w_i)_{i \in W^{-1}(n)} \right) \qquad (8.1)$$

Dans cette équation, $Y^{-1}(n)$, (resp. $X^{-1}(n)$, et $W^{-1}(n)$) représente l'ensemble des indices associés aux sorties (resp entrées et paramètres) du module F_n. Inversement, nous écrirons $Y(j)$ (resp. $X(k)$ et $W(i)$) le numéro du module associé aux sorties y_j (resp. entrées x_k et paramètres w_i).

La topologie de ces connexions entre les modules F_n sera exprimée par une fonction ϕ. Appliquer ϕ à l'indice d'une entrée donne l'indice de la sortie reliée à cette entrée.

$$\forall k,\ x_k = y_{\phi(k)} \tag{8.2}$$

Le fait que les modules soient connectés sans boucles s'exprime alors:

$$k \in X^{-1}(n) \ \Rightarrow\ \phi(k) \in \bigcup_{p=0}^{n-1} Y^{-1}(p) \tag{8.3}$$

Dans la suite, notre objectif consistera à calculer les dérivées du coût local $J(\mathbf{x},\mathbf{w})$, afin d'appliquer la forme générale d'algorithme d'optimisation stochastique (3.6). Or, il est possible de calculer simplement ces dérivées, à l'aide seulement des dérivées des fonctions f_l.

8.2 Apprentissage de systèmes modulaires.

Un formalisme Lagrangien permet alors de calculer aisément le gradient coût local $J(\mathbf{x},\mathbf{w})$ de façon peu dépendante de la forme des modules. Ce gradient peut être calculé efficacement au cours d'une simple passe arrière.

Les liens entre ce calcul du gradient, le "credit assignment problem", et l'algorithme de rétro-propagation du gradient, sont également soulignés.

8.2.2 Calcul des dérivées du coût local.

Afin de calculer facilement les dérivés de J par rapport à tous les paramètres d'un système modulaire, il est utile d'introduire un formalisme Lagrangien.

8.2.2.1 Un formalisme Lagrangien.

Considérons deux fonctions, $f(\mathbf{x})$ de $\Re^I$ vers $\Re^M$ et $g(\mathbf{y})$ de $\Re^M$ vers $\Re$. On calcule d'habitude les dérivées de la composée $g\circ f$ à l'aide de la règle de la dérivation en chaîne. Une autre façon consiste à écrire le Lagrangien de $g\circ f$, avec la contrainte. $\mathbf{y}=f(\mathbf{x})$:

$$L(\mathbf{x},\mathbf{y},\beta) = g(\mathbf{y}) - \sum_m \beta_m \left(y_m - f_m(\mathbf{x}) \right) \tag{8.4}$$

Un calcul simple montre que choisir β en résolvant les équations

$$\forall\ m,\ \left.\frac{\partial L}{\partial y_m}\right|_{\mathbf{x},f(\mathbf{x}),\beta(\mathbf{x})} = 0$$

implique

$$\left.\frac{\partial L}{\partial x_i}\right|_{\mathbf{x},f(\mathbf{x}),\beta(\mathbf{x})} = \left.\frac{\partial g\circ f}{\partial x_i}\right|_{\mathbf{x}} . \qquad (8.5)$$

Cette propriété simplifie considérablement le calcul des dérivées de notre coût local J. Cela introduit automatiquement les multiplicateurs de Lagrange β, qui sont les quantités conjuguées des variables internes $\mathbf{y}$.

8.2.2.2 Formules pour le gradient.

Calculer les dérivées est alors très simple. Les équations (8.1) et (8.2) sont les contraintes de notre système. On peut alors écrire le Lagrangien, en prenant garde aux conventions d'indiçage:

$$L = J - \sum_k \beta_k \, (x_k - y_{\phi(k)}) - \sum_j \alpha_j \left(y_j - f_j((x_k)_{k\in X^{-1}Y(j)}, (w_i)_{i\in W^{-1}Y(j)}) \right) \qquad (8.6)$$

Les quantités conjuguées β et α sont déterminées par les équations:

$$\frac{\partial L}{\partial y_j} = 0 = \quad - \alpha_l + \sum_{k\in\phi^{-1}(j)} \beta_k \qquad (8.7)$$

$$\frac{\partial L}{\partial x_k} = 0 = \quad - \beta_k + \frac{\partial J}{\partial x_k} \qquad \textit{si l'entrée } k \textit{ appartient au dernier module.}$$

$$- \beta_k + \sum_{j\in Y^{-1}X(k)} \alpha_j \frac{\partial f_j}{\partial x_k} \quad \textit{sinon.} \qquad (8.8)$$

Dans l'équation (8.8), on distingue le cas particulier où k est l'indice d'une entrée du dernier module, dont le rôle est de calculer le coût local J.

Les multiplicateurs de Lagrange α_j et β_k sont les quantités conjuguées des sorties et entrées de nos modules, i.e. $\alpha_j = \partial J/\partial y_j$, and $\beta_k = \partial J/\partial x_k$.

Dans chaque module, les conjuguées des entrées β_k sont calculées à partir des conjuguées des sorties α_j grâce à l'équation (8.8). L'équation (8.7) combine à son tour ces conjuguées β_k, pour calculer les conjuguées α_j des sorties des modules précédents, le long des connexions.

Ainsi, tous les multiplicateurs de Lagrange peuvent être calculés alternativement au cours d'une unique récurrence arrière. Il est alors facile d'obtenir la dérivées Δ_i de J par rapport au paramètre w_i:

$$\Delta_i = \frac{\partial J}{\partial w_i} = \left.\frac{\partial L}{\partial w_i}\right|_{\mathbf{x},\mathbf{y},\beta,\alpha} = \sum_{j\in Y^{-1}W(i)} \alpha_j \frac{\partial f_j}{\partial w_i} \qquad (8.9)$$

En résumé,

$$\beta_k = \frac{\partial J}{\partial x_k} \qquad \textit{si l'entrée } k \textit{ appartient au dernier système,}$$

$$\beta_k = \sum_{j \in Y^{-1}X(k)} \alpha_j \frac{\partial f_j}{\partial x_k} \qquad \textit{dans les autres cas.}$$

$$\alpha_j = \sum_{k \in \phi^{-1}(j)} \beta_k$$

$$\Delta_i = \frac{\partial J}{\partial w_i} = \sum_{j \in Y^{-1}W(i)} \alpha_j \frac{\partial f_j}{\partial w_i} \qquad (8.10)$$

Ces formules décrivent de façon générale les échanges d'information entre modules nécessaires pour effectuer l'apprentissage global de notre système. Il suffit donc de savoir effectuer trois opérations sur chaque module:

- Calculer ses sorties y_j, connaissant ses entrées x_k et ses paramètres w_i.
- Calculer les conjugués β_k de ses entrées, connaissant les conjugués α_j de ses sorties, ses entrées x_k et ses paramètres w_i.
- Calculer les dérivées de ses paramètres Δ_i, connaissant les conjuguées α_j de ses sorties, ses entrées x_k et ses paramètres w_i.

Le reste de l'algorithme ne dépend pas de la nature des modules.

8.2.2.3 Le calcul du hessien est il possible?.

Dans le chapitre 5, on a expliqué comment les dérivées secondes du coût local J donnaient divers indices pour améliorer la vitesse de convergence des algorithmes de descente de gradient, et améliorer leurs performances.

Une méthode pratique de calcul des dérivées secondes dans un système modulaire donnerait des indices comparables.

Le formalisme Lagrangien de dérivation de fonctions composées ne s'étend malheureusement pas aisément aux dérivées secondes. Il est cependant possible de calculer ces dérivées à la main.

Si, pour chaque module F_n, nous connaissons les dérivées croisées de J par rapport aux sorties de ce module, les dérivées croisées par rapport aux entrées et aux paramètres de ce même module sont données par:

$$\frac{\partial^2 J}{\partial x_{k1}\,\partial x_{k2}} = \sum_{j1,j2} \frac{\partial^2 J}{\partial y_{j1}\,\partial y_{j2}}\;\frac{\partial f_{j1}}{\partial x_{k1}}\;\frac{\partial f_{j2}}{\partial x_{k2}} + \sum_{j} \frac{\partial E}{\partial y_j}\;\frac{\partial^2 f_j}{\partial x_{k1}\,\partial x_{k2}}$$

$$\frac{\partial^2 J}{\partial w_{i1}\,\partial w_{i2}} = \sum_{j1,j2} \frac{\partial^2 J}{\partial y_{j1}\,\partial y_{j2}}\;\frac{\partial f_{j1}}{\partial w_{i1}}\;\frac{\partial f_{j2}}{\partial w_{i2}} + \sum_{j} \frac{\partial E}{\partial y_j}\;\frac{\partial^2 f_j}{\partial w_{i1}\,\partial w_{i2}} \qquad (8.11)$$

Il n'est pas facile, en revanche, de propager de façon générale ces blocs de dérivées secondes entre modules. En effet:

- Lorsque toutes les sorties d'un système sont connectées aux entrées du suivant, les dérivées croisées par rapport aux sorties du module amont sont égales aux dérivées croisées par rapport aux entrées du système aval.

- Lorsque les sorties d'un module sont partagées en plusieurs groupes, connectés en aval à autant de modules différents, on ne peut plus calculer les dérivées croisées entre deux sorties appartenant à deux groupes différents. Le hessien du coût local par rapport aux sorties de ce module ne peut plus être déterminé exactement, bien qu'une approximation en blocs diagonaux soit possible.

- Si une sortie d'un module est utilisée plusieurs fois comme entrée par les modules suivants, nous devons ajouter les dérivées croisées correspondantes. Or, quelques termes de ces sommes manqueront.

Or, dans le cas le plus général, tous les termes du hessien du coût local par rapport aux sorties sont nécessaire pour appliquer les formules (8.11). Il est parfois possible, comme dans le cas de la rétro-propagation du gradient, d'évaluer exactement la diagonale de ce hessien.

Bien que peu satisfaisante, la situation n'est pas désespérée: Une connaissance approximative du hessien peut donner malgré tout assez d'informations pour être utile en pratique.

8.2.3 Liens avec les approches usuelles.

Il est important de souligner que ce formalisme modulaire ne constitue pas un nouvel algorithme, mais une expression modulaire des algorithmes d'optimisation. Cette expression est en particulier reliée à quelques algorithmes existants, dont bien sûr la rétro-propagation du gradient.

8.2.3.1 Le "credit assignment problem".

La problématique du *credit assignment problem* a constitué le fondement de plusieurs algorithmes classiques destinés à des systèmes en étages, comme les perceptrons multi-couches.

i Modélisation.

Considérons un système en étages, constitué d'une chaîne de modules, dont les sorties servent d'entrées aux suivants. Chaque élément de notre chaîne est décrit par la dépendance paramétrique entre entrées X et sorties Y,

$$Y = f(X, \mathbf{w}) \tag{8.12}$$

et par son algorithme d'apprentissage,

$$\mathbf{w'} = \Psi^t_w(X, Y, \mathbf{w}) \tag{8.13}$$

qui calcule un nouveau jeu de paramètres $\mathbf{w'}$ à l'aide d'une fonction Ψ^t_w de l'exemple présenté (X, Y) et des paramètres courants $\mathbf{w}$. Supposons maintenant que les sorties Y de ce module servent d'entrées à un autre module,

$$Z = g(Y, \mathbf{v}) \tag{8.14}$$

$$\mathbf{v'} = \Psi^t_v(Y, Z, \mathbf{v}) \tag{8.15}$$

Notre but est d'entraîner le système complet, décrit par les équations (8.12) et (8.14), en ne disposant que d'exemples de la forme (X,Z). Malheureusement, les algorithmes (8.13) and (8.15) requièrent la connaissance des états internes Y pour chaque exemple.

Dans l'équation (8.14), comme dans l'équation (8.12), les arguments de la fonction g ont été arbitrairement partitionnés en *entrées* X et *paramètres* $\mathbf{w}$. La formule de réestimation (8.15) s'applique seulement aux paramètres. Supposons alors que l'on puisse obtenir une formule symétrique de réestimation des entrées, de la forme

$$Y' = \Psi^t_Y(Y , Z, \mathbf{v}) \tag{8.16}$$

Ayant un exemple (X, Z), on peut alors opérer en trois étapes:

- Obtenir une paire (Y, Z) en appliquant la fonction f à l'exemple X, et aux paramètres $\mathbf{w}_t$,
- Utiliser ce couple (Y, Z) comme exemple, pour appliquer (8.15) et (8.16), et déterminer de nouveaux paramètres $\mathbf{v'}$ pour le système g, et de nouveaux états internes Y'.
- Appliquer une fois (8.13) à l'exemple (X, Y'), pour déterminer de nouveaux paramètres $\mathbf{w'}$ pour le système f.

Cette méthode intuitive, appelée *propagation des états désirés* [1] soulève quelques problèmes sérieux: Le premier consiste à savoir quelles garanties de convergence et de stabilité possède un tel algorithme. Le second est plus fondamental: S'il converge, vers quoi converge-t'il?

Jusqu'à présent, nous avons introduit les algorithmes en définissant une fonction de coût, pour appliquer ensuite un algorithme itératif d'optimisation. L'objectif de cette méthode consiste à atteindre l'optimum d'une mesure d'adéquation de notre système à ses spécifications, c'est à dire à la tâche que l'on attend de lui.

Avec la problématique du credit assignment problem, cet objectif essentiel a été évacué. Dans quelques cas particulier, cependant, on peut définir un coût global.

ii Cas des coûts quadratique.

De nombreux algorithmes connexionnistes consistent à minimiser un coût quadratique par descente de gradient. Dans ce cas, les fonctions de réestimation des paramètres ont la forme

$$\Psi^t_w(X, Y, \mathbf{w}) = \mathbf{w} - \varepsilon_t \nabla_{\mathbf{w}} \left(Y - f(X, \mathbf{w})\right)^2 \qquad (8.17)$$

$$\Psi^t_v(Y, Z, \mathbf{v}) = \mathbf{v} - \varepsilon_t \nabla_{\mathbf{v}} \left(Z - g(Y, \mathbf{v})\right)^2 \qquad (8.18)$$

La fonction symétrique de (8.18) concernant l'adaptation des $\mathbf{Y}$ dérive directement du principe d'optimisation par gradient:

$$\Psi^t_Y(Y, Z, \mathbf{v}) = Y - \varepsilon_t \nabla_Y \left(Z - g(Y, \mathbf{v})\right)^2 \qquad (8.19)$$

Appliquons le principe de propagation des états désirés. On commence par obtenir des Y en appliquant f aux X. Cela permet d'appliquer (8.18) afin d'obtenir la formule d'adaptation des paramètres $\mathbf{v}$,

$$\mathbf{v}' = \mathbf{v} + \varepsilon_t \nabla_{\mathbf{v}} \left(Z - g(f(X,\mathbf{w}),\mathbf{v})\right)^2 \qquad (8.20)$$

et d'obtenir à l'aide de (8.19) une formule de mise à jour des Y.

$$Y' - Y = Y' - f(X, \mathbf{w}) = - \varepsilon_t \nabla_Y \left(Z - g(Y,\mathbf{v})\right)^2 \Big|_{Z, f(X,\mathbf{w}), \mathbf{v}} \qquad (8.21)$$

On peut alors appliquer (8.17), pour obtenir une formule de mise à jour des paramètres $\mathbf{w}$ du premier étage.

---

1 **Le Cun Y.:** *Learning processes in an asymmetric threshold network* - In "Disordered systems and biological organization", NATO ASI series in systems and computer science, n° F20, pp 233-240, Springer Verlag (1986)

$$\mathbf{w'} = \mathbf{w} + 2\,\varepsilon_t\,(Y' - f(X,\mathbf{w}))\bullet\nabla_{\mathbf{w}} f(X,\mathbf{w})$$

$$\mathbf{w'} = \mathbf{w} - 2\,\varepsilon_t^2\,\nabla_Y\,\big(Z - g(Y,\mathbf{v})\big)^2\Big|_{Z,f(X,\mathbf{w}),\mathbf{v}}\bullet\nabla_{\mathbf{w}} f(X,\mathbf{w})$$

$$\mathbf{w'} = \mathbf{w} - 2\,\varepsilon_t^2\nabla_{\mathbf{w}}\big(Z - g(f(X,\mathbf{w}),\mathbf{v})\big)^2 \qquad (8.22)$$

Aux gains près, les formules (8.20) et (8.22) constituent précisément l'algorithme d'optimisation par descente de gradient stochastique (3.6) du critère quadratique unique $E_{X,Z}\big(Z - g(f(X,\mathbf{w}),\mathbf{v})\big)^2$.

8.2.3.2 Rétro-propagation du gradient

C'est ainsi qu'en appliquant les formules (8.10) à des systèmes équivalents à un perceptron multi-couches, on obtient l'algorithme usuel de rétro-propagation du gradient (cf. §2.3.4). Cela se produit en particulier dans les trois cas suivants, qui ne diffèrent que par la finesse de l'utilisation des formules (8.10).

- Lorsque chaque module est une unité sigmoïde, les formules (8.10) sont équivalentes à la règle usuelle de rétro-propagation du gradient.

- Lorsque chaque module est une couche d'unités sigmoïdes, les formules (8.10) décrivent les interactions entre couches, sans exprimer le comportement spécifique de chaque couche.

- Lorsque chaque module est un perceptron multi-couches, les formules (8.10) décrivent les interactions entre sous-réseaux, sans exprimer le comportement exact de ces sous-réseaux.

Le principal intérêt de ces niveaux de représentation est alors la possibilité d'utiliser plusieurs niveaux d'implémentation. Il n'est pas très efficace, en effet, d'utiliser les même programmes pour effectuer une rétro-propagation de gradient entre deux couches totalement connectées (une multiplication de matrices suffit), ou entre deux couches reliées par une toile d'araignée de connexions partagées…

8.3 Illustration.

L'objet de cette section est de montrer comment le formalisme des algorithmes d'apprentissage modulaires permet de construire une grande variété de systèmes, et de mettre en œuvre des systèmes hybrides. Leur implémentation s'en trouve facilitée.

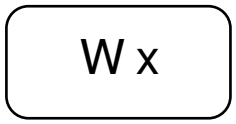
8.3.1 Exemples d'algorithmes modulaires.

Chaque module est entièrement défini par la donnée de dépendance paramétrique entre entrées et sorties $y=f(x)$. Les formules (8.10) permettent alors, à partir des dérivées de f, de calculer les conjugués β_k des entrées, et les dérivées Δ_i des paramètres, à partir des conjugués α_j des sorties.

Voici quelques exemples de modules élémentaires, et de leurs combinaisons possibles.

8.3.1.1 Systèmes linéaires.

Le produit matriciel est l'opération fondamentale de tous les systèmes linéaires, comme le perceptron et l'adaline, ou quasi-linéaires, comme le perceptron multi-couches.



Fonction: $y_i = \sum_k w_{ik} x_k$

Apprentissage: $\beta_k = \sum_i \alpha_i w_{ik}$

$\Delta_{ik} = \alpha_i x_k$

L'adaline, par exemple, utilise un critère de moindres carrés pour effectuer l'apprentissage. Les exemples sont des paires (entrées, sorties désirées). Le module ci-dessous calcule la distance entre sorties obtenues x_k et sorties désirées d_k. Il ne dépend d'aucun paramètre.

Fonction: $J = \sum_k (d_k - x_k)^2$

Apprentissage: $\beta_k = -2\,(d_k - x_k)$

Une adaline est alors représentée par la figure ci après (Fig 8.3). Cela signifie qu'appliquer les formules (8.10) dans le système composé de l'enchaînement de ces deux modules est équivalent à la règle du Delta.

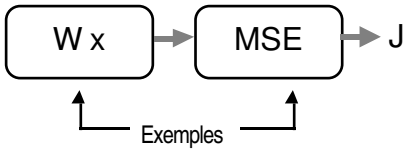


Fig 8.3 - Une adaline.

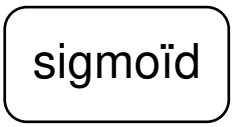
Pour obtenir un perceptron, il suffit de remplacer dans la figure 8.3 le module "MSE" par un module qui reproduit le coût du perceptron. Un tel module peut être décrit de la façon suivante:

Fonction: $J = -\sum_k (d_k - \mathbb{1}_{\Re_+}(x_k))\, x_k$

Apprentissage: $\beta_k = -\,(d_k - \mathbb{1}_{\Re_+}(x_k))$

8.3.1.2 Perceptrons multi-couches.

Il suffit, pour représenter un perceptron multi-couches, d'introduire un nouveau module, qui applique simplement une fonction sigmoïde. La encore, il n'y a pas de paramètres.



Fonction: $y_k = f(x_k)$

Apprentissage: $\beta_k = f'(x_k)\ \alpha_k$

La figure ci dessous (Fig 8.4), par exemple, représente un perceptron à deux couches, totalement connectées.

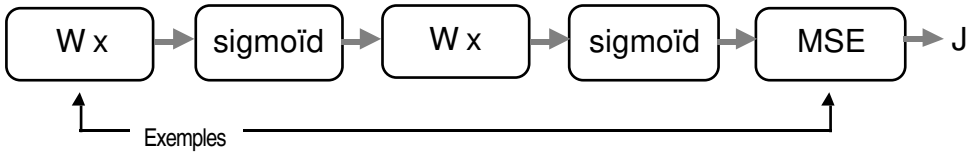


Fig 8.4 - Un perceptron multi-couches.

Bien sûr, les produits matriciels ne représentent pas toutes les architectures de connexions que peut utiliser un perceptron multi-couches. Deux solutions peuvent être envisagées:

- La première consiste à exprimer tout de même nos connexions à l'aide de produits matriciels. Cela est très possible dans le cas de connexions locales à poids partagés, comme dans les réseaux à délais.

  On définit alors un module de convolution à l'aide de modules de produits matriciels, comme le montre la figure ci après (Fig 8.5). Il est alors facile de construire un réseau à délais à l'aide de tels modules de convolution.

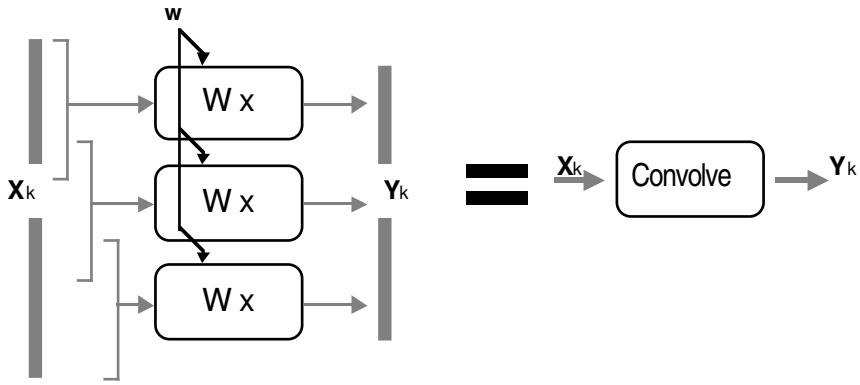


Fig 8.5 - Un module de convolution construit à partir de modules de produits matriciels, pour représenter des connexions locales à poids partagés dans un réseau à délais.

- Une seconde solution consiste à représenter chaque unité sigmoïde par la combinaison d'un module matriciel à une sortie, et d'un module sigmoïde.

On définit alors un module équivalent à une unité quasi-linéaire. Cela permet alors de représenter toute architecture de perceptron multi-couches.

Leurs implémentations constituent une différence essentielle entre ces solutions: il est bien plus efficace de programmer un produit matriciel que les multiples indirections que requièrent les perceptrons multi-couches dans le cas général.

8.3.1.3 Algorithmes utilisant des distances.

Pour représenter des algorithmes utilisant des distances euclidiennes, il est naturel d'introduire un module qui calcule les carrés de ces distances:

$(w - x)^2$

Fonction: $y_j = \sum_k (w_{jk} - x_k)^2$

Apprentissage: $\beta_k = -2 \sum_j \alpha_j (w_{jk} - x_k)$

$\Delta_{jk} = 2 \alpha_j (w_{jk} - x_k)$

En introduisant également un module qui prend le minimum de ces distances, on peut représenter l'algorithme k-means (cf. §3.2.2), comme dans la figure ci dessous (Fig 8.6).

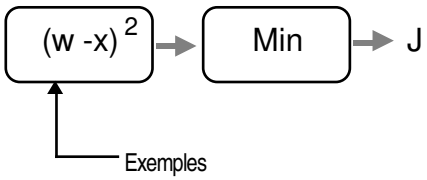


Fig 8.6 - L'algorithme k-means.

Mais il est également possible d'introduire un module qui reproduit l'équation (3.58), ce qui permet alors d'obtenir l'algorithme LVQ2. En notant en notant k^* l'indice du l'entrée la plus faible, et k^{**} l'indice de l'entrée associée à la bonne classe la plus faible,

LVQ 2

Fonction: $J = 0$ si $k^{**}=k^*$, $\mathrm{Max}\left(1, \frac{x_{k^{**}} - x_{k^*}}{\delta x_{k^*}}\right)$ sinon

Apprentissage: si $k^{**} \neq k^*$ et $x_{k^{**}} < (1+\delta) x_{k^*}$

$\beta_{k^{**}} = 1 / \delta x_{k^*}$

$\beta_{k^*} = -(x_{k^{**}} / x_{k^*}) / \delta x_{k^*}$

L'algorithme LVQ2 peut alors être représenté par la figure ci-dessous (Fig 8.7):

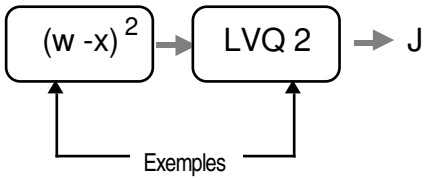


Fig 8.7 - Algorithme LVQ 2 (cf §3.4.2.2)

8.3.1.4 Algorithmes hybrides.

La plupart des algorithmes classiques peuvent ainsi être décomposés en un petit nombre de modules élémentaires. L'intérêt principal d'une telle décomposition est alors la possibilité de construire des algorithmes hybrides, comme ceux suggérés dans [1] (Fig 8.8).

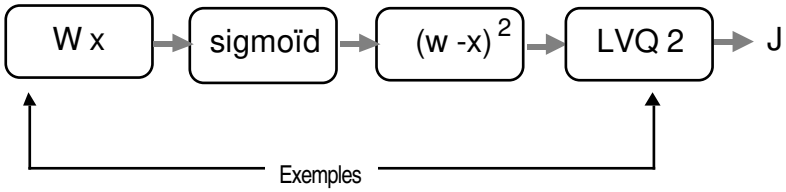


Fig 8.8 - Un système hybride perceptron multi-couches, LVQ 2.

Les formules (8.10) constituent alors un algorithme global pour l'apprentissage d'un tel système hybride.

Chaque algorithme hybride requiert toutefois une étude approfondie. Les problèmes de convergence peuvent être abordés avec les résultats de la section (3.3). Ces résultats utilisent certaines hypothèses sur la forme du coût $C(\mathbf{w})$. L'une d'elle, dans le théorème (3.46) requiert que le coût soit globalement minoré. Cela permet d'éviter à nos paramètres $\mathbf{w}$ de diverger avec des valeurs de plus en plus grandes. Or un empilement quelconque des modules présentés vérifie pas toujours cette hypothèse.

De multiples problèmes pratiques peuvent également apparaître: LVQ2 ne fonctionne efficacement que si les points de référence sont correctement initialisés. On utilise souvent l'algorithme k-means pour cela. Mais cette initialisation dépend alors des valeurs initiales du perceptron multi-couches, qu'il faut alors choisir soigneusement…

8.3.2 Implémentation d'algorithmes modulaires.

Ce formalisme d'algorithmes modulaires d'apprentissage permet d'aborder de façon unifiée un grand nombre d'algorithmes. Il peut donc être mis à profit pour réaliser une implémentation très générale.

---

1 **Bollivier M. de, Gallinari P., Thiria S.:** *Multi-Module Neural Networks for Classification* - Procs of INNC'90, vol 2, pp 777-780, (1990)

Chaque module est défini par trois opérations élémentaires, correspondant à une passe avant, une passe arrière, et le calcul des mises à jour des paramètres. Les formules (8.10) permettent d'enchaîner ces tâches, indépendamment de la nature des modules.

Un langage d'objets permet alors d'implémenter élégamment ces algorithmes. On bénéficie alors de tous les avantages autorisés par les notions d'encapsulation et de modularité. Sans changer l'organisation générale de notre programme, il est possible

- de remplacer un module par une implémentation plus efficace,
- de remplacer un module par un autre plus approprié,
- de définir des modules composés à l'aide d'autre modules,
- d'ignorer le détail de l'implémentation de chaque module.

Une telle implémentation repose sur une description de haut niveau des algorithmes en modules élémentaires. On peut alors à partir de cette description, générer automatiquement programmes ou description de dispositifs électroniques, destinés à assurer soit l'apprentissage, soit l'exploitation de tels systèmes. Cette génération consiste essentiellement à mettre bout à bout des implémentations appropriées de chaque module, et éventuellement à effectuer une optimisation éliminant les calculs inutiles.

Enfin, il est agréable de constater que les opérations les plus lourdes des algorithmes d'apprentissage peuvent s'exprimer en termes de produits matriciels, de convolutions, de calculs de distances. L'intérêt de telles opérations dépasse tant le cadre des algorithmes d'apprentissage, que, sur presque chaque machine, séquentielle ou parallèle, on dispose de bibliothèques optimisées pour les réaliser. Utiliser ces bibliothèques doit être plus efficace que les programmer à l'aide de neurones formels.

8.4 Conclusion.

On pourrait peut-être résumer ce chapitre en rappelant que l'on peut calculer des dérivées à travers n'importe quel système différentiable. Ce serait néanmoins omettre certains aspects capitaux:

- Tout d'abord, ce formalisme apporte une solutions à certains problèmes de coopération de systèmes d'apprentissage, comme ceux posés dans les chapitres suivants par les modèles de Markov discriminants.

- Il fournit également une implémentation simple et générale de systèmes pouvant devenir très complexes.

- Mais surtout, il constitue un jeu de construction d'algorithmes, permettant de construire facilement un système structuré d'apprentissage.

On a souvent, en effet, une idée a priori de la structure d'un système destiné à traiter un problème spécifique. On peut alors concevoir un algorithme adapté à chaque problème.

9 Systèmes connexionnistes et modèles de Markov.

Plusieurs arguments contribuent à l'efficacité des modèles de Markov cachés pour la reconnaissance de la parole (cf chp. 6): Ils permettent aisément d'aborder des problèmes de parole continue; ils permettent également de définir une grammaire probabiliste, simplement en jouant sur leur architecture.

En revanche, ils pèchent par leur manque de capacité discriminante: Plus exactement, approcher les vraisemblances conditionnelles avec précision requiert un système très fin, qui requiert en retour un nombre d'exemples déraisonnable pour réaliser son apprentissage. Or, une imprécision sur les vraisemblances se traduit par un mauvais comportement du système, au voisinage des frontières Bayesiennes de classification.

Or, il semble envisageable [1] de joindre les capacités discriminantes des perceptrons multi-couches, et la souplesse des modèles de Markov cachés.

---

1 **Bourlard H., Wellekens C.J.:** *Links between Markov Models and Multilayer Perceptrons*, Advances in Neural Information Processing Systems 1 - pp 502-510, Morgan-Kaufman. Denver (1989)

9.1 Descente de gradient dans les modèles de Markov.

On montre ici que les formules de Baum peuvent être obtenues au moyen d'un algorithme de gradient adapté. En effet, les probabilités qui paramétrent un modèle de Markov doivent avoir pour somme 1; ces contraintes permettent de définir un choix optimal pour les gains ε.

Il est en effet possible de réaliser l'apprentissage d'un modèle de Markov au moyen d'un *algorithme de gradient*. Il suffit en effet d'appliquer le principe du maximum de vraisemblance (3.64), et d'écrire, en reprenant les notations de la section 6.5,

$$\lambda_{t+1} = \lambda_t + \varepsilon_t \nabla_\lambda \log L_{\lambda_t}(O_1 \dots O_T) = \lambda_t + \varepsilon_t \frac{\nabla_\lambda L_{\lambda_t}(O_1 \dots O_T)}{L_{\lambda_t}(O_1 \dots O_T)} \qquad (9.1)$$

sous réserve d'avoir résolu le problème des contraintes sur les paramètres λ. En effet, ceux ci sont des probabilités; ils doivent donc être positifs, et certaines sommes doivent rester égales à 1.

- Dans la section 9.1.1, on montre, en s'inspirant de [1,2], que le calcul des dérivées de la vraisemblance, peut être effectué récursivement en remontant le temps. Ce calcul est extrêmement comparable au calcul des dérivées pour la rétro-propagation du gradient avec poids partagés.

- La section 9.1.2 est consacrée à l'étude des contraintes imposées par la nature probabiliste de nos paramètres λ. Paradoxalement, ces contraintes permettent d'effectuer un choix optimal pour le gain ε. On retrouve alors les formules de Baum. Cette démonstration des formules de Baum est également connue sous le nom de "EM Algorithm" [3].

Déterminer au moyen d'un algorithme de gradient le maximum de vraisemblance dans un modèle de Markov caché est en fait une très vieille idée. Elle n'était cependant pas à la portée des calculateurs de

---

1 **Kehagias A.:** *Optimal Control for Training: the Missing Link between Hidden Markov Models and Connectionist Networks* - Brown University, Div. of Applied Mathematics Tech. Report. (1989)

2 **Bridle J.S.:** *Alpha-nets: A recurrent "neural" network architecture with a Hidden Markov Model interpretation.* To appear in Speech Communication special NeuroSpeech89 issue. (1990)

3 **Dempster A.P., Laird N.N., Rubin D.B.:** *Maximum Likelihood from Incomplete Data via the EM Algorithm* - Journal of the Royal Statistical Society, Series B, vol 39 n°1, pp 1-38, (1977)

l'époque. Au début des années 70, les formules de réestimation de Baum apportèrent une solution aux problèmes de contraintes, équivalente à un gain optimal dans un algorithme de gradient!

9.1.1 Gradient de la vraisemblance d'une observation.

Commençons par calculer les dérivées de la vraisemblance par rapport aux probabilités de transition a_{ij}. Dans un premier temps, nous supposerons que les probabilités de transition a_{ij} dépendent du temps. Nous les noterons $a_{ij}(t)$. On a alors, d'après (6.18)

$$\frac{\partial L_\lambda(O_1 \dots O_T)}{\partial a_{ij}(t)} = \frac{\partial L_\lambda(O_1 \dots O_T)}{\partial \alpha_t(j)} \frac{\partial \alpha_t(j)}{\partial a_{ij}(t)} = \frac{\partial L_\lambda(O_1 \dots O_T)}{\partial \alpha_t(j)} \alpha_{t-1}(i)\, b_j(O_t) \qquad (9.2)$$

Il nous reste à calculer la dérivée de la vraisemblance par rapport aux $\alpha_t(j)$. On s'appuie pour cela sur l'égalité (6.17)

$$\forall\, t \in \{1 \dots T\},\ L_\lambda(O_1 \dots O_T) = \sum_{j \in S} \alpha_t(j)\, \beta_t(j)$$

qui permet de déduire directement

$$\frac{\partial L_\lambda(O_1 \dots O_T)}{\partial \alpha_t(j)} = \beta_t(j) \qquad (9.3)$$

Enfin, en procédant comme dans le cas de la rétro-propagation à poids partagés (cf. §2.3.4.2.*ii*), on obtient

$$\frac{\partial L_\lambda(O_1 \dots O_T)}{\partial a_{ij}} = \sum_{t=1}^{T} \frac{\partial L_\lambda(O_1 \dots O_T)}{\partial a_{ij}(t)} = \sum_{t=1}^{T} \alpha_{t-1}(i)\ \beta_t(j)\, b_j(O_t) \qquad (9.4)$$

En général, les $b_j(O_t)$ sont eux mêmes des modèles paramétrés, par exemple gaussiens (6.13). On peut calculer aisément les dérivées par rapport aux paramètres μ_{jk} de $b_j(O_t)$, à partir des dérivées par rapport à $b_j(O_t)$, qui s'expriment

$$\frac{\partial L_\lambda(O_1 \dots O_T)}{\partial b_j(O_t)} = \frac{\partial L_\lambda(O_1 \dots O_T)}{\partial \alpha_t(j)} \frac{\partial \alpha_t(j)}{\partial b_j(O_t)} = \beta_t(j) \sum_{i \in S} a_{ij}\, \alpha_{t-1}(i) \qquad (9.5)$$

On en déduit alors

$$\frac{\partial L_\lambda(O_1 \dots O_T)}{\partial \mu_{jk}} = \sum_{t=1}^{T} \frac{\partial L_\lambda(O_1 \dots O_T)}{\partial b_j(O_t)} \frac{\partial b_j(O_t)}{\partial \mu_{jk}}$$

$$= \sum_{t=1}^{T} \frac{\partial \log(b_j(O_t))}{\partial \mu_{jk}} b_j(O_t)\ \beta_t(j) \sum_{i \in S} a_{ij}\ \alpha_{t-1}(i)$$

par (6.18)
$$= \sum_{t=1}^{T} \alpha_t(j)\ \beta_t(j)\ \frac{\partial \log(b_j(O_t))}{\partial \mu_{jk}} \qquad (9.6)$$

Dans le cas gaussien, par exemple, on déduit de (6.13)

$$\log(b_j(O_t)) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\mathrm{Det}(\Sigma_j^{-1})\right) - \frac{1}{2}(O_t - \mathbf{m}_j)^t \Sigma_j^{-1}(O_t - \mathbf{m}_j)$$

On peut alors déterminer les dérivées de la vraisemblance $L_\lambda(O_1 \ldots O_T)$ par rapport à $\mathbf{m}_j(k)$, $k^{\text{ième}}$ composante de $\mathbf{m}_j$, et par rapport à $\mathbf{C}_j(i,k)$ élément d'indice (i,k) de la matrice Σ_j^{-1}:

$$\frac{\partial L_\lambda(O_1 \ldots O_T)}{\partial \mathbf{m}_j(k)} = \sum_{t=1}^{T} \alpha_t(j)\ \beta_t(j)\ \left(O_t(k) - \mathbf{m}_j(k)\right) \sum_{i=1}^{n} \mathbf{C}_j(i,k)$$

$$\frac{\partial L_\lambda(O_1 \ldots O_T)}{\partial \mathbf{C}_j(i,k)} = \sum_{t=1}^{T} \alpha_t(j)\ \beta_t(j)\ \left(\Sigma_j(i,k) - (O_t(i) - \mathbf{m}_j(i))(O_t(k) - \mathbf{m}_j(k))\right) \qquad (9.7)$$

Dans cette dernière formule, on a utilisé l'égalité suivante, où D_{ik} représente le cofacteur de $\mathbf{C}_j(i,k)$.

$$\frac{\partial \log\left(\mathrm{Det}(\Sigma_j^{-1})\right)}{\partial \mathbf{C}_j(i,k)} = \frac{1}{\mathrm{Det}(\mathbf{C}_j)} \frac{\partial \mathrm{Det}(\mathbf{C}_j)}{\partial \mathbf{C}_j(i,k)} = \frac{D_{ik}}{\mathrm{Det}(\mathbf{C}_j)} = \Sigma_j(i,k)$$

Cette égalité repose sur les deux résultats classiques d'algèbre linéaire suivants

$$\Sigma(i,k) = \mathbf{C}^{-1}(i,k) = \frac{D_{ik}}{\mathrm{Det}(\mathbf{C})}, \textit{ et } \quad \mathrm{Det}(\mathbf{C}) = \sum_i \mathbf{C}(i,k).D_{ik}$$

Le calcul des dérivées de $L_\lambda(O_1 \ldots O_T)$ s'effectue donc en deux opérations:

- *Passe avant* : On calcule récursivement les $\alpha_t(j)$ à l'aide de la formule (6.18).
- *Passe arrière* : On calcule récursivement les quantités conjuguées $\beta_t(j)$, à l'aide de la formule (6.19). Ce calcul se fait en remontant le temps, d'où son nom. On peut alors calculer les dérivées à l'aide des formules (9.2) et (9.7).

On ne peut cependant appliquer l'algorithme (9.1) simplement: En effet, certaines contraintes, imposées par la nature probabiliste des paramètres, doivent être respectées. En premier lieu, les probabilités de transition doivent être positives, leur somme doit être 1:

$$\forall i,j \in S,\ 0 \le a_{ij} \le 1, \qquad \forall i \in S,\ \sum_{j \in S} a_{ij} = 1$$

De plus, la forme des $b_j(O_t)$ peut imposer des contraintes sur leurs paramètres. Nous supposerons dans la suite que ces contraintes sont de même nature que les contraintes sur les probabilités de transition.

9.1.2 Résolution des contraintes.

Dans cette section, nous noterons simplement L le logarithme de la vraisemblance $\log L_\lambda(O_1 \dots O_T)$. De plus, nous noterons $\lambda=(\lambda_{jk})$ nos paramètres, en les indiçant selon leurs contraintes:

$$\forall\ j,k,\ \ 0 \le \lambda_{jk} \le 1;\ \ \forall\ j,\ \sum_k \lambda_{jk} = 1 \tag{9.8}$$

On se propose maintenant d'étudier l'emploi d'un algorithme de gradient pour réestimer les λ, et de comparer cette approche avec les formules de Baum. Appliquer un algorithme de gradient correspond à réestimer λ de la façon suivante, comparable à (9.1):

$$\lambda^*_{ij} = \lambda_{ij} + \varepsilon_{ij} \frac{\partial L}{\partial \lambda_{ij}}$$

9.1.2.1 Projection sur l'espace des contraintes.

Cependant, rien ne garantit alors que λ^* respecte les contraintes (9.8). Une solution pour garantir la satisfaction des contraintes consiste à réestimer λ en *projetant* λ^* *sur l'espace des contraintes*:

$$\lambda^*_{ij} = \lambda_{ij} + \varepsilon_{ij} \frac{\partial L}{\partial \lambda_{ij}}, \quad \textit{et} \quad \lambda'_{ij} = \frac{\lambda^*_{ij}}{\sum_k \lambda^*_{ik}} \tag{9.9}$$

Quels sont alors les ε_{ij} admissibles, garantissant que $L(\lambda') \ge L(\lambda)$? Dans le cas présent, où L est une fonction strictement croissante d'un polynôme à coefficients positifs, le théorème suivant apporte une réponse surprenante.

Théorème (Soules, [1])

Si L *est une fonction strictement croissante d'un polynôme à coefficients positifs,*

$$\forall \varepsilon > 0, \quad \varepsilon_{ij} = \varepsilon \lambda_{ij} \quad \Rightarrow \quad L(\lambda') > L(\lambda),$$

sauf sur le maximum où l'on a $\lambda = \lambda'$ *et donc* $L(\lambda') = L(\lambda)$.

Lorsque ε *tend vers l'infini, cette inégalité stricte est préservée.* (9.10)

En particulier, dans le cas où ε tend vers l'infini, on obtient les formules de réestimation suivantes:

$$\lambda'_{ij} = \frac{\lambda_{ij} \frac{\partial L}{\partial \lambda_{ij}}}{\sum_k \lambda_{ik} \frac{\partial L}{\partial \lambda_{ik}}} \tag{9.11}$$

Ces réestimations sont une *forme générale des formules de Baum* (6.23) et (6.24).

Nous allons tout d'abord démontrer directement la validité de la formules (9.11), dans le cas où L est une somme de monômes à coefficients positifs c_t:

$$L = \sum_t \quad c_t \prod_{(i,j) \in U_t} \lambda_{ij} \tag{9.12}$$

où U_t est un n-uplet de couples (i,j) caractérisant chaque monôme. La technique utilisée, dite de la fonction auxiliaire, est la même que celle introduite dans la section 6.5. On définit en effet alors une fonction auxiliaire Q:

$$Q(\lambda,\lambda') = \sum_t \quad c_t \left(\prod_{(i,j) \in U_t} \lambda_{ij} \right) \log \left(\prod_{(i,j) \in U_t} \lambda'_{ij} \right) \tag{9.13}$$

Grâce à l'inégalité $\log(x) < x - 1$, on vérifié aisément que

$$Q(\lambda,\lambda') - Q(\lambda,\lambda) \leq L(\lambda') - L(\lambda) \tag{9.14}$$

De plus, cette inégalité est stricte si $\lambda \neq \lambda'$. Cherchons maintenant le maximum de $Q(\lambda,\lambda')$. Posons le Lagrangien:

$$E = Q(\lambda,\lambda') - \sum_i \mu_i \left(\sum_j \lambda'_{ij} - 1 \right) \tag{9.15}$$

---

1 **Soules G.W.:** *A gradient derivation of PTAH* - IDA-CRD Tech Report N°80328, (september 1977)

et annulons de ses dérivées. En définissant $\theta_{ij}(U_t)$ comme le nombre d'occurrences de (i,j) dans le n-uplet U_t, on peut exprimer la dérivée du Lagrangien (9.15).

$$\frac{\partial E}{\partial \lambda'_{ij}} = -\mu_i + \sum_t c_t \frac{\theta_{ij}(U_t)}{\lambda'_{ij}} \prod_{(k,q)\in U_t} \lambda_{kq} \qquad (9.16)$$

En dérivant (9.12), on remarque

$$\frac{\partial L}{\partial \lambda_{ij}} = \sum_t c_t \frac{\theta_{ij}(U_t)}{\lambda_{ij}} \prod_{(k,q)\in U_t} \lambda_{kq} \qquad (9.17)$$

En conséquence, en annulant la dérivée (9.16) du Lagrangien, et en remplaçant par (9.17), on obtient

$$\mu_i \lambda'_{ij} = \lambda_{ij} \frac{\partial L}{\partial \lambda_{ij}} \qquad (9.18)$$

La contrainte $\sum_j \lambda'_{ij} = 1$ permet de calculer $\mu_i = \sum_j \lambda_{ij} \frac{\partial L}{\partial \lambda_{ij}}$. On en déduit alors (9.11).

$$\lambda'_{ij} = \lambda_{ij} \frac{\partial L}{\partial \lambda_{ij}} \left(\sum_j \lambda_{ij} \frac{\partial L}{\partial \lambda_{ij}} \right)^{-1}$$

Comme cette formule donne l'expression de λ' correspondant au maximum de Q, elle vérifie en particulier $Q(\lambda,\lambda') \geq Q(\lambda,\lambda)$ et donc $L(\lambda') \geq L(\lambda)$, d'après (9.15). Cette inégalité est stricte si $\lambda' \neq \lambda$.

Si φ est une fonction strictement croissante, on a également $\varphi(L(\lambda')) \geq \varphi(L(\lambda))$, ce qui nous permet d'appliquer (9.11) à toute fonction croissante d'un polynôme à coefficients positifs.

L'image par notre projection centrale sur les contraintes de la demi-droite décrite par les λ^* lorsque ε décrit $[0,+\infty[$ est le segment $[\lambda, \lambda^{opt}[$ de l'espace des contraintes, où λ^{opt} est obtenu à l'aide des formules (9.11). La concavité de la fonction Q nous permet de conclure, sauf sur le maximum où on a égalité, que

$$\forall\, \lambda' \in [\lambda, \lambda^{opt}] \,, \quad Q(\lambda,\lambda') > Q(\lambda,\lambda) \textit{ et donc } L(\lambda') > L(\lambda)$$

Ce qui démontre le théorème.

9.1.2.2 Application aux modèles de Markov.

i Formules de Baum.

L'expression (6.16) de la vraisemblance montre que $\log L_\lambda(O_1 \dots O_T)$ est bien une fonction strictement croissante d'un polynôme à coefficients positifs.

En reprenant les dérivées (9.4) de $L_\lambda = L_\lambda(O_1 \dots O_T)$, on obtient

$$\frac{\partial \log L_\lambda}{\partial a_{ij}(t)} = \frac{1}{L_\lambda} \sum_{t=1}^{T} \alpha_{t-1}(i)\beta_t(j)\, b_j(O_t) \tag{9.19}$$

En appliquant les réestimation (9.11), et en simplifiant le dénominateur avec la récurrence (6.19), on retrouve la formule de Baum pour la réestimation des probabilités de transition a_{ij}.

$$a'_{jk} = \frac{\sum_{t=1}^{T} \alpha_{t-1}(j)\, a_{jk}\, b_k(O_t)\, \beta_t(k)}{\sum_{t=1}^{T} \alpha_{t-1}(j)\, \beta_{t-1}(j)} \tag{9.20}$$

De même, avec dérivées (9.7) du cas gaussien, on calcule

$$\frac{\partial \log L_\lambda}{\partial \mathbf{m}_j(k)} = \frac{1}{L_\lambda} \sum_{t=1}^{T} \alpha_t(j)\beta_t(j) \left(O_t(k) - \mathbf{m}_j(k)\right) \sum_{i=1}^{n} \mathbf{C}_j(i,k)$$

$$\frac{\partial \log L_\lambda}{\partial \mathbf{C}_j(i,k)} = \frac{1}{L_\lambda} \sum_{t=1}^{T} \alpha_t(j)\beta_t(j) \left(\Sigma_j(i,k) - (O_t(i) - \mathbf{m}_j(i))(O_t(k) - \mathbf{m}_j(k))\right) \tag{9.21}$$

Il suffit alors de résoudre analytiquement les deux équations

$$\frac{\partial L_\lambda(O_1 \dots O_T)}{\partial \mathbf{m}_j(k)} = 0 \quad et \quad \frac{\partial L_\lambda(O_1 \dots O_T)}{\partial \mathbf{C}_j(i,k)} = 0$$

pour retrouver les formules de Baum pour les paramètres des probabilités d'émission:

$$\mathbf{m}'_i = \frac{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)\, O_t}{\sum_{t=1}^{T} \alpha_t(i)\, \beta_t(i)}, \quad \Sigma'_i = \frac{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)\, O_t\, O_t^T}{\sum_{t=1}^{T} \alpha_t(i)\, \beta_t(i)} - \mathbf{m}'_i\, \mathbf{m}'^T_i \tag{9.22}$$

Ces résultats rattachent d'une certaine façon les formules de Baum à la famille des algorithmes d'optimisation par gradient. Cependant, elles correspondent soit à un *gain "infini",* soit à une *solution analytique* de notre problème d'optimisation.

ii Formules de Baum avec plusieurs exemples.

Les formules ci dessus s'appliquent pour maximiser la vraisemblance L_λ associée à un seul exemple: la séquence d'observations $(O_1 \ldots O_T)$.

En pratique, on recherche le maximum de la vraisemblance L_λ pour une famille de n observations $\{(O^k_1 \ldots O^k_T)\}$. On cherche donc à optimiser une fonctionnelle de la forme (3.64):

$$\tilde{C}^{mle}(\lambda) = \log \prod_{k=1}^{n} L_\lambda(O^k_1 \ldots O^k_T) = \sum_{k=1}^{n} \log L_\lambda(O^k_1 \ldots O^k_T) \qquad (9.23)$$

On dispose donc de deux algorithmes de gradient, l'algorithme déterministe (3.5') et l'algorithme stochastique (3.6).

Les dérivées de $\tilde{C}^{mle}(\lambda)$ sont évidemment les sommes sur tous les exemples des dérivées (9.19) et (9.21). $\tilde{C}^{mle}{}_n(\lambda)$ est une fonction strictement croissante du produit des vraisemblances, polynômes à coefficients positifs. On peut donc appliquer (9.11), et calculer des formules de Baum pour l'algorithme déterministe.

Par exemple, la formule de réestimation des probabilités de transition est:

$$a'_{jk} = \frac{\sum_{k=1}^{n} \frac{1}{L_\lambda(O^k_1 \ldots O^k_T)} \sum_{t=1}^{T} \alpha^k_{t-1}(j)\, a_{jk}\, b_k(O_t)\, \beta^k_t(k)}{\sum_{k=1}^{n} \frac{1}{L_\lambda(O^k_1 \ldots O^k_T)} \sum_{t=1}^{T} \alpha^k_{t-1}(j)\, \beta^k_{t-1}(j)}) \qquad (9.24)$$

où $\alpha^k_t(i)$ et $\beta^k_t(i)$ sont les valeurs de $\alpha_t(i)$ et $\beta_t(i)$ pour l'exemple k.

L'algorithme stochastique consiste en toute rigueur à appliquer (9.1). Rien ne garantit, en revanche, que l'on puisse appliquer stochastiquement les formules de Baum: La démonstration de la section 3.3 ne s'applique pas à de tels gains "infinis".

9.2 Modèles de Markov discriminants.

Contrairement aux modèles de Markov cachés usuels, qui estiment les vraisemblances conditionnelles, au moyen du principe du maximum de vraisemblance (MLE), les modèles de Markov discriminants permettent d'estimer les probabilités a posteriori de mots ou de séquences de mots, à l'aide du principe du maximum de probabilité a posteriori (MAP).

Leurs paramètres sont déterminés par des probabilités conditionnelles de transition, qui peuvent être générées de plusieurs façons par un perceptron multi-couches.

9.2.1 Capacité discriminante.

L'algorithme d'apprentissage des modèles de Markov décrits dans la section 6.5 consiste donc à applique le principe du maximum de vraisemblance au moyen d'un algorithme de gradient raffiné.

On a plusieurs fois souligné que l'apprentissage de ces modèles est *peu discriminant*. Chaque modèle est en effet entraîné indépendamment; les vraisemblances $L_\lambda(O_1 \dots O_T \mid W_k)$ mesurent alors l'adéquation du signal au modèle W_k.

Un tel système cherche à approcher au mieux la vraisemblance pour toutes les observations. Or, une faible précision sur la vraisemblance suffit pour classer correctement des observations peu ambiguës. A l'opposé, une grande précision est nécessaire pour classer des observations ambiguës.

Une partie de la capacité de notre modèle paramétrique de la vraisemblance conditionnelle est donc utilisée pour améliorer inutilement la précision de l'estimation. Pour capter autant d'information utiles, cette capacité doit être plus élevée, et nécessite plus d'exemples.

Il est intuitivement plus aisé d'apprendre ce qu'est un mot, en le comparant à d'autres, et en relevant des *caractéristiques discriminantes*, c'est à dire en identifiant des traits permettant de distinguer deux mots.

Un système parfaitement discriminant n'apprend en fait que les frontières de séparation entre classes. Il s'agit donc d'une minimisation du risque moyen, qu'il est malheureusement impossible de réaliser directement (cf. §3.4). C'est en s'écartant de cette minimisation idéale que nos systèmes d'apprentissage perdent leur *capacité discriminante*.

9.2.2 Evaluation des probabilités a posteriori dans un modèle de Markov caché.

Au lieu d'évaluer les vraisemblances conditionnelles $L(O_1 \dots O_T | W_k)$, il est possible [1] d'estimer les probabilités a posteriori $P(W_k | O_1 \dots O_T)$ dans un modèle de Markov caché. On compte alors utiliser le principe du maximum de probabilité a posteriori (MAP, cf. §3.4.4.3.*i*) au lieu du traditionnel maximum de vraisemblance (MLE).

---

1 **Devijver P.A:** *Baum's forward-backward algorithm revisited* - Pattern Recognition Letters 3 pp 369-373 - North Holland (1985).

De façon analogue au calcul des vraisemblances (cf. §6.5.3.2), on calcule la quantité $P_\lambda(W_k|O_1\dots O_T)$ de la façon suivante,

$$P_\lambda(W_k|\, O_1\dots O_T) = \sum_{j\in S} P_\lambda(W_k,\, s_t=j \mid O_1\dots O_T) \qquad \forall t\in [1,T\,] \tag{9.25}$$

que l'on peut décomposer de la façon suivante, en utilisant les hypothèses habituelles sur les sources de Markov:

$$\begin{aligned} P_\lambda(s_t=j,\, W_k|\, O_1\dots O_T) &= \frac{P_\lambda(s_t=j,\, W_k,\, O_{t+1}\dots O_T \mid O_1\dots O_t)}{P_\lambda(O_{t+1}\dots O_T \mid O_1\dots O_t)} \\ &= P_\lambda(s_t=j,\, W_k \mid O_1\dots O_t)\, \frac{P_\lambda(O_{t+1}\dots O_T \mid s_t=j,\, W_k)}{P_\lambda(O_{t+1}\dots O_T \mid O_1\dots O_t)} \\ &\stackrel{\Delta}{=} \alpha^d_t(j)\ \beta^d_t(j) \end{aligned} \tag{9.26}$$

Quelques manipulations permettent de déterminer des formules de récurrences pour $\alpha^d_t(j)$ et $\beta^d_t(j)$ comparables à (6.18) et (6.19), à un coefficient de normalisation près.

$$\begin{aligned} \alpha^d_t(j) &= N_t \sum_{i\in S} \alpha^d_{t-1}(i) b_j(O_t) a_{ij} \\ \beta^d_t(j) &= N_t \sum_{k\in S} \beta^d_{t+1}(i) b_k(O_{t+1}) a_{jk} \end{aligned} \qquad \textit{avec } N_t \textit{ tel que } \sum_{i\in S} \alpha^d_t(i) = 1$$

On reconnaît les formules de récurrences (6.25) s'appliquant aux grandeurs $\alpha'_t(j)$ et $\beta'_t(j)$ proposées par Levinson pour réduire les problèmes numériques. En vérifiant que les conditions initiales sont identiques, on conclut que

$$\alpha'_t(j)=\alpha^d_t(j) \quad \textit{et} \quad \beta'_t(j)=\beta^d_t(j) \tag{9.27}$$

On a également vu (cf. § 6.5.4.2) que l'introduction du coefficient de normalisation N_t ne modifie en rien les formules de Baum lorsque l'on remplace $\alpha_t(j)$ et $\beta_t(j)$ par les grandeurs $\alpha'_t(j)=\alpha^d_t(j)$ et $\beta'_t(j)=\beta^d_t(j)$.

Les maximisations de $L_\lambda(O_1\dots O_T|W_k)$ ou de $P_\lambda(W_k|O_1\dots O_T)$, dans lesquelles $(O_1\dots O_T)$ est un exemple du mot ω_k, conduisent donc aux *même formules de réestimation* des probabilités de transition et des paramètres des probabilités d'émission. Faut-il conclure que l'approche ci dessus ne change rien ?

Ce serait oublier bien vite la contrainte:

$$\sum_i P_\lambda(W_i|\, O_1\dots O_T) = 1 \tag{9.28}$$

La modification des paramètres du modèle W_k doit être *complétée par une modification des paramètres de tous les autres modèles*, de façon à respecter cette contrainte (9.28). Cette contrainte est d'ailleurs une condition nécessaire d'application du principe du maximum de probabilité a posteriori (3.72).

C'est dans cette seule contrainte (9.28) qu'est exprimée toute la différence de capacité discriminante entre l'évaluation des vraisemblances $L(O_1 \ldots O_T|W_k)$, et l'évaluation des probabilités a posteriori $P(W_k|O_1 \ldots O_T)$.

9.2.2.1 Probabilités conditionnelles de transition.

Pour pouvoir appliquer sainement le principe du maximum de probabilité a posteriori, il faut donc trouver une expression paramétrique $P_\lambda(W_k| O_1 \ldots O_T)$ pour laquelle la contrainte (9.28) est remplie pour toute les valeurs du paramètre λ.

Une solution consiste à décomposer les probabilités a posteriori comme suit:

$$P(W_k| O_1 \ldots O_T) = \sum_{(s_1 \ldots s_T) \in \Gamma_k} P(s_1 \ldots s_T \mid O_1 \ldots O_T) \qquad (9.29)$$

où Γ_k représente l'ensemble des séquences d'états autorisées pour le modèle W_k. Pour écrire cette égalité, on a supposé que *la connaissance d'une séquence d'états permet de déterminer le modèle qui l'a produit*. Dans cette optique, un modèle est déterminé par l'ensemble des séquences d'états qu'il autorise.

Chaque terme de cette somme peut être factorisé de la façon suivante

$$P(s_1| O_1 \ldots O_T)\, P(s_2 \mid O_1 \ldots O_T, s_1) \ldots P(s_T \mid O_1 \ldots O_T, s_1 \ldots s_{T-1}) \qquad (9.30)$$

Cette expression peut être simplifiée si l'on suppose que *l'état courant ne dépend que de l'état immédiatement antérieur et l'observation courante*. C'est une hypothèse de type Markovien, c'est à dire

$$\textit{Hypothèse:}\quad P(s_t \mid O_1 \ldots O_T, s_1 \ldots s_{t-1}) = P(s_t \mid s_{t-1}, O_t) \qquad (9.31)$$

Et donc:

$$P(W_k| O_1 \ldots O_T) = \sum_{(s_1 \ldots s_T) \in \Gamma_k} \prod_{t=1}^{T} P(s_t \mid s_{t-1}, O_t) \qquad (9.32)$$

en utilisant la convention d'écriture $P(s_1|s_0,O_1) = P(s_1|O_1)$.

On peut donc utiliser un modèle paramétré $P_\lambda(s_t|s_{t-1},O_t)$ des probabilités conditionnelles de transition $P(s_t|s_{t-1},O_t)$ pour construire à l'aide de (9.32) un modèle paramétré $P_\lambda(W_k| O_1 \ldots O_T)$ des probabilités a posteriori.

Si la condition de normalisation sur les probabilités conditionnelles de transition

$$\sum_{j \in S} P_\lambda(s_t=j \mid s_{t-1}, O_t) = 1 \qquad (9.33)$$

est remplie, (9.32) nous permet d'écrire, en notant $\Gamma = \bigcup_k \Gamma_k$:

$$\sum_k P_\lambda(W_k| O_1 \ldots O_T) = \sum_{(s_1 \ldots s_T) \in \Gamma} \prod_{t=1}^{T} P_\lambda(s_t \mid s_{t-1}, O_t)$$

$$= \sum_{k_1 \in S} P_\lambda(s_1=k_1|O_1) \left(\sum_{k_2 \in S} P_\lambda(s_2=k_2|s_1=k_1, O_2) \ldots \left(\sum_{k_T \in S} P_\lambda(s_T=k_T|s_{T-1}=k_{T-1}, O_T) \right) \ldots \right)$$

$$= 1$$

La contrainte (9.28) est donc automatiquement satisfaite.

9.2.2.2 Utilisation d'un perceptron multi-couches.

De plus, en variant la nature de l'hypothèse (9.31), il est possible d'utiliser des probabilités conditionnelles de transition de formes différentes:

i) $P(s_t|s_1 \ldots s_{t-1},O_1 \ldots O_T) = P(s_t|s_{t-1},O_t)$

C'est l'hypothèse Markovienne la plus simple, qui a été utilisée ci dessus pour présenter la décomposition des probabilités a posteriori à l'aide de probabilités conditionnelles de transition.

ii) $P(s_t|s_1 \ldots s_{t-1},O_1 \ldots O_T) = P(s_t|s_{t-q} \ldots s_{t-1},O_t)$

La dépendance avec l'état précédent est étendue ici aux q états les plus récents: C'est un modèle de Markov d'ordre q. On a bien sûr conservé une dépendance avec l'observation courante O_t.

iii) $P(s_t|s_1 \ldots s_{t-1},O_1 \ldots O_T) = P(s_t|O_{t-p} \ldots O_t)$

On remplace ici la dépendance avec l'état précédent par une dépendance avec un certain contexte de longueur p sur les observations. Ce n'est plus, strictement, un modèles de Markov caché, mais le même formalisme s'applique.

Si on se contente, pour effectuer la classification, de déterminer le mot auquel est associé la probabilité a posteriori la plus élevée, on obtient un système comparable à celui de la figure 7.2.

Si on utilise une variante adaptée de l'algorithme de Viterbi pour trouver la séquence d'états la plus probable, et donc le mot (ou la séquence de mots) correspondant, on obtient un système comparable à celui de la figure 7.3.

iv) $P(s_t|s_{1\ldots}s_{t-1},O_1\ldots O_T) = P^t{}_{s_t} = P(s_t|O_t, P^{t-1}{}_k)$

On suppose cette fois ci que les probabilités conditionnelles de transition ne dépendent que de l'observation courante, et de toutes les probabilités conditionnelles à l'instant précédent.

Un modèle paramétrique des $P_\lambda(s_t|O_t, P^{t-1}{}_k)$ nous fournit alors une définition *récurrente* des probabilités conditionnelles de transition.

Ces hypothèses, exceptée l'hypothèse iv) peuvent être regroupées sous la forme suivante

$$P(s_t|s_{1\ldots}s_{t-1},O_1\ldots O_T) = P(s_t|s_{t-q\ldots}s_{t-1},O_{t-p}\ldots O_t) \quad p\geq 1,\ q\geq 0 \qquad (9.34)$$

Quelle que soit l'hypothèse choisie, il est difficile de trouver une expression paramétrique exploitable des probabilités conditionnelles de transition. Bourlard et Wellekens [1] ont suggéré d'utiliser pour cela un *perceptron multi-couches* (Fig 9.1, 9.2 et 9.3), accentuant encore la ressemblance avec les systèmes coopérants décrits dans la section 7.1.

Voici quelques possibilités:

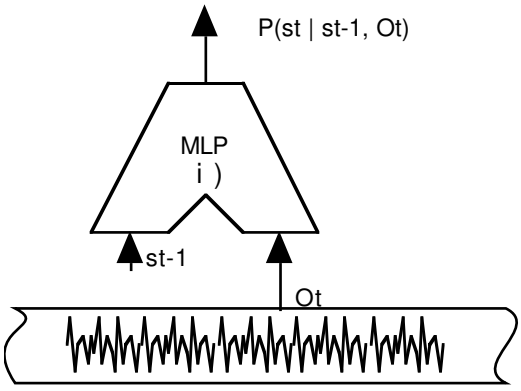


Fig 9.1 - Un perceptron multi-couches pour évaluer des probabilités de transition du type iii)

- Avec l'hypothèse i), on obtient le système étudié par Bourlard et Wellekens dans l'article cité. Nous savons que l'algorithme de rétro-propagation du gradient minimise un critère de moindres carrés. Il évalue donc les probabilités conditionnelles de chaque classe sachant ses entrées, ici l'état antérieur et l'observation courante.

---

1 **Bourlard H., Wellekens C.J.:** *Links between Markov Models and Multilayer Perceptrons*, Advances in Neural Information Processing Systems 1 - pp 502-510, Morgan-Kaufman. Denver (1989)

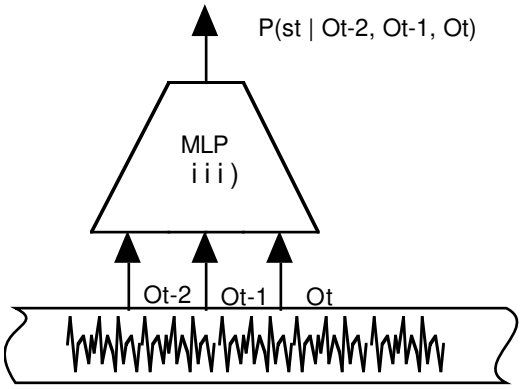


Fig 9.2 - Un perceptron pour évaluer des probabilités de transition du type iii)

- Avec l'hypothèse iii), on obtient un système qui travaille sur une fenêtre glissante du signal.

  Si, au lieu d'utiliser l'expression (9.32), on utilise une régression linéaire pour déterminer les probabilités a posteriori des mots, on obtient un *réseau à délais*.

  Si on utilise une variante adaptée de l'algorithme de Viterbi pour trouver la séquence d'états la plus probable, et donc le mot (ou la séquence de mots) correspondant, on obtient un tel *système hybride réseau à délais + alignement temporel*, comme celui décrit dans la figure 7.3 (cf. §7.2.3.4).

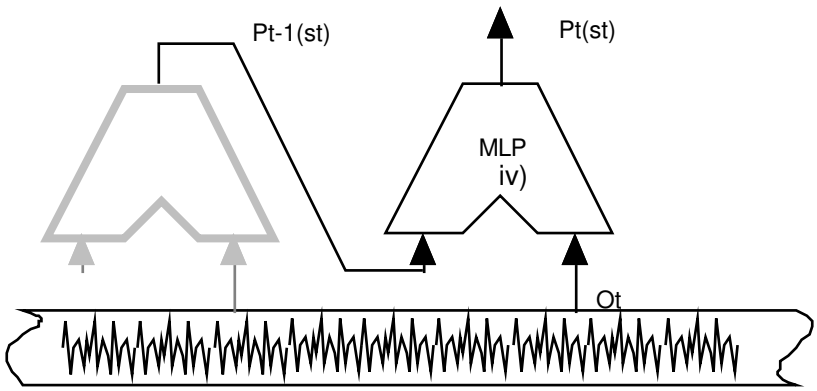


Fig 9.3 - Un perceptron pour évaluer des probabilités de transition du type iv).

- Avec l'hypothèse iv), on obtient simplement un réseau récurrent, comparable à ceux introduits dans la section 7.2.2.

Le formalisme des modèles de Markov discriminants permet donc d'aborder théoriquement le principaux types de systèmes connexionnistes pour la reconnaissance de la parole.

La faiblesse de cette approche réside dans la signification probabiliste supposée des états internes de ces systèmes.

- D'une part, cela impose de respecter la contrainte suivante:

$$\sum_{s_t \in S} P_\lambda(s_t \mid s_{t-1}, O_t) = 1$$

- D'autre part, on ne dispose jamais d'assez d'exemples pour estimer avec précision ces probabilités. En fait, les algorithmes de reconnaissance des formes les plus discriminants ne consistent pas à approcher au mieux les probabilités a posteriori ou les vraisemblances (cf §3.4).

9.2.3 Mise en œuvre.

9.2.3.1 Apprentissage.

Ces dernières remarques soulèvent la question de l'entraînement de tels systèmes. Il suffit a priori d'utiliser le principe des moindres carrés et la rétro-propagation du gradient pour entraîner nos perceptrons multi-couches à évaluer les probabilités conditionnelles de transition.

Il suffit alors de disposer d'exemples constitués d'une observation O_t (ou d'une courte séquence d'observations), de l'état antérieur s_{t-1} (ou des états antérieurs, ou encore du vecteur de probabilités des états antérieurs), et de l'état courant s_t.

Or, il est difficile de se procurer de tels exemples; cela suppose que l'on sait repérer dans le signal l'occurrence d'un état, c'est à dire que nos états correspondent à une événement phonétique. Il est alors possible de segmenter et identifier ces événements sur un signal donné, et l'on dispose alors facilement d'exemples coûteux, mais adéquats.

i Algorithme MAP pour modèles de Markov discriminants.

Une autre solution consiste à entraîner nos perceptrons multi-couches au travers de notre modèle de Markov discriminant, avec seulement des exemples composés d'un signal, et de la séquence de mots qu'il contient.

On peut en effet construire un algorithme d'apprentissage par maximum de probabilité a posteriori (MAP). Reprenant le formalisme du chapitre 8, on définit un module représentant une transition d'un modèle de Markov. Il fournit les probabilités futures y_k des états à partir des probabilités antérieures des états x_k, et des probabilités de transitions P_{ik}.

Fonction: $y_k = \sum_i P_{ik}\, x_i$

Apprentissage: $\beta_{x_i} = \sum_k P_{ik}\, \alpha_k$, $\beta_{P_{ik}} = x_i\, \alpha_k$

Pour appliquer le principe du maximum de probabilité à posteriori, il faut garantir que la somme des estimations des probabilités a posteriori est 1, et qu'elles sont positives. Nous avons vu (9.33) qu'il

suffisait pour cela que la somme des estimations des probabilités de transition soit 1. On glisse donc un module de normalisation inspiré de [1] à la sortie de nos perceptrons multi-couches.

Softmax

Fonction: $$y_k = \frac{e^{x_k}}{\sum_i e^{x_i}}$$

Apprentissage: $$\beta_k = (\alpha_k - \sum_i y_i \alpha_i) \; y_k$$

Enfin, définissons le module définissant un coût correspondant au principe du maximum de probabilité à posteriori (cf. §3.4.4.3.*i*).

MAP

Fonction: $$y_k = \log (x_{k^*})$$

Apprentissage: $$\beta_k = 0 \quad \text{si } k \neq k^*$$
$$= 1/x_k \quad \text{si } k = k^*$$

où k^* désigne l'indice de la classe associée à l'exemple courant.

Il suffit, comme dans la figure 9.4, d'agencer ces modules pour reproduire le système composé par un modèle de Markov discriminant et par son générateur de probabilités conditionnelles de transition. Les formules (8.10) fournissent alors un algorithme d'apprentissage global.

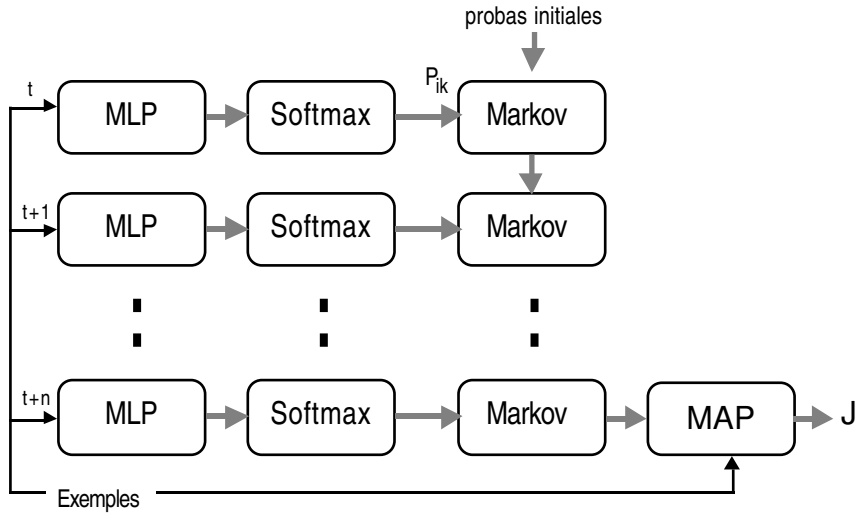


Fig 9.4 - Algorithme d'apprentissage global pour un modèle de Markov discriminant.

1 **Bridle J.S.:** *Probabilistic interpretation of feedforward classification networks outputs, with relationship to statistical pattern recognition* - In "Neuro-computing: algorithms, architectures and applications", Springer Verlag (1989)

ii Un coût global approchant le risque moyen ?

Dans la section 3.4, trois approches pour la reconnaissance des formes ont été comparées. On a signalé à cette occasion que la minimisation d'une approximation du risque moyen est plus discriminante que l'estimation des probabilités a posteriori.

Or, les modèles de Markov discriminants présentés ici appartiennent résolument à cette dernière classe. Il serait intéressant s'inspirer de ces modèles pour définir une approximation du risque moyen, dans l'esprit de LVQ2.

La procédure d'apprentissage de Viterbi (cf §6.6.2.3) appartient peut-être à cette catégorie. A ma connaissance cependant, *aucune preuve* ne confirme cette supposition incertaine.

Cette procédure est utilisée dans [1]. Elle consiste à effectuer une première segmentation, linéaire par exemple, associant états et observations. On entraîne ensuite les probabilités conditionnelles de transition avec les exemples fournis par cette segmentation. On peut alors utiliser l'algorithme de Viterbi pour obtenir une nouvelle segmentation, et recommencer.

9.2.3.2 Reconnaissance et algorithme de Viterbi.

Tout ce qui précède suggère d'appliquer le signal à l'entrée de perceptrons multi-couches, et d'appliquer l'algorithme de Viterbi aux probabilités conditionnelles de transition pour déterminer la séquence d'états la plus probable dans un modèle composé de tous les mots à reconnaître (cf fig. 6.4). Cela est très comparable à ce que l'on faisait avec les vraisemblances conditionnelles, dans le cas des modèles de Markov classiques (cf. §6.6.2).

Le résultat est *catastrophique*. Dans [1] les auteurs trouvent beaucoup plus efficace, avant d'appliquer l'algorithme de Viterbi, de diviser les probabilités conditionnelles de transition par les probabilités a priori des états mesurées sur les exemples. Avec, par exemple, les hypothèses (9.34),

$$\frac{P(s_t|O_{t-p}\dots O_t,s_{t-q}\dots s_{t-1})}{P(s_t)} = \frac{p(O_{t-p}\dots O_t|s_{t-q}\dots s_{t-1},s_t)}{p(O_{t-p}\dots O_t)} \qquad (9.35)$$

Cela revient donc à retrouver des quantités proportionnelles aux vraisemblances conditionnelles, avant d'utiliser l'algorithme de Viterbi. Quoique peu satisfaisante, cette solution permet d'atteindre des résultats sensiblement supérieurs à ceux obtenus à l'aide de modèles de Markov non discriminants.

---

1 **Bourlard H., Morgan N.:** *A Continuous Speech Recognition System Embedding MLP into HMM* - Advances in Neural Information Processing Systems 2 - pp 186-187, Morgan-Kaufman. Denver (1990)

Comprendre l'echec de l'application simple de l'algorithme de Viterbi requiert une étude plus précise. En effet, ce phénomène est lié au traitement de la parole continue. Le chapitre suivant est donc consacré à l'étude d'un modèle de Markov discriminant, pour la reconnaissance de la parole continue, et plus généralement pour la reconnaissance de séquences.

10 Reconnaissance de séquences et modèles de Markov discriminants.

10.1 Segmenter, reconnaître.

10.1.1 Extraire des séquences de symboles.

Le problème abordé dans ce chapitre consiste à extraire d'un signal une séquence de symboles. Ce problème recouvre bien des aspects des dispositifs de reconnaissance de la parole. Il se présente chaque fois que l'on désire interpréter un signal temporel.

On fait en général l'hypothèse suivante: Le signal est composé d'une séquence de formes de longueurs variables, correspondant chacune à un symbole. Le problème consiste donc à *segmenter* notre signal en formes, et à les *reconnaître*.

Une telle supposition n'est pas très satisfaisante. Dans le cas de la parole, on ne peut guère caractériser la frontière entre deux phonèmes. Lorsqu'on parle de coarticulation, d'allophones, ou de contextes phonétiques, on désigne en fait de multiples exceptions à cette hypothèse.

10.1.2 Programmation dynamique.

Il est très rare que l'on sache séparer les problèmes de segmentation et de reconnaissance. On ne peut dire qu'un mot commence à tel instant et finit à tel autre instant que parce que l'on a identifié ce mot, et vice-versa…

Deux variantes d'un même algorithme, la programmation dynamique, sont très souvent utilisées pour réaliser simultanément segmentation et reconnaissance.

- Dans le cas de l'alignement temporel, on utilise la programmation dynamique pour déterminer la séquence de formes de référence *la moins distante* de notre signal.

- Dans le cas de l'algorithme de Viterbi, on utilise la programmation dynamique pour déterminer une séquence d'états *la plus vraisemblable*, en se fondant sur une estimation empirique de ces probabilités.

Bien des méthodes d'estimation des vraisemblances en un point, comme les fenêtres de Parzen, cf [1,2], consistent justement à calculer les distances qui séparent ce point des exemples les plus proches. Cela montre bien l'identité de principe entre ces deux méthodes.

10.2 Modèles de Markov discriminants pour reconnaître des séquences.

Les modèles de Markov discriminants peuvent être étudiés dans le contexte général de la reconnaissance de séquences. Leurs propriétés diffèrent de façon importante de celles des modèles de Markov cachés usuels (MLE).

En particulier, les méthodes usuelles de juxtaposition de modèles doivent être abandonnées. Ces modèles permettent en revanche de reconnaître une séquence de mots sans segmenter simultanément le signal.

10.2.1 Motivation et contraintes.

Dans la section 9.2, on avait présenté un modèle discriminant sans se préoccuper de son utilisation pour la reconnaissance de la parole continue, et en espérant que les recettes habituelles des modèles de

---

1 **Duda R.O., Hart P.E.:** *Pattern classification and Scene analysis* - Wiley , (1973).

2 **Bourrely J.:** *Thèse de doctorat* - Université de Paris XI, Orsay (à venir)

Markov fonctionneraient. Il n'en est malheureusement rien (cf §9.2.3.2), à moins de se ramener à des vraisemblances conditionnelles en divisant les probabilités conditionelles de transition par les probabilités des états.

Une étude plus précise était donc nécéssaire. Afin d'étudier des problèmes qui n'apparaîssent que dans ce cadre, elle doit embrasser l'ensemble d'un modèle de Markov discriminant pour la reconnaissance de parole continue. Le formalisme nécéssaire est introduit dans la section 10.2.2.

La section 10.2.3 présente une propriété de conservation des probabilités, dont la principale conséquence est l'impossibilité d'utiliser l'algorithme de Viterbi et les méthodes usuelles de juxtaposition et de concaténation de modèles de Markov.

Les effets de coarticulation en parole sont tels que le signal est transformé en une succession de transitoires. *La détermination de leurs frontières est alors illusoire*, mais peut affecter sévèrement la qualité de la reconnaissance. Cette étude donne l'opportunité, dans la section 10.2.4, d'étudier une solution de remplacement pour l'algorithme de Viterbi, et de s'affranchir de la nécessité de segmenter le signal pendant le processus de reconnaissance.

10.2.2 Un modèle Markovien discriminant.

i Définitions préalables.

On définit à nouveau Ω, l'*espace des observations*, c'est à dire l'espace mesurables dans lequel le signal prend ses valeurs. On notera $(O_1 \dots O_T) \in \Omega^T$ une *séquence d'observations*, c'est à dire un signal de durée T.

Un modèle de Markov discriminant W est défini par:

S L'ensemble de ses états. On notera s_t l'état à l'instant t.

$P_{ij}(O_t)$ $i,j \in S$, $O_t \in \Omega$

Le modèle paramétrique de la probabilité $P(s_t=j|O_1 \dots O_T, s_1 \dots s_{t-1}=i)$ de l'état j sachant le signal et les états antérieurs. On a une hypothèse du type (9.31): ces probabilités de transition ne dépendent que de l'état immédiatement antérieur i et l'observation courante O_t.

Dans la suite, comme au chapitre 9, on notera $P(\dots)$ les probabilités réelles et $P_\lambda(\dots)$ les modèles paramétriques de ces probabilités, exprimés à l'aide des $P_{ij}(O_t)$.

IJ Un ensemble d'indices (i,j) définissant les transitions autorisées: si $(i,j) \notin IJ$, $P_{ij}(O_t) = 0$

Dans cette approche, un modèle W représente l'ensemble des séquences d'observations. L'hypothèse suivante est destinée à nous donner un moyen d'extraire une séquence de symboles d'un signal.

On note Ψ l'ensemble des séquences de symboles ω pouvant être reconnues. On supposera que certaines transitions sont étiquetées par un symbole. Dans un chemin $s_1 \ldots s_n$, les transitions étiquetées définissent donc sans ambiguïté une séquence de symboles $\psi(s_1 \ldots s_n)=(\omega_1 \ldots \omega_M)$, éventuellement vide.

Soit Γ_T, l'ensemble des chemins $(s_1 \ldots s_T)$ du modèle W, de longueur T, de probabilité non nulle. Les ensemble

$$\Gamma_T(\omega_1 \ldots \omega_M) = \psi^{-1}\{(\omega_{m_1} \ldots \omega_{m_M})\}$$

constituent une partition de Γ_T. On notera IJ_ω l'ensemble des transitions étiquetées par le symbole ω. On notera IJ_0 les autres transitions. Ces ensembles sont disjoints, et forment une partition de IJ.

ii Calcul des probabilités a posteriori des séquences de symboles.

Nanti de ce formalisme savant, évaluons maintenant la probabilité d'une séquence de symboles, sachant le signal. On peut écrire

$$P_\lambda(\omega_1 \ldots \omega_M | O_1 \ldots O_T) = P_\lambda((s_1 \ldots s_T) \in \Gamma_T(\omega_1 \ldots \omega_M) \mid O_1 \ldots O_T)$$

$$P_\lambda(\omega_1 \ldots \omega_M | O_1 \ldots O_T) = \sum_{(s_1 \ldots s_T) \in \Gamma_T(\omega_1 \ldots \omega_M)} P_\lambda(s_1 \ldots s_T \mid O_1 \ldots O_T) \qquad (10.1)$$

Posons maintenant:

$$\alpha_t(m,j) = P_\lambda(s_t=j,\ (s_1 \ldots s_t) \in \Gamma_t(\omega_1 \ldots \omega_m) \mid O_1 \ldots O_T)$$

$$= \sum_{\substack{(s_1 \ldots s_t) \in \Gamma_t(\omega_1 \ldots \omega_m) \\ s_t=j}} P_\lambda(s_1 \ldots s_t \mid O_1 \ldots O_T) \qquad (10.2)$$

La quantité $\alpha_t(m,j)$ est simplement la probabilité que l'on soit dans l'état j à l'instant t, tout en étant à l'élément m de la séquence de symboles $(\omega_1 \ldots \omega_M)$, sachant le signal. On remarque bien sûr que:

$$P_\lambda(\omega_1 \ldots \omega_M \mid O_1 \ldots O_T) = \sum_{s \in S} \alpha_T(M,s) \qquad (10.3)$$

On compte évaluer les $\alpha_t(m,j)$ de façon récurrente au moyen des probabilités de transition $P_{ij}(O_t)$. On a en effet, comme en (9.30) et (9.32),

$$\alpha_t(m,j) = \sum_{\substack{(s_1 \ldots s_t) \in \Gamma_t(\omega_1 \ldots \omega_m) \\ s_t = j}} \prod_{p=1}^{t} P_{s_{p-1} s_p}(O_p))$$

que l'on décompose sur l'avant dernier état

$$\alpha_t(m,j) = \sum_{k \in S} P_{kj}(O_t) \sum_{\substack{(s_1 \ldots s_t) \in \Gamma_t(\omega_1 \ldots \omega_m) \\ s_t = j, s_{t-1} = k}} \prod_{p=1}^{t-1} P_{s_{p-1} s_p}(O_p))$$

$$\alpha_t(m,j) = \sum_{k \text{ tq. } (k,j) \in IJ_0} P_{kj}(O_t) \sum_{\substack{(s_1 \ldots s_{t-1}) \in \Gamma_{t-1}(\omega_1 \ldots \omega_m) \\ s_{t-1} = k}} \prod_{p=1}^{t-1} P_{s_{p-1} s_p}(O_p))$$

$$+ \sum_{k \text{ tq. } (k,j) \in IJ_{\omega_m}} P_{kj}(O_t) \sum_{\substack{(s_1 \ldots s_{t-1}) \in \Gamma_{t-1}(\omega_1 \ldots \omega_{m-1}) \\ s_{t-1} = k}} \prod_{p=1}^{t-1} P_{s_{p-1} s_p}(O_p))$$

On obtient alors la formule de récurrence sur les $\alpha_t(m,j)$:

$$\alpha_t(m,j) = \sum_{k \text{ tq. } (k,j) \in IJ_0} P_{kj}(O_t)\ \alpha_{t-1}(m,k) + \sum_{k \text{ tq. } (k,j) \in IJ_{\omega_m}} P_{kj}(O_t)\ \alpha_{t-1}(m-1,k))$$

Les probabilités initiales des états, $\alpha_1(0,j) = P_{0j}(O_1)$ sont en général connues: Si un signal $(O_1 \ldots O_T)$ est toujours entouré de deux silences, on considère alors que seule est non nulle la probabilité initiale d'un état unique correspondant au silence.

$$\alpha_1(0,j) = P_{0j}(O_1)$$

$$\alpha_t(m,j) = \sum_{k \text{ tq. } (k,j) \in IJ_0} P_{kj}(O_t)\ \alpha_{t-1}(m,k) + \sum_{k \text{ tq. } (k,j) \in IJ_{\omega_m}} P_{kj}(O_t)\ \alpha_{t-1}(m-1,k)) \qquad (10.4)$$

La formule (10.4) est l'équivalent, pour les modèles discriminants, de la formule (6.18) appliquée, à un modèle obtenu en concaténant les modèles correspondant aux symboles $\omega_1 \ldots \omega_M$.

Les formules (10.3) et (10.4) permettent alors de calculer, à partir des probabilités conditionnelles de transition $P_{ik}(O_t)$, une expression paramétrique de la probabilité a posteriori d'une séquence de symboles, $P_\lambda(\omega_1 \ldots \omega_M \mid O_1 \ldots O_T)$.

iii Contraintes sur les probabilités conditionnelles de transition.

Or, ces probabilités doivent respecter la condition suivante

$$\sum_{(\omega_1 \dots \omega_M) \in \Psi} P_\lambda(\omega_1 \dots \omega_M \mid O_1 \dots O_T) = 1 \qquad (10.5)$$

On montre, comme précédemment (cf. §9.2.2.1), qu'il suffit que les probabilités conditionnelles de transition vérifient la condition de normalisation suivante

$$\sum_{k \in S} P_{ik}(O_t) = 1 \qquad (10.6)$$

En effet, comme les $\Gamma_T(\omega_1 \dots \omega_M)$ constituent une partition de Γ_T, on peut écrire:

$$\sum_{(\omega_1 \dots \omega_M) \in \Psi} P_\lambda(\omega_1 \dots \omega_M \mid O_1 \dots O_T) = \sum_{(s_1 \dots s_T) \in \Gamma_T} P_\lambda(s_1 \dots s_T \mid O_1 \dots O_T) =$$

$$= \sum_{k_1 \in S} P_{0k_1}(O_1) \left(\sum_{k_2 \in S} P_{k_1 k_2}(O_2) \dots \left(\sum_{k_T \in S} P_{k_{T-1} k_T}(O_T) \right) \dots \right) = 1$$

Une méthode d'apprentissage fondée sur le principe du maximum de probabilité a posteriori (cf. fig 9.4) requiert que les $P_\lambda(\omega_1 \dots \omega_M \mid O_1 \dots O_T)$ soient normalisées. Il est alors crucial de respecter la contrainte (10.6) sur les probabilités conditionnelles de transition.

10.2.3 Topologies des modèles discriminants.

Les dispositifs Markoviens classiques de reconnaissance de mots connectés consistent en général en un grand modèle de Markov, construit en juxtaposant des petits modèles de mots, ou d'autres unités phonétiques (cf. §6.6.2).

Or, les modèles de Markov discriminants possèdent une propriété de conservation des probabilités, qui handicape ces méthodes simples de juxtaposition.

i Evolutions de la probabilité des états.

On notera $p_t(i) = P_\lambda(s_t = i \mid O_1 \dots O_T)$, la probabilités de l'état i à l'instant t, sachant le signal. Comme d'habitude, cette probabilité peut être calculée récursivement, à l'aide d'une formule analogue à (10.4).

$$p_1(j) = P_{0j}(O_1), \quad p_t(j) = \sum_{i \text{ tq. } (i,j) \in IJ} P_{ij}(O_t)\, p_{t-1}(i) \qquad (10.7)$$

On peut alors vérifier que la probabilité d'un état évolue *conservativement*: Elle augmente de la somme des probabilités entrantes, et diminue de la somme des probabilités sortantes.

$$p_t(j) - p_{t-1}(j) = \left(\sum_{i \neq j \text{ tq. } (i,j) \in IJ} P_{ij}(O_t)\ p_{t-1}(i) \right) - (1-P_{jj}(O_t))\ p_{t-1}(j)$$

$$= \left(\sum_{i \neq j \text{ tq. } (i,j) \in IJ} P_{ij}(O_t)\ p_{t-1}(i) \right) - \left(\sum_{k \neq j \text{ tq. } (j,k) \in IJ} P_{jk}(O_t)\ p_{t-1}(j) \right)$$

On peut raisonner avec une analogie hydraulique. Chaque état j est un réservoir de probabilité; des canalisations entrantes, de débit $P_{ij}(O_t)p_{t-1}(i)$ le remplissent, et des canalisations sortantes, de débit $P_{ik}(O_t)p_{t-1}(j)$ le vident.

Cela se généralise bien sûr à la probabilité d'un groupe d'états W. On définit

$IN = \{ (i,j) \in IJ \text{ tq. } i \in S-W \ \ j \in W \}$

l'ensemble des transitions d'un état extérieur à W vers un état de W,

$OUT = \{ (i,j) \in IJ \text{ tq. } i \in W \ \ j \in S-W \}$

l'ensemble des transitions d'un état de W vers un état extérieur à W.

On obtient une équation de conservation en additionnant les évolutions des probabilités de chaque état de W. En effet, les flux le long des transitions internes de W s'équilibrent.

$$\sum_{s \in W} p_t(s) - \sum_{s \in W} p_{t-1}(s) =$$

$$= \left(\sum_{(i,j) \in IN} P_{ij}(O_t)\ p_{t-1}(i) \right) - \left(\sum_{(i,j) \in OUT} P_{ij}(O_t)\ p_{t-1}(i) \right) \qquad (10.8)$$

Ce résultat s'exprime simplement avec notre analogie hydraulique: On considère un ensemble de réservoir de probabilités. Les variations de la probabilité totale ne dépendent que des canalisation reliées à l'extérieur, et non des échanges de probabilité entre réservoirs.

ii Influences sur la topologie des modèles de Markov discriminants.

Supposons, comme on le fait classiquement (cf. fig 6.4), que la séquence de symboles associée à un chemin soit définie par l'appartenance des états du chemin à tel ou tel modèle de mot, c'est à dire à tel ou tel ensemble d'état.

La probabilité d'un mot à un instant donné est alors la probabilité des états composant le modèle de ce mot. Elle est donc *entièrement définie* par les probabilités des transitions vers l'état initial, ou partant de l'état final de ce modèle de mot.

Considérons par exemple deux mots "pavement" et "parement". Il est vraisemblable que les probabilités des transitions initiales et finales des modèles de ces mots seront comparables. Les

probabilités (fig 10.1) que l'on soit à un instant donné dans l'un ou l'autre mot sont alors voisines, et ne dépendent pas de la syllabe centrale du mot réellement prononcé!

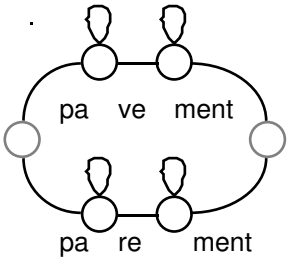


Fig 10.1 - Seules les transitions /pa/ et /ment/ déterminent la probabilité globale des états des mots "pavement" et "parement" à un instant donné!.

L'algorithme de Viterbi ne peut alors pas déterminer le mot réellement prononcé!

Une solution (fig. 10.2) consiste alors à mettre en commun les parties initiales ressemblantes des mots, ou des phrases, et à utiliser les transitions finales pour déterminer l'occurrence d'un mot. On organise ainsi un *arbre de classification*.

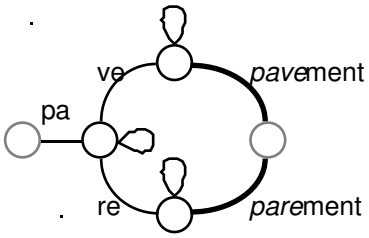


Fig 10.2 - Mise en commun des parties identiques. Aux transitions en gras sont associés les mots "pavement" et "parement".

Une autre approche consiste à autoriser beaucoup de transitions, et laisser l'algorithme d'apprentissage déterminer leur usage. On peut par exemple définir un modèle de Markov "en couches" (cf fig 10.3).

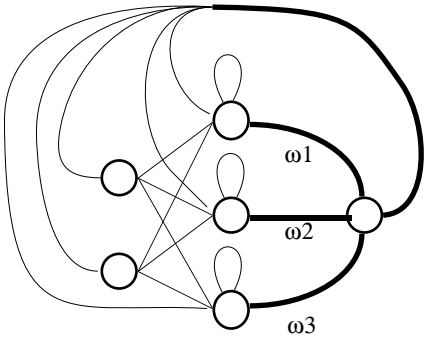


Fig 10.3 - Un modèle "en couches", composé de trois couches à respectivement 2, 3 et 1 état. Les transitions associées à des symboles sont indiquées.

10.2.4 Algorithme de Viterbi, et dérivés.

On dispose d'une séquence d'observations $(O_1 \dots O_T)$. On désire trouver la séquence de symboles $(\omega^*_1 \dots \omega^*_N)$ que nos estimations des probabilités à posteriori déclarent la plus probable. La probabilité de cette séquence s'exprime de la façon suivante:

$$\zeta = \max_{(\omega_1 \dots \omega_M) \in \Psi} \sum_{(s_1 \dots s_T) \in \Gamma_T(\omega_1 \dots \omega_M)} P_\lambda(s_1 \dots s_T \mid O_1 \dots O_T) \qquad (10.9)$$

On retrouve alors la séquence de symboles optimale en regardant quel terme est retenu par l'opérateur maximum. Cette formule ne permet malheureusement d'envisager le calcul direct de ζ . Il est intéressant de comparer cette formule avec les quantités calculées par les algorithmes de Viterbi (ζ^v) et Forward (ζ^f). Dans le cas discriminant, on a:

$$\zeta^v = \max_{(s_1 \dots s_t) \in \Gamma_t} P_\lambda(s_1 \dots s_T \mid O_1 \dots O_T)$$

$$\zeta^f = \sum_{(s_1 \dots s_T) \in \Gamma_T} P_\lambda(s_1 \dots s_T \mid O_1 \dots O_T) = 1$$

Et l'on montre aisément en posant:

$$\zeta^v_t(j) = \max_{(s_1 \dots s_t) \in \Gamma_t} P_\lambda(s_1 \dots s_t \mid O_1 \dots O_t)$$

$$\zeta^f_t(j) = \sum_{(s_1 \dots s_t) \in \Gamma_t} P_\lambda(s_1 \dots s_t \mid O_1 \dots O_t)$$

que ces deux valeurs ζ^v et ζ^f se calculent facilement à l'aide des récurrences:

$$\zeta^v_0(j) = P_{0j}(O_1), \quad \zeta^v_t(j) = \max_{(k,j) \in IJ(W)} \zeta^v_{t-1}(j) P_{kj}(O_t), \quad \zeta^v = \max_{s \in S(W)} \zeta^v_{t-1}(j)$$

$$\zeta^f_0(j) = P_{0j}(O_1), \quad \zeta^f_t(j) = \sum_{(k,j) \in IJ(W)} \zeta^f_{t-1}(j) P_{kj}(O_t), \quad \zeta^f = \sum_{s \in S(W)} \zeta^f_{t-1}(j) \qquad (10.10)$$

On ne peut cependant pas en faire autant avec l'expression (10.9). La démonstration habituelle échoue à cause de l'impossibilité de permuter les signes de sommation et de maximum. On ne dispose en effet que des inégalités:

$$\forall a_{ij} \geq 0, \quad \max_{i \in I} \max_{j \in J} a_{ij} \leq \max_{i \in I} \sum_{j \in J} a_{ij} \leq \sum_{j \in J} \max_{i \in I} a_{ij} \leq \sum_{j \in J} \sum_{i \in I} a_{ij}$$

Avec lesquelles on déduit un ordre sur les valeurs ζ^v, ζ^f, et ζ.

$$0 < \zeta^v \leq \zeta \leq \zeta^f = 1 \qquad (10.11)$$

Les inégalités larges de (10.11) deviennent des égalités lorsque le chemin le plus probable est le seul possible. Si on peut approcher ζ plus finement que l'algorithme de Viterbi ζ^v, on se libère de l'hypothèse initiale, selon laquelle une portion bien définie du signal est associée à chaque symbole.

i Calcul explicite de ζ.

Il ne semble donc pas possible de calculer ζ au moyen d'une récurrence sur les états, comme dans le cas des algorithmes Forward et Viterbi. On peut néanmoins écrire:

$$\zeta_t(\omega_1 \ldots \omega_m, j) = \sum_{(s_1 \ldots s_t) \in \Gamma_T(\omega_1 \ldots \omega_m)} P_\lambda(s_1 \ldots s_t \mid O_1 \ldots O_t)$$

$$\zeta = \max_{(\omega_1 \ldots \omega_M) \in \Psi} \sum_{s \in S} \zeta_t(\omega_1 \ldots \omega_m, s)$$

Il existe alors une formule de récurrence similaire à (10.4) sur les $\zeta_t(\omega_1 \ldots \omega_m, j)$:

$$\zeta_t(\omega_1 \ldots \omega_m, j) = \sum_{k \text{ tq. } (k,j) \in IJ_0} P_{kj}(O_t)\ \zeta_{t-1}(\omega_1 \ldots \omega_m, k) + \sum_{k \text{ tq. } (k,j) \in IJ_{\omega_m}} P_{kj}(O_t)\ \zeta_{t-1}(\omega_1 \ldots \omega_{m-1}, k) \qquad (10.12)$$

Ces formules permettent de calculer exactement ζ. Elles ne sont cependant pas praticables dans le cas général. La récurrence (10.12) implique en effet la construction de l'arbre de toutes les séquences de symboles. Cela exige à la fois une capacité de stockage et un temps de calcul prohibitifs.

i Approximation de ζ.

Essayons donc de limiter les besoins en mémoire, en ne stockant qu'un seul terme $\zeta_t(\omega_1 \ldots \omega_m, j)$ par état et par unité de temps. La formule (10.12) ne pourra donc pas être calculée exactement, car certains termes des sommes ne seront pas disponibles. On designera par $\zeta^1{}_t(\omega_1 \ldots \omega_m, j)$ la valeur approchée de $\zeta_t(\omega_1 \ldots \omega_m, j)$ ainsi calculée.

On notera $M_t(j)$ le singleton composé par l'unique séquence $(\omega_i) = \omega_1 \ldots \omega_m$ pour laquelle on *mémorise* la grandeur $\zeta^1{}_t(\omega_1 \ldots \omega_m, j)$. Certains termes de la formule de récurrence (10.12) ne sont pas disponibles; on les élimine alors de la somme:

$$\zeta^1{}_t(\omega_1 \ldots \omega_m, j) \approx \sum_{\substack{k \text{ tq. } (k,j) \in IJ_0 \\ \text{et } (\omega_1 \ldots \omega_m) \in M_{t-1}(k)}} P_{kj}(O_t)\ \zeta^1{}_{t-1}(\omega_1 \ldots \omega_m, k) \quad + \sum_{\substack{k \text{ tq. } (k,j) \in IJ_{\omega_m} \\ \text{et } (\omega_1 \ldots \omega_{m-1}) \in M_{t-1}(k)}} P_{kj}(O_t)\ \zeta^1{}_{t-1}(\omega_1 \ldots \omega_{m-1}, k) \qquad (10.13)$$

Afin de minimiser l'erreur effectuée par l'approximation ci dessus, il est judicieux de choisir

$$M_t(j) = \left\{ \text{ArgMax}_{(\omega_1 \ldots \omega_m)}\ \zeta^1{}_t(\omega_1 \ldots \omega_m, j) \right\} \qquad (10.14)$$

Cela assure en effet que les termes négligés dans la somme (10.13) seront les termes les plus faibles de la somme (10.12). Définissons alors:

$$\zeta^1 = \max_{(\omega_1 \ldots \omega_M) \in M_T(S)} \sum_{s \in S} \zeta^1{}_t(\omega_1 \ldots \omega_M, s)_{(\omega_1 \ldots \omega_M) \in M_T(s)}$$

Nous cherchons maintenant à placer la grandeur ζ^1 dans l'inégalité (10.11). Il est évident que $\zeta^1 \leq \zeta$. En effet, le calcul (10.13) de ζ^1 est effectué en omettant dans le calcul de ζ certains termes positifs non mémorisés de la somme (10.12).

Un examen des formules (10.13) et (10.10) permet également conclure $\zeta^v \leq \zeta^1$. En effet, (10.10) consiste à ne conserver de la somme (10.12) que le terme le plus fort. La somme (10.13) consiste à ne conserver de la somme (10.12) que les termes disponibles, dont le terme le plus fort, d'après le choix de $M_t(j)$ effectué par (10.14). On conclut donc:

$$0 < \zeta^v \leq \zeta^1 \leq \zeta \leq \zeta^f = 1$$

De la même façon, il est possible de définir $\zeta^2, \ldots \zeta^n$, etc... en mémorisant deux (resp. n) termes par état et par unité de temps. La formule (10.14) est alors encore correcte: Le seul changement est que l'ensemble $M_t(j)$ contient alors 2 (resp. n) éléments. La formule (10.13) doit être modifiée: il faut mémoriser les 2 (resp. n) termes les plus élevés.

On obtient alors une famille d'algorithmes de complexité croissante, dont le plus simple est l'algorithme de Viterbi. Les grandeurs calculées par ces algorithmes vérifient:

$$0 < \zeta^v \leq \zeta^1 \leq \zeta^2 \leq \zeta^3 \ldots\ldots \leq \zeta \leq \zeta^f = 1 \qquad (10.15)$$

Il a déjà été souligné (cf. §6.6.3.2) que l'approximation sur laquelle repose l'algorithme de Viterbi ne devient pénalisante que si l'on améliore la capacité discriminante des modèles de Markov usuels. Cependant, la longueur des zones de coarticulation étant limitée, on peut espérer que l'un des premier ζ^n suffit à donner une bonne approximation de ζ.

• Epilogue

Confronter des connaissances théoriques et les résultats pratiques semble le meilleur moyen de comprendre les divers phénomènes qui accompagnent l'apprentissage.

Malgré une opposition de langage et de démarche, algorithmes connexionnistes et méthodes statistiques reposent sur un même fondement. Les statistiques ont donc fourni des outils mathématiques élégants, mais parfois complexes pour aborder les phénomènes de l'apprentissage. L'absence de connaissance analytique du problème, que l'on souhaite justement apprendre, limite cependant la force d'une telle théorie.

Je me suis attaché, ici, à montrer le versant statistique de l'apprentissage connexionniste. Il permet d'aborder des problèmes complexes, comme la reconnaissance de la parole, avec un peu plus de compréhension.

Cela se fait peut-être au détriment de la simplicité initiale du connexionnisme, qui est en partie responsable de son succès: Il ne suffit plus d'assembler quelques neurones formels pour obtenir des résultats pratiques.

Une voie de recherche prometteuse consiste à étudier les système modulaires présentés au chapitre 8. En effet, ils peuvent peut-être fournir un cadre formel, mathématiquement plus sain, et qui répond mieux à la grande variété des besoins pratiques.

Leur étude sur des cas pratiques requiert cependant un volumineux travail de préparation, d'écriture de logiciels, et de tests en vraie grandeur. Seul ce travail nous permettra de cerner les nouveaux problèmes que poserons ces systèmes.

Enfin, je serai heureux que les idées développées ici contribuent à faire avancer le fantastique problème de la reconnaissance automatique de la parole. Je ne peux décrire le plaisir extrême que j'ai eu à aborder ce sujet versatile et fascinant, ni l'étrange conviction que quelque chose de fondamental nous échappe toujours.

• Références bibliographiques

1. **Ackley D.H., Hinton G.E., Sejnowski T.J.:** *A learning algorithm for Boltzmann Machines* - Cognitive Science, 9, pp 147-169 (1985)

2. **Alexander K.S.:** *Probability inequality for empirical processes and a law of the iterated logarithm* - Ann. Prob. vol 12, pp 1041-1067, (1984).

3. **Amari S.I.:** *Learning patterns and pattern sequences by self-organizing net of threshold elements* - IEEE Trans. computer, vol C-21, n°11, (november 1972)

4. **Amari S.I:** *A theory of adaptive pattern classifiers* - IEEE Trans. on Elec. Com., EC16, pp 279-307, (1967)

5. **Atal B.S., Hanauer S.:** *Speech analysis and synthesis by linear prediction of the speech wave*. Journal of the Acoustical Society of America. n° 50, pp 637-655, (1971)

6. **Bahl L.R., Brown P.F., de Souza P.V., Mercer R.L.:** *Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition* -Proc. IEEE ICASSP 86, pp 49-52, Tokyo, (1986)

7. **Bahl L.R., Jelinek F., Mercer R.L.:** *A maximum likelihood Approach to Continuous Speech Recognition* - IEEE Trans. PAMI, vol5, n°2, pp 179-190 (march 1983)

8. **Baldi P., Hornik K.:** *Neural networks and principal component analysis: learning from example without local minima.* - Neural Networks, Vol 2, pp 53-58, (1989)

9. **Baum E., Haussler D.:** *What size net gives valid generalization* - Neural Computation, vol 1, n° 1, pp 151-160, (1989)

10. **Baum L. E., Petrie T., Soules G., Weiss N.**: *A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov chains*, Annals of mathematic statistics, vol41, n°1, pp 164-171 (1970)

11. **Becker S., Le Cun Y.:** *Improving the Convergence of Back-Propagation Learning with Second Order Methods.*- In Procs. of the 1988 Connectionist Models Summer School, pp 29-37, San Mateo, Morgan Kaufmann, (1989)

12. **Bedworth M.D., Bottou L., Bridle J.S., Fallside F., Flynn L., Fogelman F., Ponting K.M., Prager R.W.:** *Comparison of neural and conventioal classifiers on a speech recognition problem.* In IEE 1st International Conference on Artificial neural networks, London, (1989)

13. **Bennani Y., Charouar N., Gallinari P., Mellouk A.:** *Comparing Neural net models on speech recognition tasks*, Procs of Neuro-Nîmes 90, to appear (1990)

14. **Benveniste A., Metivier M., Priouret P.:** *Algorithmes adaptatifs et approximations stochastiques* - Masson (1987)

15. **Bollivier M. de, Gallinari P., Thiria S.:** *Cooperation of neural nets for robust classification*, IJCNN 90, San Diego, vol1, pp 113-120, (1990)

16. **Bollivier M. de, Gallinari P., Thiria S.:** *Multi-Module Neural Networks for Classification* - Procs of INNC'90, vol 2, pp 777-780, (1990)

17. **Bollivier, M. de., Lemer A., Tanguy J.:** *Reconnaissance de bruits sous-marins par réseaux multi-couches* - Neuro-Nîmes 88, EC2, pp 423-430, (1988)

18. **Bottou L., Fogelman Soulié F., Blanchet P., Liénard J.S.:** *Experiments with Time Delay Networks and Dynamic Time Warping for speaker independent isolated digits recognition*- Procs of EuroSpeech 89, vol II, pp 537-540 (1989)

19. **Bottou L., Fogelman Soulié F., Blanchet P., Liénard J.S.:** *Speaker independent isolated digit recognition: Multilayer perceptron vs Dynamic Time Warping*, Neural Networks, vol3, pp 453-465, (1990)

20. **Bottou L., Le Cun Y.:** *SN: Un simulateur pour réseaux connexionnistes* - Neuro Nîmes 88, pp 371-382, EC2 ed. (1988)

21. **Bottou L.:** *Reconnaissance de la parole par réseaux multi-couches* - Neuro-Nîmes 88 - pp 197-218, EC2 (1988)

22. **Bottou L.:** *Reconnaissance de la parole par réseaux multi-couches* - Rapport de stage, DEA d'Informatique du LRI, Univ. Paris XI, Orsay (1988)

23. **Bourlard H., Kamp Y., Wellekens C.J.:** *Speaker Dependent Connected Speech Recognition via Phonemic Markov Models* - Proc IEEE ICASSP 85, 31.5.1-4, (1985)

24. **Bourlard H., Kamp Y.:** *Auto-Association by Multilayer Perceptrons and Singular Value Decomposition* - Biological cybernetics, n°59, pp 291-294, (1988) (preprint september 1987)

25. **Bourlard H., Morgan N.:** *A Continuous Speech Recognition System Embedding MLP into HMM* - Advances in Neural Information Processing Systems 2 - pp 186-187, Morgan-Kaufman. Denver (1990)

26. **Bourlard H., Wellekens C.J.:** *Links between Markov Models and Multilayer Perceptrons*, Advances in Neural Information Processing Systems 1 - pp 502-510, Morgan-Kaufman. Denver (1989)

27. **Bourlard H., Wellekens C.J.:** *Speech Pattern Discrimination and Multilayer Perceptrons* - Computer, Speech and Language - vol 3, pp 1-19, (1989) & also Philips Research Lab. Brussels, Manuscript M211 (1987)

28. **Bourrely J., Bottou L.:** *Neur3: un réseau de neurones pour la reconnaissance de caractères* - Manuscrit non publié (décembre 1986).

29. **Bourrely J.:** *Parallelization of a Neural learning algorithm on a Hypercube* - In "Hypercube and distributed computers", Elsiever Science Publishing, North Holland (1989)

30. **Bourrely J.:** *Thèse de doctorat* - Université de Paris XI, Orsay (à venir)

31. **Bridle J.S.:** *Alpha-nets: A recurrent "neural" network architecture with a Hidden Markov Model interpretation.* To appear in Speech Communication special NeuroSpeech89 issue. (1990)

32. **Bridle J.S.:** *Probabilistic interpretation of feedforward classification networks outputs, with relationship to statistical pattern recognition* - In "Neuro-computing: algorithms, architectures and applications", Springer Verlag (1989)

33. **Bryson A.E.Jr., Yu Chi Ho:** *Applied Optimal Control*, Blaisdel Publishing Company, (1969)

34. **Cantelli F.P.:** *Sulla determinazione empirica della legi di probabilita*, Giornale dell'Instituto Italiano degli Attuari, n°4 (1933)

35. **Changeux J.P.:** *L'homme neuronal*, Fayard, (1983)

36. **Cover T.M.:** *Geometrical and statistical properties of linear inequalities with application to pattern recognition* - IEEE Trans. Electronic Computer, Vol EC-14, n°3, pp 326-334, (1965)

37. **Cybenko G.:** *Approximation by Superpositions of a Sigmoidal Function.* Math. Control Systems Signals, vol 2, pp 303-314 (1989)

38. **Decker M., Gauvain J.L., Mariani J.:** *Reconnaissance de mots isolés par diphonèmes enchaînés* - 14èmes J.E.P. Paris, (1985)

39. **Dempster A.P., Laird N.N., Rubin D.B.:** *Maximum Likelihood from Incomplete Data via the EM Algorithm* - Journal of the Royal Statistical Society, Series B, vol 39 n°1, pp 1-38, (1977)

40. **Derouault A.M.,** *Context Dependent Phonetic Markov Models for Large Vocabulary Speech Recognition* - Proc. IEEE ICASSP 1987, pp 360-363 (1987)

41. **Devijver P.A:** *Baum's forward-backward algorithm revisited* - Pattern Recognition Letters 3 pp 369-373 - North Holland (1985).

42. **Devillers L.:** *Reconnaissance monolocuteur des phonèmes français au moyen de réseaux à masques temporels* - Procs. des XVIIIèmes Journées d'Etudes sur la Parole - Montreal (1990)

43. **Devroye L.:** *Automatic Pattern Recognition: A Study of the Probability of Error* - IEEE Trans. on Pattern Anal. and Mach. Intell., vol 10, n° 4, pp 530-543, (1988)

44. **Devroye L.:** *Bounds for the uniform deviation of empirical measures* - J. Multivariate Anal., vol 12, pp 72-79 (1982)

45. **Driancourt X.D., Bottou L.:** *TDNN-Extracted Features*, Procs. of Neuro-Nîmes 90, EC2, (1990)

46. **Duda R.O., Hart P.E.:** *Pattern classification and Scene analysis*, Wiley (1973)

47. **Durbin R., Rumelhart D.E.:** *Product Units: A Computationally Powerful and Biologically Plausible Extension to Backpropagation Networks* - Neural Computation 1, pp 133-142, MIT Press (1989)

48. **Fisk D.L.:** *Quasi-martingales.* Trans. Amer. Math. Soc, n°120, pp 359-388 (1965)

49. **Gallinari P., Thiria S., Fogelman Soulié F.:** *Multilayer Perceptrons and Data Analysis* - Procs of IEEE 2nd ICNN, San Diego, vol I, pp 391-401 (1988)

50. **Gauvain J.L., Mariani J., Liénard J.S.:** *On the use of time compression for word-based speech recognition* - IEEE - ICASSP - Boston, (1983)

51. **Gauvain J.L.:** *A syllable based isolated word recognition experiment* - Procs. IEEE Conf. on ASSP 1986 (1986)

52. **Gladyshev E.G.:** *On stochastic approximation*. Theory of Probability and its Applications, vol 10, pp 275-278, (1965)

53. **Gori M., Bengio Y., De Mori R.:** *BPS, A learning algorithm for capturing the dynamic nature of speech*, Proc of Intl. Joint Conf. on Neural Networks, Washington DC, Vol II, pp 417-423, (1989)

54. **Greville T.N.E.:** *Some applications of the pseudo inverse of a matrix*- SIAM Rev 2, pp 15-22 (1960)

55. **Haffner P., Waibel A., Shikano K.:** *Fast Back-Propagation Methods for Neural Networks in Speech*. Proc. from the Fall meeting of the Acoustical Society of Japan. (1988)

56. **Haffner P., Waibel A., Shikano K.:** *Fast Back-Propagation Methods for Neural Networks in Speech*. Procs of European Conf. on Speech Communication and Technology, Paris, Vol 2, pp 553-556 (1989)

57. **Haussler D.:** *Generalizing the PAC Model for Neural Net and other Learning Application* - Technical Report UCSC-CRL-89-30, University of Calif., Computer Research Laboratory, Santa Cruz, CA (1989)

58. **Hebb D.:** *Organization of behavior*, Science Edition (1961)

59. **Hoeffding W.:** *Probability inequalities for sums of bounded random variables* - J. Amer. Statist. Ass., vol 58, pp 13-30 (1963)

60. **Hopfield J.J.:** *Neural networks and physical systems with emergent collective computational abilities* - P.N.A.S. USA, Vol 79, pp 2554-2558, (1982)

61. **Itakura F., Saito F.:** *Speech Analysis-Synthesis System Based on the Partial Autocorrelation Coefficients*. Acoust Soc of Japan Meeting (1969)

62. **Elman, D. Zipser:** *Learning the hidden structure of speech*. UCSD Institute for Cognitive Science Tech. Report 8701, San Diego, (1987).

63. **Jacobs, David A.H.:** *The State of the Art in Numerical Analysis*, Chapter III.1, §3.6, Academic Press, (edition 1977)

64. **Jelinek F.:** *Continuous Speech Recognition by Statistical Methods* - Proc. IEEE vol64, n°4, pp 532-556 (april 1976)

65. **Jouvet D., Monné J., Dubois D.:** *A new network-based speaker independent connected word recognition system.* - IEEE ICASSP 86, pp 1109-1112, Tokyo (1986)

66. **Jouvet D.:** *Reconnaissance de mots connectés indépendamment du locuteur par des méthodes statistiques* - Thèse de Doctorat ENST-88E006 (juin 1988)

67. **Kai Fu Lee, Hsiao Wuen Hou:** *Speaker Independent Phone Recognition using Hidden Markov Models* - IEEE Trans. on Acoustics, Speech and Signal Processing, vol37, n°11, pp 1641-1648.

68. **Kai Fu Lee:** *Large Vocabulary Speaker Independent Continuous Speech Recognition: The SPHINX system* - CMU PhD Thesis - CMU-CS -88-148 (1988)

69. **Kehagias A.:** *Optimal Control for Training: the Missing Link between Hidden Markov Models and Connectionist Networks* - Brown University, Div. of Applied Mathematics Tech. Report. (1989)

70. **Kirkpatrick S., Gelatt C.D.Jr, Vecchi M.P.:** *Optimisation by Simulated Annealing* - Science, vol 220, N° 4598, pp 671-680, (1983)

71. **Kohonen T., Barna G., Chrisley R.:** *Statistical Pattern Recognition with neural networks: Benchmarking Studies* - Second International Conference on Neural Networks, San Diego, IEEE proc. of ICNN, vol I, pp 61-68, (1988)

72. **Kohonen T., Ruohonen M.:** *Representation of associated data by matrix operators* - IEEE Trans Computers (july 1973).

73. **Kohonen T.:** *An adaptive associative memory principle* - IEEE Trans Computers (April 1974)

74. **Kohonen T.:** *Content addressable memories* - Springer series in information sciences, vol 2, Springer Verlag (1981)

75. **Kohonen T.:** *Self organisation and associative memories* - Springer series in information sciences, vol 8, Springer Verlag (1984)

76. **Kohonen T.:** *The "Neural" phonetic typewriter*. IEEE Computer, March 1988, pp 11-22, (1988)

77. **Krasovskii A.A:** *Dynamics of Continuous Self-Organizing Systems* - Fizmatgiz Moscow (en Russe) (1963)

78. **Kuhn G., Watrous R.L., Ladendorf B.:** *Connected recognition with a recurrent network*- Procs of NeuroSpeech 89, Edinburgh, Scotland (1989)

79. **Lamel L.F., Kassel R.H., Seneff S.:** *Speech Database development: Design and analysis of the acoustic-phonetic corpus* - Proc. DARPA Speech Recogn. Workshop, L.S.Baumann Ed, pp 100-109 (1986)

80. **Lang K., Hinton G.:** *the development of the Time Delay Neural Network Architecture for Speech Recognition*, Carnegie Mellon University TR CMU-CS-88-152, (1988)

81. **Le Cam L.:** *Asymptotic Methods in Statistical Decision Theory* - Springer Series in Statistics, Springer Verlag (1986)

82. **Le Cun Y., Boser B., & al.,** (Bell Laboratories)**:** *Handwritten Digit Recognition with a Back-Propagation Network*- Advances in Neural Information Processing Systems II, Morgan Kaufmann, (1990)

83. **Le Cun Y., Denker J.S., Solla S.:** *Optimal Brain Damage* - Advances in Neural Information Processing Systems II, Morgan Kaufmann Denver (1990)

84. **Le Cun Y.:** *A theoretical framework for back-propagation* - Proceedings of the 1988 Connectionist Models Summer School, pp 21-28, CMU, Pittsburgh, PA, Morgan Kaufmann (1988)

85. **Le Cun Y.:** *Generalization and Network Design Strategies - (Expanded version)* - Technical report CRG-TR-89-4, University of Toronto (1989)

86. **Le Cun Y.:** *Generalization and Network Design Strategies* - Connectionism in Perspective, Zurich, Switzerland, Elsiever (1989)

87. **Le Cun Y.:** *Learning processes in an asymmetric threshold network* - In "Disordered systems and biological organization", NATO ASI series in systems and computer science, n° F20, pp 233-240, Springer Verlag (1986)

88. **Le Cun Y.:** *Manuel du simulateur HLM* - dans "Modèles Connexionnistes de l'apprentissage": Thèse de Doctorat de l'Université Paris 6, Paris (1987)

89. **Le Cun Y.:** *Modèles Connexionnistes de l'Apprentissage*- Thèse de doctorat de l'Université de Paris 6, (1987)

90. **Levinson S.E., Rabiner L.R. Sondhi M.M.:** *An introduction to the application of the theory of probabilistic functions of a Markov process to Automatic Speech Recognition* - Bell Systems Technical Journal Vol62, n°4, pp 1035-1074 (april 1983)

91. **Liénard J.S.**: *Les processus de la communication parlée* - Masson (1977).

92. **Liporace Louis A.** *PTAH on continuous multivariate functions of Markov chains* , IDA-CRD report N° 80193 (1976)

93. **Liporace Louis.A.**. *Maximum Likelihood Estimation for Multivariate Observations of Markov Sources*, IEEE Trans. on Information Theory, vol IT-28, n°5 (1982)

94. **Lippmann R.P, Gold B.:** *Neural Classifiers useful for Speech Recognition* - Procs of 1st International Conf. on Neural Networks, IEEE IV-417 (1987)

95. **Lippmann R.P.:** *Review of Neural Networks for Speech Recognition* - Neural Computation, vol 1, pp 1-38, (1989)

96. **Ljung L., Söderström T.,** *Theory and Practice of Recursive Identification* - MIT Press (1983)

97. **Mariani J., Prouts B., Gauvain J.L., Gangolf J.T.:** *Man Machine Speech Communication Systems, including word-based recognition and text to speech synthesis* - IFIP World Computer Congress, Paris, (1983).

98. **McCulloch W., Pitts W. L.R.:** *A logical calculus for the ideas immanent in nervous activity* - Bull. Math. Biophysics, 5, pp 115-133, (1943)

99. **McDermott E., Katagiri S.:** *Shift-invariant, Multi-category Phoneme Recognition using Kohonen's LVQ2* - Procedings of ICASSP89, S3.1, (1989)

100. **Mejia C., Bottou L., Fogelman Soulié F.:** *Galatea: a C library for connectionist applications* - Procs. of INNC'90, Paris, 9-13, (1990)

101. **Metivier M.:** *Martingales et convergence p.s. d'algorithmes stochastiques,* Dans "Outils et modèles mathématiques pour l'automatique, l'analyse de système et le traitement du signal", Editions du CNRS, vol 1, partie 3, "Techniques probabilistes et automatique et traitement du signal", pp 529-552 (1981)

102. **Minsky M., Papert S.:** *Perceptrons* - MIT Press (1968)

103. **Morgan N., Bourlard H.:** *Generalization and Parameter Estimation in Feedforward Nets: Some Experiments* - Advances in Neural Information Processing Systems II, Morgan Kaufmann (1990)

104. **Neveu J.:** *Martingales à temps discret,* Masson, (1972)

105. **O'Shaugnessy D.:** *Speech Communication, human and machine* -Addison Wesley. (1987).

106. **Parker D.B.:** *Learning logic* - Technical report TR-47, Sloan School of Management, MIT, Cambridge, MA

107. **Pearlmutter B.:** *Learning State Space trajectories in Recurrent Neural Networks* - CMU Tech. Report CMU-CS-88-191, (1988)

108. **Peretto P.:** *Collective properties of neural networks: a statistical physics approach.* - Biological cybernetics. n°50, pp 51-62 (1984)

109. **Personnaz L., Guyon I., Dreyfus G.:** *Collective computational properties of neural networks: new learning mechanism* - Phys. Rev. A, vol 34, pp 4217-4228, (1986)

110. **Picone J., Johnson M.A., Hartwell W.T.:** *Enhancing the Performance of Speech Recognition with Echo Cancellation* - Procedings IEEE ICASSP 1988, S-Vol.1, pp 529-532, (1988)

111. **Pineda F.J.:** *Generalization of Back-Propagation to Recurrent Neural Networks,* Physical Review Letters, Vol 59,n°19, pp 2229-2232, (1987)

112. **Prager R.W., Harrison T.D., Fallside F.:***Boltzmann machines for speech recognition* - Computer, Speech & Language - Vol 1, n°1, pp 3-27 (1986)

113. **Press W.H., Flannery B.P., & al.:** *Numerical Recipes* - Cambridge University Press, Cambridge (1988)

114. **Rabiner L.R., Wilpon J.G., Quinn A.M., Terrace S.G.:** *On the applications of embedded digit training to speaker independent connected digit recognition* - IEEE Trans. Acoustics, Speech and Signal Processing, vol32, n°2, pp 272-279 (april 1984)

115. **Rosenblatt F.:** *The Perceptron: a perceiving and recognizing automaton* - Project PARA, Cornell Aeronautical Lab. Report 85-460-1. (january 1957)

116. **Rumelhart D., Hinton G.E., Williams R.:** *Learning internal representations by error propagation* - in Parallel Distributed Processing, vol1: exploring the microstructure of cognition, D.Rumelhart, J.McClelland eds., MIT Press, (1986)

117. **Rumelhart D., Mc Clelland J. (eds):** *Parallel Distributed Processing, vol1&2*, D.Rumelhart, J.McClelland eds., MIT Press, (1986)

118. **Sejnowski T.J., Rosenberg, C.R.:** *NETtalk, a parallel network that learns to read aloud* - Tech Report 86-01, Dept of EE-CS, John Hopkins University, Baltimore MD - (1986)

119. **Singer H.:** *Utilisation de dissylabes pour la reconnaissance de la parole* - Rapport LIMSI, 88-4, (1988)

120. **Soules G.W.:** *A gradient derivation of PTAH* - IDA-CRD Tech Report N°80328, (september 1977)

121. **Tsypkin Ya.:** *Adaptation and Learning in Automatic systems* - Mathematics in science and engineering, vol 73, Academic Press, (1971)

122. **Tsypkin Ya.:** *Foundations of the Theory of Learning Systems* - Mathematics in science and engineering, vol 101, Academic Press, (1973)

123. **Vapnik V.N., Chervonenkis A. Ya:** *The Theory of Pattern Recognition* - Nauka, Moscou (1974)

124. **Vapnik V.N., Chervonenkis A. Ya.:** *On the uniform convergence of relative frequencies of events to their probabilities* - Theory of Probability and its Applications, vol 16, n°2 pp 264-280 (1971)

125. **Vapnik V.N.:** *Estimation of Dependences Based on Empirical Data* - Springer Series in Statistics, Springer Verlag (1982)

126. **Varfis A.:** *Univariate Economic Time Series Forecasting by Connectionnist Methods*, Proc. of INNC'90, vol 1, pp 342-345, Paris (1990)

127. **Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K.:** *Phoneme recognition: neural networks vs. Hidden Markov Models*. Proceedings ICASSP 88, S-Vol.1, 107-110, (1988).

128. **Watrous R.L.:** *Learned Phonetic Discrimination Using Connectionists Networks*, in European Conference on Speech Technology, pp 377-380, Edinburgh (sept 1987)

129. **Watrous R.L.:** *Learning Algorithms for Connectionist Networks: Applied Gradient Methods for Nonlinear Optimisation*, Procs of 1st Int. Conf. on Neural Networks, San Diego, CA, vol IV, pp 619-627, (1987)

130. **Werbos P.J.:** *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD Thesis, Harvard University, Cambridge, MA.

131. **Widrow B., Hoff M.E.:** *Adaptive switching circuits* - IRE WESCON Conv. record, part 4 1960, pp 96-104 (1960)

132. **Widrow B., Stearn S.D.:** *Adaptive signal processing*- Prentice Hall (1985)

133. **Wilks S. S.**: *Mathematical Statistics*, New-York: Wiley (1962)