Orsay
N° d'ordre: 3206

UNIVERSITÉ DE PARIS SUD
CENTRE D'ORSAY

THÈSE

Présentée pour obtenir
Le titre de Docteur es Sciences, Spécialité Informatique

Par

Xavier DRIANCOURT

Sujet:

Optimisation par descente de gradient stochastique de systèmes modulaires combinant réseaux de neurones et programmation dynamique.

Application à la reconnaissance de la parole.

Soutenue le 20 juin 1994, devant la commission d'examen, composée de:

M	Léon	BOTTOU	Examinateur	,
M	Patrick	GALLINARI	Directeur de thèse	,
M	Gérard	LIGOZAT	Président	,
Mme	Sylvie	MARCOS	Rapporteur	,
M	Laurent	MICLET	Rapporteur	,
M	Jean-Charles	POMEROL	Examinateur	.

Remerciements

Conformément à la tradition, mes remerciements vont tout d'abord aux membres du jury. Sylvie Marcos et Laurent Miclet ont accepté d'en être les rapporteurs, fonction toujours lourde de responsabilité et de labeur. Patrick Gallinari fut mon directeur de thèse et, en compagnie de Léon Bottou, me guida dans mon travail. Enfin Gérard Ligozat et Jean-Charles Pomerol ont accepté d'être examinateurs.

Je tiens à remercier ceux qui m'ont accompagné, précédé, ou succédé au sein de notre équipe qui sut toujours être un modèle d'intelligence distribuée: Thierry Artières, Léon Bottou, Younes Bennani, Tautvidas Cibas, Michel de Bollivier, Jean Bourrely, André Kouam, Carlos Mejia, Abdelhamid Mellouk, Isabelle Oliva, Nicolas Puech, Adélaïde Stévenin et tout ceux dont je ne peux citer le nom. Leur bonne humeur au sein d'un univers hostile et leurs conseils infiniment précieux m'ont permis de mener à bien cette thèse.

J'ai également une pensée reconnaissante pour les membres de la communauté scientifique qui ont fait progresser le domaine de l'apprentissage automatique; en particulier ceux avec qui j'ai été en relation: Yoshua Benjio, Laurence Devillers, Patrick Haffner, Eric Mac Dermott, Patrice Simard, Christian Wellekens et tous les autres, que je ne saurais citer ici.

Bien sur, je suis infiniment redevable à ceux qui ont formé mon esprit. Mes proches m'ont appris à vivre dans l'univers mouvant qui est le nôtre et m'ont donné foi en la vie. Mes professeurs ont su m'inculquer quelques notions en dépit de mon caractère plus prompt à la critique qu'à la louange. Je dois également beaucoup à ceux, toujours nombreux, qui m'ont montré ce qu'il ne fallait pas faire.

Sommaire

Remerciements....2

Sommaire....1

Notations....6

1 Introduction....9

1.1 Traiter l'information a posteriori....9

1.1.1 Quoi....9

1.1.2 Pourquoi....10

1.1.3 Comment....12

1.2 Réseaux de neurones formels....13

1.2.1 Contexte historique....13

1.2.2 Contexte émotionnel....16

1.2.3 Contexte scientifique....17

1.3 Reconnaissance de la parole....18

1.3.1 Un domaine complexe....18

1.3.2 Systèmes contemporains de décodage acoustique....18

1.3.3 Utilité des méthodes neuronales pour le décodage acoustique....19

1.4 Démarche fondamentale et appliquée....19

1.4.1 Nécessité d'un travail théorique....20

1.4.2 Nécessité d'un travail appliqué....20

1.5 Chronologie....21

1.6 Plan du mémoire....24

2 Apprentissage par descente de gradient stochastique 26
2.1 Apprentissage par optimisation 27
2.1.1 Critère d'optimisation ou fonction de coût 27
2.1.2 Optimisation 28
2.1.3 Cas différentiable 28
2.2 Optimisation par descente de gradient 29
2.2.1 Algorithme déterministe 29
2.2.2 Algorithme stochastique 31
2.2.3 Opérateurs compatibles, opérateurs de lissage 31
2.2.4 Une fonction de coût alternative 32
2.2.5 Descente de gradient sous contrainte 34
2.3 Apprentissage et adaptation 37
2.3.1 Deux problématiques proches 37
2.3.2 Algorithmes optimaux et quasi-optimaux 38
2.4 Convergence de l'algorithme de descente de gradient stochastique 43
2.4.1 Généralités 43
2.4.2 Quasi-Martingales 44
2.4.3 Un théorème de convergence de l'algorithme de descente de gradient stochastique pour un pas d'apprentissage anisotrope 46
2.5 Conclusion 58

3 Classification 59
3.1 Quelques éléments de la théorie de décision statistique 60
3.1.1 Notations 60
3.1.2 Risque moyen 61
3.1.3 Règle de décision fondamentale 62
3.1.4 Fonctions discriminantes 63
3.1.5 Condition de minimalité du risque moyen 64
3.1.6 Décision bayésienne classique 66
3.1.7 Limites de la théorie de décision statistique 66
3.1.8 Approches palliatives 67
3.2 Optimisation directe d'une approximation du risque bayésien 68
3.3 Systèmes probabilistes 75
3.3.1 Estimation des vraisemblances conditionnelles 75
3.3.2 Estimation directe des probabilités a posteriori 82
3.3.3 Conclusion sur les méthodes probabilistes 93
3.4 Conclusion sur les critères d'optimisation pour systèmes de classification 93

4 Quelques systèmes utilisant l'information a posteriori 95
4.1 Introduction 95
4.1.1 Supervision et discrimination 96
4.1.2 Les trois problématique des systèmes utilisant l'information a posteriori 97
4.1.3 Descente de gradient, rétropropagation du gradient, systèmes modulaires 99
4.2 Plus proches voisinages 102
4.2.1 Généralités 102
4.2.2 Critères de décision 103
4.2.3 Méthodes d'apprentissage et d'adaptation 109
4.3 Associateurs paramétriques 120
4.3.1 Généralités 120
4.3.2 Réseaux neuronaux à propagation 126
4.4 Optimisation de systèmes modulaires 132
4.4.1 Les systèmes modulaires 132
4.4.2 Exemples de systèmes modulaires enchaînés: 134
4.4.3 Exemples de systèmes modulaires par aiguillage 141
4.5 Conclusion 142

5 Quelques systèmes utilisant l'information temporelle 143
5.1 Utilisation la nature temporelle de l'information 144
5.1.1 La nature temporelle de l'information prise en tant que connaissance a priori 144
5.1.2 Notion de classe pour séquences de vecteurs 144
5.2 Plus proches voisinages 145
5.2.1 Programmation dynamique 145
5.2.2 Stratégies de vote: l'exemple du shift-LVQ 157
5.3 Modèles de Markov cachés 157
5.3.1 Chaînes de Markov, modèles de Markov cachés 157
5.3.2 Mesure de l'adéquation 159
5.3.3 Étiquetage 160
5.3.4 Optimisation du modèle 161
5.3.5 Conclusion 162
5.4 Associateurs paramétriques 163
5.4.1 Utilisation du partage de paramètres 163
5.4.2 Application au MLP: le TDNN 163
5.4.3 Réseaux récurrents 164
5.5 Systèmes modulaires 165

5.5.1 Un précurseur 166
5.5.2 Quelques références 166
5.5.3 Un système modulaire MLP-DP-1NN 167
5.5.4 Un système modulaire MLP-prédictif-DP-1NN 168
5.6 Conclusion 170

6 Un système robuste de reconnaissance de la parole en mots isolés 172
6.1 Introduction 172
6.1.2 Projet Pygmalion 173
6.1.3 Base de donnée de parole Pygmalion 173
6.1.4 Logiciels 174
6.2 Réseaux à délais temporels 175
6.2.1 Éléments techniques 175
6.2.2 TDNN ordinaires 177
6.2.3 FTDNN 179
6.2.4 Synthèse des résultats obtenus avec les TDNN 180
6.3 Extraction de codage par réseau à délais temporels 180
6.3.1 Principe 180
6.3.2 Mesures expérimentales 181
6.5 Systèmes modulaires MLP-DP optimisés coopérativement 183
6.7 Quelques comparaisons 184
6.7 Conclusion 186

7 Conclusion 188
7.1 Apprentissage numérique 188
7.2 Systèmes modulaires 189
7.3 Reconnaissance de la parole 189

A1 La parole 190
A1.1 Production naturelle de la parole 190
A1.1.1 Système phonatoire 191
A1.1.2 Sources d'excitation 191
A1.1.3 Articulation 191
A1.1.4 phonèmes 192
A1.2 Reconnaissance naturelle de la parole 192
A1.2.1 Physiologie du système auditif 192
A1.2.2 Psycho-acoustique 195
A1.2.3 Perception de la parole 197
A1.3 Traitements bas niveau 198
A1.3.1 Obtention d'un signal digitalisé 198

A1.3.2 Analyse temporelle.....199
A1.3.3 Analyse prédictive linéaire.....200
A1.3.4 Analyse fréquentielle.....201
A1.3.4 Analyse en ondelettes.....202
A1.4 Conclusion sur le signal de parole.....202

A2 Un autre exemple de système modulaire.....203
A2.0 Abstract.....204
A2.1 Introduction.....204
A2.2 Algorithm.....205
A2.2.1 Topological maps as an adaptive gradient algorithm.....205
A2.2.2 A modular architecture.....206
A2.2.3 Training algorithms.....207
A2.2.4 Learning in the presence of constraints.....208
A2.3 Experiments.....210
A2.3.1 Data sets.....210
A2.3.2 Performances.....211
A2.3 Discussion.....213
A2.4 References.....213

Références bibliographiques.....215

Notations

Les notations concernant les sujets approfondis dans ce mémoire seront spécifiées au cours des chapitres.

Les notations qui suivent, en revanche, ont un caractère général.

Notations relationnelles

$\equiv$	identique égalité. Deux fonctions sont identiquement égales quand elles ont le même domaine de définition et qu'en chaque point de ce domaine, elles sont égales.
$\stackrel{\Delta}{=}$	égalité faisant office de définition de son membre droit.

Notations ensemblistes

$\mathbb{1}_{\mathbb{E}}$	toute fonction caractéristique de l'ensemble $\mathbb{E}$; une telle fonction est définie pour un sur-ensemble de $\mathbb{E}$; elle prend la valeur 1 sur $\mathbb{E}$ et 0 pour le reste du sur-ensemble. Le sur-ensemble est précisé implicitement par son contexte d'utilisation.
Vect (**x**)	l'opérateur qui associe à une famille de vecteurs **x** d'un espace vectoriel le sous-espace vectoriel qu'ils engendrent (par combinaisons linéaires).

Notations probabiliste

$E_{\mathbf{x}} \{ \mathbf{x} \mid \mathbf{y} \}$	Espérance de la variable aléatoire **x** sachant la réalisation de la valeur **y**.

Notations vectorielles

$\bullet$ Produit scalaire

$*$ Produit termes à termes de deux matrices (resp. vecteurs) de mêmes dimensions.

$\frac{\mathbf{P}}{\mathbf{Q}}$ Division termes à termes de deux matrices (resp. vecteurs) de mêmes dimensions.

$\angle$ Application terme à terme d'une fonction scalaire sur un vecteur ou une matrice de coefficients.

$\otimes$ Produit de Kronecker de deux matrices ou vecteurs. Dans le cas de deux matrices, ce produit s'écrit:

$$\mathbf{P} \otimes \mathbf{Q} = \begin{pmatrix} P_{11}\mathbf{Q} & \cdots & P_{1,p}\mathbf{Q} \\ P_{21}\mathbf{Q} & \cdots & P_{2n}\mathbf{Q} \\ \cdots & \cdots & \cdots \\ P_{p'1}\mathbf{Q} & \cdots & P_{p'p}\mathbf{Q} \end{pmatrix}$$

$mat^k(\mathbf{x})$ Opérateur qui, à un vecteur $\mathbf{x}$ de dimension $k \bullet n$, associe canoniquement une matrice de dimension $k * n$. Cet opérateur est distributif pour l'addition.

$$mat^k(\mathbf{p}) = \begin{pmatrix} p_1 & \cdots & p_{(n-1)k+1} \\ p_2 & \cdots & p_{(n-1)k+2} \\ \cdots & \cdots & \cdots \\ p_k & \cdots & P_{nk} \end{pmatrix}$$

$$mat^k(\mathbf{p}+\mathbf{q}) = mat^k(\mathbf{p}) + mat^k(\mathbf{q})$$

$vec(\mathbf{X})$ Opérateur qui, à une matrice $\mathbf{X}$ de dimension $k * n$, associe canoniquement un vecteur de dimension $k \bullet n$. Cet opérateur est distributif pour l'addition.

$$vec(\mathbf{P}) = (\mathbf{P}_{11},\ldots,\mathbf{P}_{k1},\mathbf{P}_{21},\ldots,\mathbf{P}_{2k},\ldots,\ldots,\mathbf{P}_{n1},\ldots,\mathbf{P}_{nk})^T$$

$$vec(\mathbf{P}+\mathbf{Q}) = vec(\mathbf{P}) + vec(\mathbf{Q})$$

$\mathbf{1}^k$ Vecteur unitaire k, c'est à dire vecteur dont chaque coefficient est égal à 0, sauf le coefficient d'indice k qui vaut 1. La dimension du vecteur est précisée implicitement par son contexte d'utilisation.

$\mathbf{1}_1^k$ Vecteur ou matrice de dimension verticale k et dont chaque coefficient est égal à 1.

$\mathbf{1}$ Vecteur ou matrice dont chaque coefficient est égal à 1. La dimension du vecteur ou de la matrice est précisée implicitement par son contexte d'utilisation.

Notations différentielles

∇ Opérateur gradient. $\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \dots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix}$; $\nabla_{\mathbf{x}} \mathbf{f}^T(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f_n(\mathbf{x})}{\partial x_1} \\ \dots \quad \dots \quad \dots \\ \frac{\partial f_1(\mathbf{x})}{\partial x_m}, \dots, \frac{\partial f_n(\mathbf{x})}{\partial x_m} \end{pmatrix}$

∇^T Opérateur gradient-transposé d'une fonction vecteur.

$$\nabla_{\mathbf{x}}^T \mathbf{f}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \dots \quad \dots \quad \dots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{pmatrix}$$

Notations infinitésimales

$\sigma_{x \to y}(f(x))$ terme quelconque *dominé* par $f(x)$ quand x est au voisinage de y. Par exemple, x^2 est $\sigma_{x \to 0}(x)$.

$\omicron_{x \to y}(f(x))$ terme quelconque *au plus de l'ordre* de $f(x)$ quand x est au voisinage de y. Par exemple, $x+x^2$ est $\omicron_{x \to 0}(x)$.

Notations fonctionnelles

C^n On dit que f est C^n si sa $n^{ième}$ différentielle existe et est continue.

C^∞ On dit que f est C^∞ si, quelque soit n, sa $n^{ième}$ différentielle existe.

1 Introduction

1.1 Traiter l'information *a posteriori*

1.1.1 Quoi

L'expérience procure des données *a posteriori* dont la synthèse permet de construire des modélisations et des processus destinés à traiter *a priori* de nouvelles données de même nature.

Cependant l'information *a priori* et l'information *a posteriori* sont étroitement mêlées. Nous ne percevons clairement que les événements à portée de nos moyens sensoriels. Or nos moyens sensoriels sont conçus *a priori* pour percevoir de tels événements. De plus les événements perçus sont à portée; cela n'est pas toujours innocent, soit de notre fait, soit du fait d'une main étrangère, soit des biais naturels dont nous ne sommes pas toujours conscients.

Afin de réduire la part d'*a priori*, on conçoit des expériences, des campagnes de mesures, en prenant de nombreuses précautions méthodologiques… prises *a priori* ! De telles précautions sont utiles et nécessaires, mais elles ne permettent pas de séparer complètement l'*a priori* de l'*a posteriori*.

Aussi, bien qu'opposés conceptuellement, l'*a priori* et l'*a posteriori* sont naturellement indissociables.

1.1.2 Pourquoi

1.1.2.1 Un vieux rêve de l'Homme

Comprendre et décider.

La simplicité de ces deux mots dissimule mal l'infinie complexité des concepts sous-jacents. Pourtant comprendre et décider sont deux principes élémentaires que nous vivons quotidiennement. Mais ils nous résistent: notre compréhension n'est qu'approximative et nos décisions sont incertaines.

L'Homme lui-même ne les domine pas. La Nature a pourtant déployé des trésors de sophistication. Pour permettre à ses créatures de comprendre une plus grande variété de phénomènes et décider plus sûrement, elle les a dotées de capacités d'adaptation. L'information que nous percevons ne fait pas que transiter en attente d'une réaction de notre part; elle nous modifie en profondeur: nous sommes doués de la faculté d'apprentissage. La Nature est économe. Si de simples comportements figés avaient suffit, nous n'aurions pas bénéficié de cette faculté.

La relation entre la Nature et ses créatures ressemble fort à la relation entre l'Homme et ses machines. La fantastique variété de l'univers et la multitude de perceptions qu'il faut prendre en compte pour comprendre et décider se contentent mal du seul usage de procédures pré programmées de façon simple. La Nature a doté ses créatures de la faculté d'apprentissage. L'Homme rêve aujourd'hui de doter ses machines de la même faculté.

1.1.2.2 Vers une nouvelle démarche scientifique

Pour explorer l'univers qui se trouve au delà des limites de l'expression analytique de nos connaissances *a priori* (modèles physiques), on a recours à l'utilisation de connaissances *a posteriori* (exemples réels).

Cette problématique est très ancienne, puisqu'elle fonde les sciences expérimentales. La physique, la biologie, la sociologie, reposent sur des modèles mathématiques *a priori* dont on ajuste les paramètres *a posteriori* afin que le modèle se comporte de façon aussi proche que possible des mesures expérimentales. Quand un modèle s'avère trop approximatif au regard de données nouvelles, on en change et la science accomplit une nouvelle mutation.

Sans doute de nombreux modèles physiques ont-ils été conçus essentiellement pour formaliser et étendre des connaissances ancestrales (e.g. mécanique des solides, théorie des jeux…), ou pour analyser des phénomènes connus mais encore inexpliqués (e.g. poulie, machine à

vapeur…).

Cependant toutes les connaissances ancestrales et tous les phénomènes connus ne sont pas encore formalisés ni expliqués. La lecture, l'équilibre, la reproduction des espèces, parmi tant d'autres domaines, échappent grandement à notre analyse.

De plus, aujourd'hui, l'Homme se trouve confronté à des données inconnues il y a quelques années. La physique des particules, la cosmologie des premiers âges, les mutations de l'ADN sont autant de domaines pour lesquels nous ne disposons que de récentes données, rares au regard de la complexité des problèmes.

Les modèles simples d'antan laissent progressivement place à des modèles d'une complexité croissante. La mécanique newtonienne se trouve remplacée par la mécanique relativiste d'un maniement plus ardu. La chimie de Mendeleïev est presque rendue caduque par l'absconse mécanique quantique. Les modèles *a priori* se succèdent, sans cesse plus exacts au regard des données *a posteriori* que les expériences procurent.

L'automatisation du traitement de données *a posteriori*, rendue possible par les développements de l'informatique et des mathématiques de l'apprentissage apparaît ainsi comme un outil essentiel pour les années futures.

1.1.2.3 De nouveaux outils de production et de gestion

De nombreuses tâches de gestion tendent à être systématiquement automatisées. De nombreuses informations sont accessibles via des supports informatiques et télématiques. Ces données, pour l'instant exploitées le plus souvent de façon mécanique et statique, sont susceptibles d'être utilisées par des outils d'aide à la décision qui exploiteraient l'information *a posteriori* pour fournir des prévisions ou des estimations hypothétiques ou mettre en lumière des interactions entre différents facteurs.

Le besoin de systèmes intégrés, permettant à la chaîne de l'information et de ses traitements d'être entièrement conçue sous un support informatique, se fait sentir. Un exemple de système encore partiellement manuel est l'exploitation informatique de documents papiers: une organisation reçoit de nombreux documents sur support papier. Aujourd'hui, sauf cas particuliers, on ne sait pas encore convertir automatiquement ces documents vers un support informatique sous une forme structurée. Cela permettrait d'alimenter des systèmes de gestion de documents, fournissant des services tels que la recherche par thèmes ou par mots clés, la réutilisation d'une partie des informations d'origine (schémas…), le secrétariat sans papier (dit "paperless").

Enfin, la robotisation est encore largement limitée par des problèmes d'ordre perceptif ou cognitif. Il apparaît de plus en plus que des avancées importantes viendront, non de la programmation structurée *a priori* face à ces problèmes que l'on sait mal formaliser ou résoudre, mais de l'exploitation de nombreuses informations *a posteriori* que l'on sait souvent collecter et exploiter.

1.1.3 Comment

1.1.3.1 Méthodologie d'exploitation

Quand on cherche à automatiser une tâche pour laquelle on manque d'informations *a priori* (i.e. principes de traitement), on est amené à faire appel à l'information *a posteriori* (i.e. exemples pratiques). De plus, l'usage de données *a posteriori* a d'autres vertus que de faire ce qui ne peut être fait sans lui. Il permet souvent de faire l'économie de développements coûteux, d'approcher des systèmes lents par des processus plus rapides ou encore d'adapter en permanence des systèmes confrontés à des conditions évolutives imprévisibles.

Il faut alors bâtir une stratégie adaptée au problème.

Une telle stratégie répond à trois problématiques: l'adéquation, la généralisation et l'apprentissage.

La problématique de l'adéquation consiste à étudier la nature de la tâche pour en déduire la nature des procédures susceptibles de la traiter correctement. La famille de ces procédures est d'autant plus vaste que l'on manque d'information *a priori*, ou de temps pour l'exploiter.

La problématique de la généralisation consiste à régler, en vue d'un apprentissage, l'étendue optimale d'une famille de procédures en fonction de la quantité et de la régularité des informations *a posteriori*.

La problématique de l'apprentissage consiste à tirer directement partie des informations *a posteriori*. L'apprentissage est réduit à sa plus simple expression quand le système se contente de mémoriser ces informations (ex: méthode du plus proche voisin). Le plus souvent, on cherche à synthétiser l'information *a posteriori* en recherchant une procédure optimale au sein d'une famille de procédures, à l'aide d'une mesure de qualité portant sur le comportement des procédures vis-à-vis des informations *a posteriori*.

Ces trois problématiques sont conflictuelles. La problématique de la généralisation conduit souvent à dégrader volontairement les performances de l'apprentissage. La problématique de l'adéquation tend à engendrer des systèmes pour lesquels on ne connaît pas de procédure

d'optimisation convenable. Pourtant, au sein d'un système, ces trois problématiques sont souvent peu dissociables.

1.1.3.2 Techniques numériques

Parmi les techniques exploitant directement les informations *a posteriori*, la plupart sont des techniques numériques (par opposition aux techniques symboliques). On peut classer ces techniques en plusieurs catégories.

Les statistiques visent généralement à optimiser des systèmes de traitement linéaires.

Les modèles probabilistes visent à optimiser des automates stochastiques générateurs de données synthétiques en vue d'obtenir un comportement conforme avec les informations *a posteriori*.

Les techniques de plus proche voisinage consistent, pour chaque cas à traiter, à sélectionner les données *a posteriori* les plus proches, puis à fournir une réponse en fonction de cette sélection.

Les techniques neuronales consistent à optimiser des systèmes possédant de nombreux paramètres interagissant de façon simple, organisés à la façon de réseaux d'automates élémentaires.

Enfin, il convient de signaler que cette liste n'est pas exhaustive; par exemple il existe des méthodes hybrides combinant des techniques neuronales et de plus proche voisinage.

1.2 Réseaux de neurones formels

La complexité de l'univers, alliée aux possibilités nouvelles offertes par la puissance croissante des calculateurs a appelé un développement des méthodes informatiques d'apprentissage automatique. Les réseaux de neurones formels en constituent une famille de plus en plus importante. Dans une optique industrielle, les analogies avec la neurobiologie laissent aujourd'hui place à l'étude mathématique des systèmes d'apprentissage.

1.2.1 Contexte historique

Deux approches concurrentes et parfois complémentaires permettent d'appréhender les processus de la compréhension humaine. La première est macroscopique; elle fait appel à la psychologie et la philosophie. La seconde est microscopique; elle fait appel à la neurobiologie.

Dès la fin du XIX^{ème} siècle, des psychologues prennent de la hauteur vis-à-vis de leur discipline et tentent d'analyser *ce qu'est* et *ce que peut être* la compréhension. Pour cela, ils s'inspirent de considérations philosophiques, d'expériences psychologiques et des premières bribes de neurobiologie. L'analyse de certains d'entre eux, tels James [1], est d'une intelligence pénétrante et les idées qu'ils promeuvent sont toujours d'actualité.

Le milieu du XX^{ème} siècle voit apparaître l'informatique et avive le rêve d'une intelligence artificielle. Le premier problème posé fut celui des architectures possibles pour les ordinateurs. McCulloch et Pitts [2] posent la première pierre du connexionnisme en proposant une modélisation très simple du neurone, dont l'assemblage en réseau permet d'engendrer des fonctions fort complexes. Cependant cette technique est généralement plus ardue que d'écrire un programme pour une machine séquentielle [3].

Le connexionnisme apparaît néanmoins comme une idée séduisante. Parallèlement au début de l'informatique, les travaux des neurobiologistes et des psychologues se poursuivent. Hebb propose une règle d'apprentissage pour le neurone, plausible tant sur le plan biologique que sur le plan des capacités d'apprentissage et de calcul qu'elle induit [4]. De même, Lashley met en évidence l'aspect délocalisé de l'information cérébrale [5].

Peu après, les premiers systèmes neuronaux adaptatifs voient le jour: le Perceptron de Rosenblatt [6] et l'adaline de Widrow [7]. Ils sont capables d'apprendre à partir d'exemples, même quand ceux-ci revêtent un caractère contradictoire voire évolutif… Mais ces machines neuronales ont leurs déficiences. Ainsi Minsky et Papert mettent en évidence qu'elles ne peuvent résoudre certaines catégories de problèmes logiques simples [8]. Les méthodes symboliques ravissent pour un temps au connexionnisme les fastes des projecteurs de l'actualité

---

1 **James W.:** *Psychology (Briefer Course)* - New York, Holt (1890)

2 **Pitts W., McCulloch W.S.:** *A logical calculus of the ideas immanent in nervous activity* - Bulletin of Mathematical Biophysics 5:115-133

3 **von Neumann J.:** *The Computer and the Brain* - New Haven: Yale University Press (1958)

4 **Hebb D.O.:** *The Organization of Behavior* - New York Wiley (1947)

5 **Lashley K.S.:** *In search of the engram* - in *Society of Experimental Biology Symposium*, No. 4: *Psychological Mechanisms in Animal Behavior*, Cambridge University Press, pp. 454-455, 477-480 (1950)

6 **Rosenblatt F.:** *The perceptron: a probabilistic model for information storage and organisation of the brain* - Psychological review 65: 386-408 (1958)

7 **Widrow B., Hoff M.E.:** *Adaptive switching circuits* - 1960 IRE WESCON Convention Record, New York: IRE, pp. 96-104 (1960)

8 **Minsky M., Papert S.:** *Perceptrons*, Cambridge, MA: MIT Press (1969)

de l'Intelligence Artificielle.

Cependant les télécommunications, en pleine expansion, connaissent un besoin crucial de systèmes adaptatifs pour traiter le signal. Les solides qualités de l'adaline en la matière assurent son succès pratique et donnent lieu à de nombreux dérivés [1]. Ces algorithmes bénéficient d'études mathématiques poussées mais, dans ce contexte, leur aspect neuronal n'est plus mis en valeur. Une grande partie de la théorie statistique des systèmes d'apprentissage a été produite par l'école russe: Tsypkin en étudie l'adaptativité [2], [3] et Vapnik aborde les problèmes de généralisation [4]. Plus à l'ouest, Kohonen traduit ces systèmes en termes de mémoires associatives linéaires à l'aide de régressions et d'équations matricielles [5]: le connexionnisme sous-jacent est estompé. Mais Kohonen sera considéré, avec d'autres tels Amari [6] ou Grossberg [7] restés explicitement "connexionnistes", comme porteur des germes du renouveau des réseaux de neurones formels.

---

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

2 **Tsypkin Y.:** *Adaptation and learning in Automatic systems* - Mathematics in science and ingineering, vol 73, Academic Press (1971)

3 **Tsypkin Y.:** *Foundation of the Theory of Learning Systems* - Mathematics in science and ingineering, vol 101, Academic Press (1973)

4 **Vapnik V.N., Chernovskys A.Y.:** *On the uniform convergence of relative frequencies of events to their probabilities* - Theory of Probability and its Applications, vol 16, No 2, pp. 264-280 (1971)

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

6 **Amari S.I.:** *Neural theory ofasociation and concept-formation* Biological Cybernetics 26: 175-185 (1977)

7 **Grossberg S.:** *Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors* - Biological cybernetics 23: 121-134 (1976)

Le connexionnisme renaît étrangement avec Hopfield dont le modèle [1] obéit à la loi de Hebb et connaît de sévères limitations, similaires à celles de l'adaline et du Perceptron. Quelques temps après elles sont enfin dépassées. Rumelhart [2] propose le Perceptron multicouche, Le Cun qui travaillait sur un modèle proche en fournit, en compagnie de Bottou, une version très efficace [3], [4]. A la même époque, apparaît la machine de Bolzman imaginée par Ackley, Hinton et Sejnowski [5].

Ce bref exposé montre combien l'histoire du connexionnisme est placée sous le signe de la diversité. Certains de ses parrains désireraient le voir se concentrer sur la neurobiologie ou la psychologie, d'autres sur les mathématiques de l'apprentissage ou bien celles du chaos. C'est sans doute pourquoi, aujourd'hui, le monde du connexionnisme est quelque peu schizophrène…

1.2.2 Contexte émotionnel

Il n'est pas inutile de dire quelques mots sur les raisons du regain d'intérêt porté par l'industrie au connexionnisme: progrès soudains ou victoire de la mode ? Il semble que les deux facteurs aient joué.

Parmi les progrès, soulignons en premier lieu l'apparition de nouveaux algorithmes plus efficaces. Alliée à l'accroissement des performances des ordinateurs (fort utile lors des phases d'apprentissage ou de compulsion de bases de données), elle a ouvert un champ très vaste d'applications potentielles. Quelques cas bien choisis ont été mieux traités que par les méthodes précédemment employées [6].

Par ailleurs, le "look" neuronal est très seyant. Certains, enflammés, ont mis bruyamment en avant les (faibles) analogies des méthodes industrialisables avec le cerveau humain. Certes,

---

1 **Hopfield J.J.:** *Neural netwoks and physical systems with emergent collective computational abilities* - Proceedings of the National Academy of Sciences 70: 2554-2558 (1982)

2 **Rumelhart D.E., Hinton G.E., Williams R.J.:** *Learning representations by back-propagating errors* - Nature 323:533-536 (1986)

3 **Bottou L., Le Cun Y.:** *SN: A Simulator for Connectionist Models* - Neuro Nîmes 88, pp. 371-382, EC2 ed. (1988)

4 **Le Cun Y.:** *Modèles conexionnistes de l'Apprentissage* - Thèse de doctorat de l'université de Paris 6 (1987)

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

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

historiquement, l'inspiration est venue de là. Mais il est naïf de réconforter son ego humain en imaginant que les tâches jusque là réservées à l'homme, telle la lecture, ne puissent être convenablement traitées que par des machines imitant le cerveau… La mode fit le reste.

Les sciences cognitives sont nées récemment du désir récemment avivé d'obtenir, au sujet de la pensée, la synthèse de la psychologie, des mathématiques, de la physique, de la neurobiologie et de l'informatique. Assurément ces sciences ont des points communs et des éléments utiles à partager. Mais il est cependant à craindre que dans l'immédiat peu de systèmes industriels doivent naître de ce mélange des genres… Peut-être un siècle à venir verra la jonction de ces domaines; pour l'heure il peut sembler prématuré d'en attendre plus qu'un relatif enrichissement mutuel.

Cette thèse porte sur certaines techniques d'apprentissage automatique en vue d'obtenir des systèmes opérationnels. L'approche qui y prévaut est donc mathématique et informatique.

1.2.3 Contexte scientifique

A l'échelle d'une thèse, l'approche mathématique et informatique est encore trop vaste, tant la croissance du connexionnisme a été foudroyante. On peut y distinguer deux familles principales: les réseaux à relaxation stochastique (Hopfield, Boltzmann,…) et les réseaux à relaxation déterministe (Perceptron multicouche, LVQ,…).

La première famille semble pour l'instant handicapée par la lourdeur informatique des algorithmes qu'on lui connaît. Néanmoins, il est raisonnable de penser qu'elle pourrait être à même de bien traiter certaines catégories de problèmes, tels que les problèmes d'optimisation combinatoire.

La seconde famille comporte un grand nombre d'algorithmes efficaces en reconnaissance des formes et en approximation de fonctions, incluant notamment des procédures d'apprentissage par descente de gradient et/ou de relaxation par plus proche voisinage. La plupart des réalisations industrielles déjà opérationnelles sont à porter à son actif. Ces algorithmes disposent d'un capital de départ non négligeable, dans le cadre plus large de la reconnaissance des formes et dans celui de l'optimisation par descente de gradient (stochastique ou bien déterministe) qui ont fait l'objet de nombreuses recherches, études et applications depuis plusieurs décennies.

C'est sur cette seconde famille que nous avons concentré nos efforts.

1.3 Reconnaissance de la parole

1.3.1 Un domaine complexe

La reconnaissance de la parole représente un défi de taille. Malgré de nombreuses recherches souvent conduites par des équipes de qualité, il n'existe pas aujourd'hui de système globalement satisfaisant traduisant les sons articulés que nous émettons en l'information lexicale ou conceptuelle qu'ils véhiculent.

Une chaîne complète de reconnaissance de la parole est un ensemble complexe composé de différents modules spécialisés dans le traitement du signal, la segmentation, la reconnaissance des formes et l'analyse lexicale, syntaxique ou sémantique. Les chaînes de traitements les plus sophistiquées effectuent des aller et retour entre les différents modules afin de résoudre au mieux les ambiguïtés et contradictions qui ne manquent pas d'apparaître dès qu'on s'attaque au langage parlé naturellement.

Nous nous sommes concentrés sur la reconnaissance des formes: les méthodes connexionnistes s'y montrent particulièrement efficaces. Plus précisément, nous avons appliqué et développé nos méthodes pour le décodage acoustico-phonétique (ou acoustico-lexical dans le cas de la parole en mots isolés). Il n'est pas impossible que très bientôt d'autres modules de ces chaînes de traitement soient constitués de systèmes connexionnistes [1].

1.3.2 Systèmes contemporains de décodage acoustique

La difficulté de la reconnaissance du langage parlé naturellement a conduit les chercheurs à travailler dans un premier temps sur des problèmes restreints.

Les systèmes opérationnels à ce jour imposent différentes restrictions d'utilisation. Le système peut être monolocuteur, multilocuteur, ou indépendant du locuteur selon qu'il est adapté à un unique locuteur, à plusieurs ou à n'importe lequel. Le système peut reconnaître la parole en mots isolés, enchaînés (pas d'arrêt long de l'élocution, mais les mots doivent être bien distincts) ou la parole continue. L'environnement peut être calme ou bruité. Les conditions d'enregistrement peuvent inclure une acoustique parfaite et une configuration locuteur/micro constante. Le langage reconnu peut être composé de quelques mots, de nombreux mots ou d'un

1 **Stevenin A., Gallinari P.:** *Concept Prediction for Spontaneous Speech Retranscription* - WCNN (1994)

dictionnaire complet. Les accents, les tons, les voix enrouées,… peuvent ne pas être traitées. La vitesse d'élocution peut être imposée.

Ces restrictions sont justifiées. Un signal de parole de bonne qualité représente un débit de l'ordre de 40 KOctets/sec. La séquence de phonèmes correspondante représente un débit inférieur à 8 octets/sec. Pourtant, en dépit de dons naturels extrêmement sophistiqués, la perception que nous en avons est souvent incertaine, voire erronée: la difficulté du problème est considérable.

Dans ce cadre, concevoir un système meilleur signifie souvent assouplir certaines des contraintes imposées sans dégrader excessivement le taux de reconnaissance.

1.3.3 Utilité des méthodes neuronales pour le décodage acoustique

Les multiples sources de variabilité de la parole (locuteur, intonation, conditions acoustiques, environnement bruyant, vitesse d'élocution et coarticulation,…) la rendent en partie contradictoire et en tout cas confuse. La combinatoire induite par ce signal empêche de se reposer sur des méthodes trop simples comme la comparaison euclidienne de références nombreuses. Il faut modéliser la parole, intégrer ses aspects aléatoires et temporels, en produire des descriptions intermédiaires plus synthétiques…

Dans cette optique, certains méthodes connexionnistes, telles les nombreuses variantes de perceptrons multicouches, disposent de sérieux atouts. La théorie de l'optimisation nous apprend qu'elles constituent des estimateurs de probabilités ou d'espérances. Leur aspect distribué est susceptible de leur conférer une bonne résistance au bruit. Elles sont capables d'intégrer des aspects flous et de traiter des problèmes mal définis avec un moindre dysfonctionnement que des systèmes plus rigides tels les systèmes experts. Comme nous le verrons, leur propriété d'association universelle leur permettent de fournir des codages synthétiques contenant l'essentiel de l'information utile; de plus elles peuvent travailler en coopération active avec une large gamme de méthodes, permettant ainsi de combiner des avantages variés au sein de systèmes modulaires hybrides.

1.4 Démarche fondamentale et appliquée

On peut remarquer que dans le monde, les équipes connexionnistes les plus productives allient théorie et pratique. Cela n'est pas sans raison; il est peut être utile de fournir quelques explications, certaines générales et d'autres spécifiques au domaine connexionniste.

1.4.1 Nécessité d'un travail théorique

Aujourd'hui, il n'existe pas de méthode universelle pour l'apprentissage automatique. Pour guider le développement techniques, pour incorporer des connaissances *a priori*, pour faire coopérer différentes techniques, pour répondre à des besoins spécifiques, il faut faire progresser la théorie.

La capacité des méthodes connexionnistes de traiter à moindre mal les problèmes flous ou mal définis est à double tranchant. Elle peut donner l'illusion qu'une méthode est correctement exploitée alors qu'elle aurait dû donner bien plus. Elle incite à la paresse. Elle peut compromettre toute amélioration ultérieure.

Beaucoup de domaines d'applications disposent déjà de méthodes dont les qualités peuvent être combinées avec celles des réseaux de neurones. Pour que cet espoir de coopération ne soit pas déçu, Il est essentiel d'analyser ces algorithmes avec un formalisme commun.

La nécessité de développements théoriques visant à analyser le comportement prévisible des algorithmes d'apprentissage apparaît clairement. L'étude théorique permet d'établir certaines conditions d'utilisation adéquates ou même optimales. Elle permet également de comprendre quelles modifications apporter aux algorithmes pour changer leur cadre d'utilisation.

De plus, il est fréquent de rencontrer des spécifications qui sortent de l'ordinaire. Il est parfois souhaitable de pouvoir modifier à moindre frais un système conçu pour d'autres circonstances (ex: ajout d'un mot dans un vocabulaire déjà opérationnel) ou de pondérer l'importance des erreurs (ex: ne pas reconnaître à la légère une commande de mise à feu).

Enfin, il y a souvent un écart important entre les performances du système sur la base d'apprentissage et lors de son exploitation. Cela peut être dû à un problème mal posé; il est alors souvent possible d'évaluer qualitativement l'importance de cette erreur de conception. Mais surtout la théorie de la généralisation a mis en lumière les différences qu'on pouvait attendre entre le comportement d'un système sur l'ensemble fini des exemples utilisés pour son apprentissage et le comportement du système sur l'ensemble de la distribution dont les exemples sont issus. Cette théorie conduit naturellement à analyser les algorithmes et induit des améliorations permettant de réduire les effets de la finitude des exemples en incitant le système à utiliser des configurations plus robustes.

1.4.2 Nécessité d'un travail appliqué

La complexité des réseaux de neurones formels ne permet en général pas à la théorie de prévoir leur comportement avec certitude. Une validation pragmatique apparaît alors nécessaire.

Dans un premier temps la pratique peut très bien se contenter de problèmes “jouets”, en particulier quand il s’agit de mettre au point un nouvel algorithme. Mais rapidement l’utilité de ces problèmes se tarit: il est souvent facile de modifier une méthode pour résoudre “sur mesure” un problème synthétique. Une tâche plus ambitieuse est nécessaire pour valider et faire progresser la méthode.

Par ailleurs, la recherche fondamentale a ses limites tant morales qu’intellectuelles. Il est souhaitable que toute recherche fondamentale puisse un jour se rendre utile. Mieux encore, pour se développer pleinement et se transcender, elle a la possibilité de trouver dans la réalité retorse un défi immense et une inspiration sans cesse renouvelée.

Cette démarche est garante d’un effort de qualité: dans un monde changeant où la science se fractionne en multiples entités trop souvent aveugles les unes des autres, revenir au réel fournit une mesure imparfaite mais intégralement représentative du chemin parcouru. Elle permet d’éviter la fuite en avant qui conduit au monde virtuel des microcosmes stériles.

1.5 Chronologie

Le plan du mémoire présenté au § 1.6 est synthétique; il ne respecte pas la chronologie du travail de thèse. Les lignes qui suivent ont pour objet de décrire cette chronologie et de replacer les composants de ce travail dans leur contexte.

Lors de l'été 1989, j'ai effectué mon stage de DEA dans l'équipe réseaux connexionnistes de l'EHEI. Le travail de notre équipe, outre ses tâches de recherche fondamentale, était également consacré à des applications développées à l'occasion de projets de recherche ESPRIT Pygmalion et BRAIN. Je fus chargé de réaliser une librairie de traitement du signal (transformées de Fourier, bancs de filtres, cepstres, analyse linéaire prédictive…), puis je participais aux expériences de reconnaissance de la parole et de réduction du bruit.

Après ce stage, j'intégrais l'équipe pour la durée de ma thèse. Notre équipe migra au LRI de la Faculté d'Orsay. A l'époque, quelques équipes, notamment celle de C. Wellekens et H. Bourlard et celle de H. Sakoe [1] essayaient de faire collaborer réseaux de neurones et techniques plus

1 **Sakoe H., Isotani R., Yoshida K. Iso K. Watanabe T.:** *Speaker Independant Word Recognition Using Dynamic Programming Neural Networks* - Proceeding of ICASSP, pp 107-110 (1988)

classiques [1]. Je développais alors le système de reconnaissance de la parole française de Pygmalion et Léon Bottou, avec qui je travaillais, me conseilla d'orienter mes expériences vers les systèmes modulaires, hybrides de réseaux neuronaux multicouches et de programmation dynamique adaptative. Les premières expériences de tels systèmes hybrides furent menées au début de l'année 1990. Les résultats, positifs, firent l'objet d'une communication [2]. Le domaine semblait d'autant plus prometteur que les communications de bonne qualité se multipliaient sur le sujet [3].

Parallèlement, nous redécouvrions à travers les travaux de Y.A.Z. Tsypkin un contexte théorique, celui de l'optimisation par descente de gradient stochastique. Patrick Gallinari et Léon Bottou ont mis en lumière que ce formalisme permettait de réexprimer de façon homogène la quasi-totalité des systèmes neuronaux adaptatifs. De plus, il était parfaitement compatible avec l'algorithme de rétropropagation du gradient qui avait contribué grandement à l'efficacité calculatoire de certaines techniques connexionnistes. Ceci fournissait un cadre théorique pour l'exploration expérimentale de systèmes adaptatifs modulaires, dont l'optimisation se fait conjointement dans tous ses modules. Ce cadre fut explicité au cours de l'année 1990 [4].

Je participais activement à l'élaboration de cette approche. Ceci me permis de construire un nouveau système d'apprentissage hybride pour la reconnaissance de la parole.

Le projet Pygmalion devait se terminer au début de l'année 1991. Une base de donnée avait été constituée. De multiples expériences avaient été menées, permettant de comparer l'efficacité de nombreuses techniques. Ces comparaisons, ainsi que la définition du nouveau système hybride et ses premiers résultats firent l'objet de plusieurs communications [5], [6], [7]. Ceci

---

1 **Bourlard H., Wellekens C.:** *Links between Markov models and multi-layer perceptrons* - Proceedings of NIPS 1, Morgan Kauffman Publi., pp 502-510. (1988)

2 **Driancourt X., Bottou L.:** *TDNN Extracted Features* - Actes de Neuronimes, Nimes (1990)

3 **Bengio Y., Cardin R., De Mori R., Normandin Y.:** *A hybrid coder for hidden Markov models using a recurrent network* - Proceeding of ICASSP 90, pp 537-540 (1990)

4 **Bottou L.:** *A Framework for the Cooperation of Learning Algorithms* - Proceeding of NIPS (1990)

5 **Driancourt X., Bottou L., Gallinari P.:** *Multi-Layer Percpetron, Learning Vector Quantization and Dynamic Programming: Comparison and Cooperation* - Proceedings of IJCNN (1991)

6 **Driancourt X., Bottou L., Gallinari P.:** *Comparison and Cooperation of Several Classifiers* - Proceedings of ICANN (1991)

7 **Driancourt X.:** *Extraction de primitives caractéristiques du signal par réseaux neuronaux pour la reconnaisance de la parole* - Rapport de recherche DRET, CJC 89/1591 (1991)

constituait un contexte expérimental attractif dans lequel j'allais continuer de travailler quelques mois, en parallèle avec un travail plus fondamental portant sur l'étude de l'apprentissage global au sein de systèmes modulaires [1] et sur les propriétés de convergences (théorème § 2.4.3).

A la même époque, d'autres systèmes proches portant sur la reconnaissance de mots enchaînés ou de la parole continue, firent l'objet de publications [2], [3], [4].

Mes travaux se poursuivirent avec succès et conduisirent à une série d'articles et de communications orales, lors de conférences internationales en 1992 [5], [6], [7].

La base de parole continue TIMIT, que l'on pouvait se procurer facilement et gratuitement, apparaissait comme une base de référence dans le domaine du décodage acoustico-phonétique de la parole américaine continue. Elle était volumineuse et nous eûmes les moyens de l'exploiter pour la reconnaissance de la parole seulement en 1992. A ce moment là, j'avais adapté l'essentiel de notre environnement logiciel pour le traitement de la parole continue. Nous avions défini deux directions de recherche en matière de systèmes hybrides, l'une utilisait les réseaux de neurones multicouches comme module de filtrage, l'autre les utilisait comme module prédictif. Abdelhamid Mellouk devait conduire avec succès le développement et les expériences du système prédictif [8]. Pour ma part, mon travail de thèse prenait fin avec mon incorporation au sein de l'armée française.

---

1 **Schwenk H., Gallinari P., Driancourt X. :** *a modular system which improves the topological maps* - Proceedings of IJCNN (1992)

2 **Bengio Y., De Mori R., Flammia G. and Kompe R.:** *Gloabal Optimization of a Neural Network-Hidden Markov Model Hybrid* - Proceedings of IJCNN (1991)

3 **Bourlard H., Morgan N.:** *A continuous Speech Recognition System Embedding MLP into HMM* - Proceedings of NIPS 2 pp. 502-510, Morgan Kaufman (1991)

4 **Haffner P., Fanzini M., Waibel A.: Integrating** *Time alignment and neural networks for high performance continuous speech recognition* - Proceeding of ICASSP, S2.17, pp 105-108 (1991)

5 **Driancourt X., Gallinari P.:** *A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer perceptron* - Proceedings of ICASSP (1992)

6 **Driancourt X., Gallinari P.:** *An empirical risk optimizer for speech recognition* - Proceedings of IJCNN (1992)

7 **Driancourt X., Gallinari P.:** *Empirical risk optimisation: neural networks and dynamic programming* - Proceedings of NNSP (1992)

8 **Mellouk A., Gallinari P.:** *A Discriminative Neural Prediction System for Speech Recognition System* - ICASSP (1993)

1.6 Plan du mémoire

Le plan de ce mémoire provient des considérations mentionnées dans l'introduction.

La **première partie** (fondamentale) est consacrée aux outils théoriques dont on dispose en reconnaissance des formes par algorithmes de descente de gradient. Elle comprend trois chapitres. Le second chapitre traite la théorie de l'apprentissage par descente de gradient, en particulier de gradient stochastique; nous y étudions les notions de gradient total, global, local, les contraintes, les algorithmes du second degré et leurs approximations et nous fournissons un théorème original de convergence pour les algorithmes de descente de gradient stochastique. Le troisième chapitre se concentre sur les problèmes de classification et de reconnaissance des formes.

La **seconde partie** (algorithmique) traite des systèmes de traitement de l'information *a posteriori*. Nous y présentons de nombreux algorithmes. Les algorithmes faisant appel à l'apprentissage sont presque toujours interprétables comme des algorithmes de descente de gradient. Cette optique relativement originale de notre équipe fonde notre approche algorithmique et expérimentale. Le quatrième chapitre est consacré aux systèmes généraux de traitement de l'information *a posteriori*. Nous y étudions notamment quelques systèmes modulaires originaux. Le cinquième chapitre est consacré aux systèmes de traitement de l'information *a posteriori* ayant une nature temporelle.

La **troisième partie** (expérimentale) est consacrée à la parole et à sa reconnaissance, notamment par des systèmes modulaires hybrides. Le sixième chapitre propose un système modulaire hybride original pour la reconnaissance de la parole, faisant appel à la programmation dynamique et au Perceptron multicouche. Nous l'introduisons progressivement en retraçant notre démarche expérimentale. Nous fournissons des tableaux comparatifs de performances. Nous en analysons les principales caractéristiques et proposons quelques variantes.

Suit la conclusion de cet ouvrage.

Les Annexes regroupent des informations qui, si elles avaient été disposées au milieu de l'ouvrage, aurait probablement nuit à sa progression pédagogique. La première annexe propose une description des processus biologiques de production et de reconnaissance de la parole; nous y présentons les traitements de bas niveaux couramment utilisés pour compresser la parole en conservant un maximum d'informations utiles à sa reconnaissance. La deuxième annexe reproduit une de nos publications décrivant un système modulaire combinant perceptrons multicouches et cartes topologiques.

2 Apprentissage par descente de gradient stochastique

Une des techniques fondamentales employées pour l'utilisation de données *a posteriori* est l'apprentissage par descente de gradient stochastique.

Nous avons privilégié cette technique. Elle conduit à des procédures d'apprentissage particulièrement adaptées aux bases de données redondantes. Elle permet l'optimisation de systèmes modulaires (cf. § 4.4, § 5.5). Elle fournit un formalisme unifié permettant d'exprimer ou de réexprimer de nombreux systèmes d'apprentissage (dont beaucoup de techniques neuronales) par adaptation ou par optimisation (cf. § 4.2, § 4.3, § 5.2, § 5.4).

Ce chapitre est consacré aux algorithmes d'apprentissage par descente de gradient stochastique. Il a été grandement inspiré du livre de Y.A.Z. Tsypkin [1] et de la thèse de L.Y. Bottou [2].

Nous introduirons la problématique de l'apprentissage par optimisation de fonctions de coûts (cf. § 2.1). Nous étudierons les techniques de descentes de gradient (cf. § 2.2), notamment les techniques stochastiques (cf. § 2.2.2 et suivantes). Nous discuterons la problématique de l'apprentissage et de l'adaptation (cf. § 2.3). Nous présenterons les méthodes adaptatives du second ordre (cf. § 2.3.2). Puis nous énoncerons et démontrerons un théorème de convergence

---

1 **Tsypkin Y.:** *Foundation of the Theory of Learning Systems* - Mathematics in science and ingineering, vol 101, Academic Press (1973)

2 **Bottou L.:** *Une approche théorique de l'apprentissage connexionniste; Application à la reconnaissance de la parole* - Thèse de doctorat, Univerité Paris XI, Orsay (1991).

original dont on dispose pour les algorithmes stochastiques (cf. § 2.4).

2.1 Apprentissage par optimisation

2.1.1 Critère d'optimisation ou fonction de coût

A défaut d'exprimer analytiquement une tâche, il arrive fréquemment que l'on sache mesurer l'(in)adéquation d'un système envers elle. L'apprentissage peut alors consister en l'optimisation des paramètres du système en fonction de cette mesure. Pour la désigner, on parle de *fonction de coût*, de *critère d'optimisation* ou encore de *fonctionnelle*.

La fonction de coût est *a priori* un critère global, en ce sens qu'on juge le comportement du système dans son ensemble et non pas dans la multiplicité des cas de figure qu'il sera amené à traiter: on parle de critère *local* (un cas de figure) par opposition au critère *global* (tous les cas de figure connus) ou *total* (tous les cas de figure concevables).

Le plus souvent le critère total n'est rien d'autre que l'espérance mathématique d'un critère local:

$$C(\mathbf{w}) = \int_{\mathbf{X}} J(\mathbf{x},\mathbf{w})\ p(\mathbf{x})\ d\mathbf{x} \ \stackrel{\Delta}{=}\ E_{\mathbf{x}}\{\ J(\mathbf{x},\mathbf{w})\ \} \qquad (2\text{-}1)$$

où l'ensemble des exemples $\mathbf{X}$ est décrit par la variable $\mathbf{x}$ de densité de probabilité $p(\mathbf{x})$. La fonction J est le critère local d'optimisation. Sauf mention contraire, on supposera dans la suite que J est positive et donc que C est minorée sur l'espace des paramètres.

En principe on ne dispose pour l'apprentissage que d'un échantillon dénombrable (souvent fini) $(\mathbf{X},\hat{p})$ composé d'exemples $\mathbf{x}_i$, l'intégrale devient une simple somme discrète et on parle de critère global; ce critère est en fait une *estimation* du critère total: on parle de *fonctionnelle empirique*.

Il arrive que l'aspect dénombrable de la distribution pose problème, en raison de la discontinuité que cela engendre dans la fonction indicatrice de la distribution. On peut retrouver le cas continu en appliquant aux exemples des transformations stochastiques selon des lois connues *a priori* (cf. § 2.2.3).

2.1.2 Optimisation

Quand on manque de connaissance *a priori*, en particulier quand on n'a pas la possibilité de

déterminer $p(\mathbf{x})$, on se trouve dans l'impossibilité de calculer directement le(s) jeu(x) de paramètres $\mathbf{w}$ minimisant la fonction de coût total (2-1). On doit alors utiliser des algorithmes d'apprentissage.

L'optimisation se fait souvent par ajustement progressif des paramètres, et notamment par des algorithmes de la famille des descentes de gradient [1], [2], [3]. Il arrive cependant que, faute d'information ou faute de régularité du problème, la descente de gradient ne soit pas possible. On doit alors faire appel à l'optimisation non-dérivative [4], [5], [6]. Il faut souligner que, même quand on peut employer les méthodes dérivatives, l'optimisation non-dérivative ne doit pas être négligée [7].

On distingue les algorithmes *discrets* où on réévalue les paramètres du système par étapes successives, par opposition aux algorithmes *continus* où la réévaluation des paramètres se fait continûment au cours du temps. Les algorithmes continus conviennent bien à l'électronique analogique et les algorithmes discrets à l'électronique digitale et à l'informatique. Nous nous limiterons ici aux algorithmes discrets.

2.1.3 Cas différentiable

Le cas différentiable est celui que nous allons étudier ici. L'hypothèse de différentiabilité permet d'employer des techniques d'optimisation par descente de gradient.

Quand la fonction de coût est définie sur un ouvert, sur lequel elle est partout différentiable, on peut qualifier ses extrema par un critère d'optimalité de la forme:

$$\nabla C(\mathbf{w}) = E_{\mathbf{x}} \{ \nabla_{\mathbf{w}} J(\mathbf{x},\mathbf{w}) \} = \int_{\mathbf{X}} \nabla_{\mathbf{w}} J(\mathbf{x},\mathbf{w})\, p(\mathbf{x})\, d\mathbf{x} = \mathbf{0} \qquad (2\text{-}2)$$

---

1 **Polack E.:** *Computational Methods in Optimization* - Academic Press, New York (1971)

2 **Acton F.S.:** *Numerical Methods that Work* - Harper and Row, pp 467-468 (1970)

3 **Tsypkin Y.:** *Foundation of the Theory of Learning Systems* - Mathematics in science and ingineering, vol 101, Academic Press (1973)

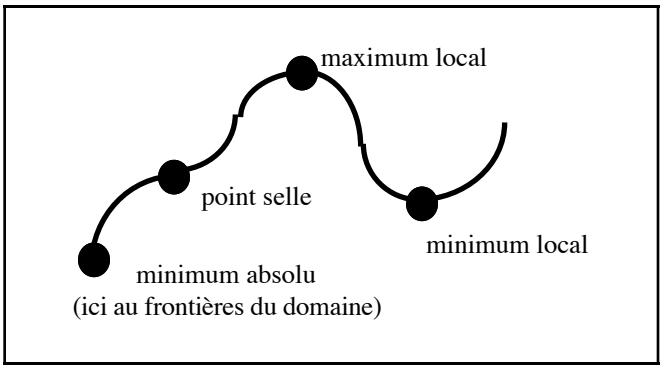
4 **Brent R.P.:** *Algorithms for minimization without Derivatives* - Prentice-Hall, Englewood Cliffs, N.J. (1973)

5 **Polack E.:** *Computational Methods in Optimization* - Academic Press, New York (1971)

6 **Nelder J.A., Mead R.:** *Computer Journal* - Vol 7, pp 308, (1965)

7 **Press W.H., Flannery B.P. Teukolsky S.A., Vetterling W.T.:** *Numerical recipes: the art of scientific computing* - Cambridge University Press (1986)

Ce critère est simplement un critère **nécessaire** de minimalité locale. Il n'est pas suffisant car les maxima locaux et les points selles le vérifient également.



Critère nécessaire d'optimalité

Le critère de nullité du gradient est un critère nécessaire mais pas suffisant pour établir la minimalité locale d'une configuration de paramètres maxima et points selles le vérifient également.

Ce critère d'optimalité a une grande importance; nous le retrouverons à l'occasion de l'étude des algorithme d'apprentissage adaptatif optimaux et quasi-optimaux ainsi que pour l'étude du risque moyen. Elle traduit l'achèvement de l'optimisation; en général c'est la seule propriété intéressante que l'on peut garantir après la convergence d'un algorithme de descente de gradient.

2.2 Optimisation par descente de gradient

2.2.1 Algorithme déterministe

L'optimisation d'une fonction de coût peut se faire de nombreuses façons. Quand elle est suffisamment régulière, on peut notamment procéder par descente de gradient:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} E_{\mathbf{x}} \{ \nabla_{\mathbf{w}} J(\mathbf{x},\mathbf{w}_n) \} \qquad (2\text{-}3)$$

où Γ_n est une matrice symétrique définie positive (donc pour laquelle il existe un repère orthonormé dans lequel elle est diagonalisable; ses valeurs propres sont toutes strictement positives). Dans le cas où Γ_n est de la forme εI (multiple de l'identité), la littérature emploie l'expression *pas d'apprentissage* (learning step). Nous emploierons cette expression également pour la forme générale de Γ_n, en distinguant les deux cas par l'adjectif *anisotrope* dans le cas général Γ_n et *isotrope* dans le cas particulier ε I. En général on ne peut évaluer la quantité (2-3) sinon en l'*estimant* à l'aide d'un nombre fini d'exemples:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} \hat{E}_{\mathbf{x}} \{ \nabla_{\mathbf{w}} J(\mathbf{x},\mathbf{w}_n) \} \qquad (2\text{-}4)$$

Comme tous les algorithmes de la famille des descentes de gradient, cet algorithme est parfois d'utilisation délicate et pose des problèmes de conditionnement et d'initialisation.

La convergence de la fonction de coût est assurée pour peu que (condition suffisante):

- la fonction de coût soit minorée,
- le gradient soit borné et
- la matrice symétrique définie positive Γ_n tend asymptotiquement vers $\mathbf{0}$.

La convergence de l'algorithme vers un point vérifiant le critère d'optimalité (2-2) est garantie si (condition suffisante) en plus des conditions précédentes, les valeurs propres de la matrice Γ_n n'engendrent que des séries divergentes et que le gradient n'approche $\mathbf{0}$ que dans de tels voisinages (pas de fuites vers l'infini).

Ces conditions donnent une garantie théorique et non une garantie pratique. Il est en effet difficile de faire la distinction entre la fin de l'apprentissage et une évolution désespérément lente du système. Les problèmes de *conditionnement* ont donc reçu une attention particulière de ceux qui mettent au point les procédures d'apprentissage.

Il faut souligner que le minimum de C obtenu à l'issue de l'apprentissage est *a priori* un simple minimum local; en général ce n'est pas le minimum global. Il est donc important de partir d'une configuration raisonnable de paramètres pour que les paramètres n'aient pas une probabilité trop grande de rester dans un minimum local trop décevant. De même que les problèmes de conditionnement, les problèmes d'*initialisation* reçoivent souvent une attention particulière.

Ces deux points délicats sont inhérents à la notion de descente de gradient.

Dans le cas (2-1), avant d'améliorer la configuration de paramètres, l'algorithme de descente de gradient déterministe examine le gradient de la fonction de coût local pour tous les exemples $\mathbf{x}$ qui composent la distribution empirique. On parle d'algorithmes de gradient *déterministe*: *total* (par référence aux distributions théoriques) ou *global* (par référence aux distributions empiriques).

Or, bien souvent, la distribution empirique est une base de donnée fortement redondante. L'algorithme de descente de gradient déterministe ne tire pas parti de l'existence de ces redondances et souffre d'une vitesse d'apprentissage d'autant plus réduite que la distribution empirique contient de nombreux exemples.

Comme nous allons le voir, l'algorithme stochastique remédie radicalement à ce défaut, au prix de sacrifices souvent secondaires (plus grande sensibilité au pas d'apprentissage, moindre certitude de convergence).

2.2.2 Algorithme stochastique

Par opposition à la descente de gradient *déterministe*, la descente de gradient *stochastique* [1], [2] consiste à effectuer une modification des paramètres du système sur la simple base du gradient relatif au critère local (d'un exemple):

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n) \ , \qquad (2\text{-}5)$$

où Γ_n est une matrice symétrique définie positive (cf. 2-3) et où $\mathbf{x}_n$ est l'exemple utilisé à la $n^{ième}$ itération pour l'apprentissage. On suppose que $\mathbf{x}_n$ est tiré aléatoirement de la distribution ($\mathbf{X}$, p) étudiée. Cette procédure est également connue sous le terme de descente de gradient généralisée (generalized probabilistic gradient descent).

L'algorithme de descente de gradient stochastique n'est pas une procédure de descente de gradient au sens strict du terme: chaque itération de l'algorithme ne garantit pas une réduction de la fonction de coût globale. Cette procédure repose sur l'espoir que cette réduction aura lieu *en moyenne*, c'est à dire que les oppositions entre les gradients stochastiques se moyenneront sans rendre instable ou cyclique le processus d'apprentissage. Comme nous le verrons au § 2.4.3, la convergence est assurée dans de nombreux cas.

En échange, cette procédure permet d'espérer d'une part une moindre sensibilité aux phénomènes de minima locaux et d'autre part une convergence moins coûteuse en temps de calcul puisqu'on met à jour les paramètres du système exemple après exemple au lieu de les mettre à jour après l'examen conjoint de tous les exemples disponibles, tirant ainsi parti de leur redondance. De plus, elle est naturellement adaptative, ce qui permet de traiter des phénomènes évolutifs.

2.2.3 Opérateurs compatibles, opérateurs de lissage

Les opérateurs laissant invariant en moyenne la fonction de coût, selon l'équation:

$$\hat{E}_{\mathbf{x}} \{ \mathbf{B} \nabla_{\mathbf{w}} J(\mathbf{x},\mathbf{w}_n) \} = \hat{E}_{\mathbf{x}} \{ \nabla_{\mathbf{w}} J(\mathbf{x},\mathbf{w}_n) \} \qquad (2\text{-}6)$$

conservent les propriétés de convergence du théorème § 2.4.3. Les opérateurs de lissage tels que la moyenne du gradient stochastique à court terme ou dégressive selon l'ancienneté des exemples

---

1 **Fabian V. :** *Pregled deterministichych a stochastilhych approximacnich method pro minimalizaci funkci* - Kybernetica (Kiev) 1,6 (1965)

2 **Ermolyev Y. M., Nekrilova Z. V. :** *Method of stochastic gradient and its application* - First seminary on the theory of optimal decisions, Kiev (1967)

passés. Ces opérateurs permettent d'adoucir la descente de gradient stochastique.

Une autre utilisation des opérateurs de lissage consiste à transformer une distribution empirique originelle. On peut notamment appliquer des transformations stochastiques aux exemples selon des lois connues *a priori*. Cela permet (1) d'étendre une distribution empirique pour mieux représenter la distribution totale et (2) de travailler avec une distribution empirique transformée continue, par opposition à une distribution empirique originelle souvent discrète, donc ayant une fonction indicatrice discontinue…

2.2.4 Une fonction de coût alternative

La fonction de coût (2-1) peut donner lieu à de multiples cas particuliers. On peut proposer notamment:

$$C(\mathbf{w}) = E_{\mathbf{y}} \left\{ \Phi \left(E_{\mathbf{x}} \left\{ \mathbf{q}(\mathbf{x},\mathbf{w}) \right\} , \mathbf{s}(\mathbf{y}, \mathbf{w}) \right) \right\} \qquad (2\text{-}7)$$

où $\mathbf{x}$ et $\mathbf{y}$ sont deux variables aléatoires indépendantes; $\mathbf{q}$ et $\mathbf{s}$ deux fonctions à valeurs vectorielles, Φ une fonction à valeur dans $\mathbb{R}$.

Cette formule peut paraître absconse. On peut cependant lui donner une interprétation physique. Il arrive que l'on doive prendre des décisions relatives à un cas particulier au sein d'un environnement. La variable aléatoire $\mathbf{y}$ décrit le cas particulier et la variable aléatoire $\mathbf{x}$ décrit les représentants de l'environnement. On en résume l'influence respective à la moyenne sur $\mathbf{x}$ de la fonction $\mathbf{q}$ et à la fonction $\mathbf{s}$. L'expression (2-6) donne un rôle presque symétrique aux variables aléatoires $\mathbf{y}$ et $\mathbf{x}$.

Un exemple en est une fonction d'analyse d'un marché. Le vecteur $\mathbf{w}$ définit les caractéristiques organiques et fonctionnelles de produits, ce qui inclue leurs prix et les caractéristiques des actions promotionnelles dont ils sont l'objet. Le vecteur $\mathbf{x}$ définit les caractéristiques statistiques utiles d'un consommateur de la population cible. Le vecteur $\mathbf{y}$ définit le marché des matières et services entrant dans la composition des produits. La fonction $\mathbf{q}$ détermine les fréquences d'achats en fonction de la population des consommateurs $\mathbf{x}$ et des prix et des caractéristiques fonctionnelles des produits $\mathbf{w}$. La fonction $\mathbf{s}$ détermine la structure des marges en fonction des prix et caractéristiques organiques spécifiés des produits $\mathbf{w}$ et des prix d'achats $\mathbf{y}$. La fonction Φ détermine le bénéfice de la gamme de produits. On cherche à optimiser le bénéfice sur une période de temps.

Les deux expressions sont équivalentes. La fonction de coût (2-7) est un cas particulier de la fonction (2-1). Réciproquement (2-1) est un cas particulier de (2-7). En effet quand :

$$\Phi(\mathbf{q}, \mathbf{s}) \equiv \mathbf{s} \quad , \quad \dim(\mathbf{s}) = 1 \text{ et avec la notation : } \mathbf{S} = J \tag{2-8}$$

l'expression (2-7) reprend la forme (2-1).

La seule différence réside dans le naturel avec lequel l'une ou l'autre de ces deux expressions permet de modéliser tel ou tel phénomène.

Le gradient de C s'écrit ici:

$$\nabla C(\mathbf{w}) = E_{\mathbf{xy}} \left\{ \begin{array}{l} \left(\frac{d\, \mathbf{q}^T(\mathbf{x},\mathbf{w})}{d\, \mathbf{w}} \right) \nabla_{\mathbf{q}} \Phi \left(\bar{\mathbf{q}} , \mathbf{s}(\mathbf{y},\mathbf{w}) \right) \\ + \left(\frac{d\, \mathbf{s}^T(\mathbf{y},\mathbf{w})}{d\, \mathbf{w}} \right) \nabla_{\mathbf{s}} \Phi \left(\bar{\mathbf{q}} , \mathbf{s}(\mathbf{y},\mathbf{w}) \right) \end{array} \right\} \tag{2-9a}$$

$$\bar{\mathbf{q}} = E_{\mathbf{x}} \{ \mathbf{q}(\mathbf{x},\mathbf{w}) \} \tag{2-9b}$$

On déduit de cette expression un algorithme "doublement" stochastique: afin de conserver un caractère stochastique à cet algorithme, *on va devoir estimer* $\bar{\mathbf{q}}$ *adaptativement*, car $\bar{\mathbf{q}}$ dépend de la configuration de paramètres $\mathbf{w}$ que l'on tente d'optimiser et que l'on modifie à chaque étape de l'apprentissage. Cette évaluation adaptative va être utilisée pour évaluer le gradient stochastique de $\bar{\mathbf{q}}$ à l'étape n:

$$\begin{aligned} \mathbf{w}_{n+1} = \mathbf{w}_{n-1} \\ - \Gamma_{1,n} \Big[& \left(\frac{d\, \mathbf{q}^T(\mathbf{x}_n,\mathbf{w}_n)}{d\, \mathbf{w}} \right) \nabla_{\mathbf{q}} \Phi \left(\bar{\mathbf{q}}_n , \mathbf{s}(\mathbf{y}_n,\mathbf{w}_n) \right)) \\ & + \left(\frac{d\, \mathbf{s}^T(\mathbf{y}_n,\mathbf{w}_n)}{d\, \mathbf{w}} \right) \nabla_{\mathbf{s}} \Phi \left(\bar{\mathbf{q}}_n , \mathbf{s}(\mathbf{y}_n,\mathbf{w}_n) \right) \Big] \end{aligned} \tag{2-10a}$$

$$\bar{\mathbf{q}}_{n+1} = \bar{\mathbf{q}}_n - \Gamma_{2,n} \left(\bar{\mathbf{q}}_n - \mathbf{q}(\mathbf{x}_n , \mathbf{w}_n) \right) = (1 - \Gamma_{2,n}) \bar{\mathbf{q}}_n + \Gamma_{2,n} \bar{\mathbf{q}}_n \tag{2-10b}$$

Cette technique n'est pas sans problèmes: si l'évaluation adaptative de $\bar{\mathbf{q}}$ est trop imprécise tout le travail d'optimisation de $\mathbf{w}$ est faussé. Cet algorithme est potentiellement beaucoup plus instable que l'algorithme (2-5) (cf. Annexe 2).

Il faut insister également sur l'importance du réglage de $\Gamma_{1,n}$ et de $\Gamma_{2,n}$. Ainsi dans l'exemple de la fonction d'analyse du marché, si on suppose que le comportement du consommateur $\mathbf{x}$ est stable, et que le comportement des coûts $\mathbf{y}$ est très chaotique, on pourra utiliser des valeurs telles que: $\Gamma_{2,n} = \frac{1}{n}$ et $\Gamma_{1,n} = \frac{1}{2}$.

2.2.5 Descente de gradient sous contrainte

Il est souvent utile d'imposer des contraintes aux paramètres d'une descente de gradient. On peut

imaginer deux stratégies pour assurer le respect de contraintes.

- La première stratégie consiste à exiger que les contraintes soient vérifiées à chaque étape de l'apprentissage.

  C'est généralement le cas quand on optimise les critères de nature probabiliste (MMI, MLE, MDI, MAP…, cf. § 3.3) dont on désire conserver la cohérence tout au long de l'apprentissage (e.g. estimation des probabilité sommant à 1). De tels processus d'optimisation sont utilisés notamment pour l'apprentissage des modèles de Markov cachés (HMM) utilisés pour la reconnaissance de formes de natures temporelles et notamment la parole.

  Cette stratégie est également employée quand on désire traduire une connaissance *a priori* simple à implémenter. Ainsi de nombreux systèmes incorporent des connaissances géométriques sur le problème (invariance par translation, par symétrie…) à l'aide d'implantation simple (partage de paramètres ayant un rôle identique à une translation ou une symétrie… près)

- La deuxième stratégie consiste à exiger que les contraintes soient vérifiées à la fin de l'apprentissage. Elle conduit généralement à des algorithmes plus coûteux et moins sûrs. Cependant elle est souvent incontournable, car moins sensible aux minima locaux, plus générale et naturellement adaptative.

  Pour cela, on emploie généralement la technique des multiplicateurs lagrangiens.

Nous allons étudier rapidement les outils dont on dispose au service de ces deux stratégies.

2.2.5.1 Vérification des contraintes au long de l'apprentissage

Pour qu'une contrainte soit vérifiée tout au long de l'apprentissage, il est généralement nécessaire d'exprimer analytiquement les règles d'ajustement des paramètres, respectant la perpétuelle validité de la contrainte. Aussi, il n'existe pas de méthode générale pour tout système et toute contrainte garantissant une telle invariance.

Cependant quelques grandes familles de systèmes ou de contraintes disposent de méthodes relativement générales.

Ainsi il est aisé de garantir l'égalité de certains paramètres. Pour cela, il suffit d'additionner les gradients des paramètres considérés séparément (au besoin, le conditionnement des algorithmes pourra tenir compte du nombre de partage d'un paramètre, afin d'éviter une réévaluation brutale, trop sensible aux phénomènes du second degré). L'égalité de certains paramètres permet souvent

d'incorporer des connaissances à priori telles que les invariances remarquables de la distribution pour laquelle on optimise un système. Ainsi la reconnaissance de caractères est invariante par translation, et dans une large mesure, par rotation, homothétie… De nombreux systèmes de reconnaissance de caractères rendent compte de certaines invariances par partage de paramètres ayant un rôle identique par translation…

On désire souvent garantir des contraintes normatives. Ainsi, les règles de réévaluation des modèles de Markov cachés tiennent presque toujours compte des contraintes naturelles des jeux de probabilité (somme égale à 1, chaque probabilité comprise entre 0 et 1…). Si les règles de réévaluation avec lesquelles on travaille ne respectent pas spontanément ces contraintes, il est généralement facile de les modifier en leur adjoignant quelques normalisations adéquates (division par la somme des pseudo-probabilités,…). De plus, ces contraintes permettent souvent d'atteindre directement des cas limites (pas d'apprentissage tendant vers l'infini, suivit de normalisation). C'est le cas de l'estimation-maximisation employée par l'algorithme de Baum.

De même, il est possible de forcer le respect de contraintes à l'aide d'une reformulation du problème, notamment par un changement de paramètres. Ainsi, pour respecter la contrainte $x>0$ il est possible de remplacer x par e^z ou z est réel.

2.2.5.2 Vérification des contraintes en fin d'apprentissage

Nous détaillons ici la méthode des multiplicateurs lagrangiens dans le cas des contraintes de type égalité. Celle-ci est conçue pour assurer, en fin d'apprentissage, le respect de contraintes de la forme:

$$\mathbf{g}(\mathbf{w}) = E_{\mathbf{z}} \{ \mathbf{h}(\mathbf{z},\mathbf{w}) \} = \mathbf{0} = (g_1(\mathbf{w}),\ldots,g_I(\mathbf{w}))^T \qquad (2.11)$$

où la variable aléatoire $\mathbf{z}$ est indépendante des variables aléatoires $\mathbf{x}$ (et $\mathbf{y}$) introduites aux paragraphes précédents. On remarquera qu'une telle contrainte peut revêtir un caractère évolutif (la contrainte change de nature dans le temps).

En supposant $\mathbf{g}$ dérivable, on peut employer la méthode des multiplicateurs Lagrangiens, selon laquelle tout extremum $\mathbf{w}$ de la fonction de coût C sous contraintes (2-10) est caractérisé par:

$$\begin{cases} \nabla C(\mathbf{w}) \in \mathbf{Vect} \{ \nabla g_i \} \\ \mathbf{g}(\mathbf{w}) = E_{\mathbf{z}} \{ \mathbf{h}(\mathbf{z},\mathbf{w}) \} = \mathbf{0} \end{cases} \qquad (2.12)$$

où $\mathbf{Vect}$ désigne l'opérateur qui, à une famille de vecteur, associe l'espace vectoriel qu'ils engendrent. La condition (2-12) peut se réécrire sous la forme:

$$\begin{cases} \exists\ \lambda : \nabla\ C(\mathbf{w}) = \lambda^T\ \nabla\ \mathbf{g} \\ \mathbf{g}(\mathbf{w}) = E_{\mathbf{z}}\ \{\ \mathbf{h}(\mathbf{z},\mathbf{w})\ \} = \mathbf{0} \end{cases} \tag{2.12bis}$$

Dans le cas de la fonction de coût (2-6), la condition d'optimalité s'écrit:

$$\begin{aligned} \nabla\ C(\mathbf{w}) \quad &= E_{\mathbf{xyz}} \Big[\quad \Big(\nabla_{\mathbf{w}}\ \mathbf{q}(\mathbf{x},\mathbf{w}) \Big) \nabla_{\mathbf{q}}\ \Phi\ (\ \bar{\mathbf{q}}\ ,\ \mathbf{s}(\mathbf{y},\mathbf{w})\ ) \\ &\qquad + \Big(\nabla_{\mathbf{w}}\ \mathbf{s}(\mathbf{y},\mathbf{w}) \Big) \nabla_{\mathbf{s}}\ \Phi\ (\ \bar{\mathbf{q}}\ ,\ \mathbf{s}(\mathbf{y},\mathbf{w})\ ) \\ &\qquad + \Big(\nabla_{\mathbf{w}}\ \mathbf{h}(\mathbf{z},\mathbf{w}) \Big) \lambda \quad \Big] \qquad (2.13a) \\ &= \mathbf{0} \end{aligned}$$

$$E_{\mathbf{z}}\ \{\ \mathbf{h}(\mathbf{z},\mathbf{w})\ \} = \mathbf{0} \tag{2.13b}$$

L'équation (2-13a) représente la condition de minimalité ordinaire, à laquelle est ajoutée un terme représentant l'amélioration locale possible du respect de la contrainte. L'équation (2-13b) représente le respect de la contrainte.

On obtient alors l'algorithme stochastique de descente de gradient à convergence vers les contraintes pour la fonction de coût de forme sophistiquée (2-6):

$$\begin{aligned} \mathbf{w}_{n+1} = \ & \mathbf{w}_{n-1} \\ & - \Gamma_{1,n} \Big[\quad \Big(\nabla_{\mathbf{w}}\ \mathbf{q}^T(\mathbf{x}_n,\mathbf{w}_n) \Big) \nabla_{\mathbf{q}}\ \Phi\ \Big(\bar{\mathbf{q}}_n\ ,\ \mathbf{s}(\mathbf{y}_n,\mathbf{w}_n) \Big) \\ & \qquad + \Big(\nabla_{\mathbf{w}}\ \mathbf{s}^T(\mathbf{y}_n,\mathbf{w}_n) \Big) \nabla_{\mathbf{s}}\ \Phi\ \Big(\bar{\mathbf{q}}_n\ ,\ \mathbf{s}(\mathbf{y}_n,\mathbf{w}_n) \Big) \\ & \qquad + \Big(\nabla_{\mathbf{w}}\ \mathbf{h}^T(\mathbf{z}_n,\mathbf{w}_n) \Big) \lambda_n \quad \Big] \qquad (2.14a) \end{aligned}$$

$$\bar{\mathbf{q}}_{n+1} = \bar{\mathbf{q}}_n - \Gamma_{2,n}\ (\ \bar{\mathbf{q}}_n - \ \mathbf{q}(\ \mathbf{x}_n\ ,\ \mathbf{w}_n)\ ) \tag{2.14b}$$

$$\lambda_{n+1} = \lambda_n + \Gamma_{3,n}\ \mathbf{h}(\ \mathbf{z}_n\ ,\ \mathbf{w}_n) \tag{2.14c}$$

L'équation (2-14a) représente la descente de gradient proprement dite; elle à une forme proche de la condition de minimalité (2-13a). L'équation (2-14b) effectue l'ajustement de l'estimation de $\bar{\mathbf{q}}$. L'équation (2-14c) implante un accumulateur du non-respect de la contrainte.

En faisant l'hypothèse (2-8), on obtient directement l'algorithme stochastique de descente de gradient à convergence vers contrainte pour la fonction de coût de forme simple (2-1) :

$$\begin{aligned} \mathbf{w}_{n+1} = \ & \mathbf{w}_{n-1} \\ & - \Gamma_{1,n} \Big[\ \nabla_{\mathbf{w}} J(\mathbf{x}_n,\mathbf{w}_n)\ \Big) + \Big(\frac{d\ \mathbf{h}^T(\mathbf{z}_n,\mathbf{w}_n)}{d\ \mathbf{w}} \Big) \lambda_n \Big] \qquad (2.15a) \end{aligned}$$

$$\lambda_{n+1} = \lambda_n + \Gamma_{3,n}\, \mathbf{h}(\, \mathbf{z}_n\, ,\, \mathbf{w}_n) \qquad (2.15b)$$

On peut observer que l'algorithme est évolutif. Dans un premier temps, grâce à $\Gamma_{3,n}$ λ peut prendre des valeurs arbitrairement petites et l'algorithme se comporte alors comme une descente de gradient classique. Progressivement, au cours de l'apprentissage et en particulier quand les valeurs de $\mathbf{w}$ commencent à se stabiliser, λ va prendre de plus en plus d'importance. A la fin de l'apprentissage λ ne cesse de croître jusqu'à ce que $\mathbf{h}$ s'annule.

Cet algorithme est potentiellement très instable. Sa stabilisation exige que la contrainte soit parfaitement respectée Au besoin, l'algorithme fera tendre λ vers une valeur suffisamment grande pour sortir le jeu de paramètre d'un éventuel bassin d'attraction. Ce coefficient ne va se réduire que progressivement, et l'agitation engendrée dans cet intermède aura peut-être conduit le jeu de paramètre à une situation métastable ou fort éloignée des minima locaux.

Il convient également de signaler que l'algorithme (2-15) connaît une variante pour contraintes telles que les inégalités.

Une autre variante de l'algorithme (2-14), plus stable mais qui se contente du respect approximatif de la contrainte, consiste à ajouter à la fonction de coût (2-1) une pénalité croissant avec l'importance du non-respect de la contrainte. L'algorithme effectue donc un compromis entre la fonction de coût originelle et la pénalité de contrainte. La convergence ne s'effectue donc pas vers un jeu de paramètres respectant strictement la contrainte. Il reste toujours possible d'augmenter la valeur de la pénalité de l'algorithme (2-14) si la convergence semble se faire dans un irrespect de la contrainte par trop insupportable…

2.3 Apprentissage et adaptation

2.3.1 Deux problématiques proches

On distingue classiquement deux problématiques proches mais différentes:

- L'adaptation consiste à améliorer un système au cours de son utilisation. L'adaptation prend tout son sens et est nécessaire quand les données traitées - c'est-à-dire la distribution ($\mathbf{X}$, p) - évoluent avec le temps. Un exemple en est le filtrage adaptatif

du bruit électrique perturbant les télécommunications [1], [2].

- L'apprentissage consiste à optimiser complètement les paramètres d'un système. Ceci peut être fait en considérant un ensemble (dé)fini d'exemples représentatifs. Ainsi en est-il de la lecture automatique de caractères manuscrits.

Ces deux problématiques diffèrent. Par l'adaptation, on cherche à évaluer aussi vite que possible - en termes de temps de calcul et surtout de nombre d'exemples nouvellement présentés - l'évolution des conditions extérieures afin d'obtenir une configuration raisonnable. Par l'apprentissage, on cherche à obtenir une configuration optimisée sans forcément se focaliser sur la vitesse de convergence ou sur les problèmes de divergences spécifiques aux cas évolutifs. On peut alors utiliser un grand nombre de techniques d'estimation.

Cependant, ces deux problématiques restent proches. Il est possible d'initialiser un système adaptatif par un apprentissage. Ce sont souvent des versions presque identiques des mêmes algorithmes qui sont employées pour les deux cas [3], [4]. Quand on peut tirer aléatoirement un nombre indéfini d'exemples de la distribution, il peut être plus efficace d'utiliser des algorithmes d'apprentissage adaptatif que des algorithmes d'apprentissage global. Enfin, certains algorithmes d'apprentissages font apparaître une évaluation adaptative des conditions extérieures sur lesquels l'évolution des paramètres de l'apprentissage a une incidence: quand le système interagit avec l'extérieur d'une façon trop mal comprise, l'apprentissage ne peut se faire sans adaptativité. C'est le cas notamment des algorithmes de contrôle et d'asservissement.

L'algorithme de descente de gradient stochastique est un algorithme d'apprentissage fondamentalement adaptatif.

2.3.2 Algorithmes optimaux et quasi-optimaux

Afin d'utiliser au mieux les qualités adaptatives de la descente de gradient stochastique, il ne

---

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

2 **Macchi O.:** *Advances in adaptive filtering* - pp 41-57 in E. Biglieri and Prati G. (eds) Digital Communications, North Holland, Amsterdam (1986)

3 **Marcos S., Vignat C., Macchi O.:** *L'algorithme de rétropropagation du gradient en réseaux de neurones comparés aux alorithmes adaptatifs du traitement du signal* - Rapport Interne., Laboratoire des signaux et Systèmes, Gif sur Yvette (September 1989)

4 **Marcos S., Macchi O., Vignat C., Dreyfus G., Personnaz L., Roussel-Ragot P.:** *A unified framework for gradient algorithms used for filter adaptation and neural network training* - International journal of circuit theory and applications,Vol 20, 159-200 (1992)

suffit pas d'évoluer paisiblement dans le sens du gradient, il faut évoluer suffisamment rapidement pour suivre au plus près les évolutions de la distribution ($\mathbf{X}$, p). Dans cette optique se pose alors le problème de cette "optimalité" propre à l'adaptation, que l'on tente de résoudre par des méthodes du second degré.

2.3.2.1 Critère d'optimalité de l'adaptation

Pour juger de l'état de la convergence de l'algorithme, on serait tenté d'utiliser un critère de la forme:

$$V_n^2 = E \{ | \mathbf{w}_n - \mathbf{w}^* |^2 \} , \tag{2.16}$$

qui, bien évidemment, est non évaluable quand on manque d'information *a priori*.

On en est réduit à utiliser une *estimation* de $C(\mathbf{w})$:

$$\hat{C} (\mathbf{w}_n) = \frac{1}{n} \sum_{m=1}^{n} J (\mathbf{x}_m , \mathbf{w}_n) \tag{2.17}$$

Ce critère présente l'inconvénient majeur de réclamer une mémorisation de tous les $\mathbf{x}_m$, c'est-à-dire une mémoire indéfiniment extensible, ainsi que des calculs sans cesse plus longs. Par ailleurs, il tend à rendre le système insensible aux évolutions des conditions dans lesquelles les $\mathbf{x}_m$ sont engendrés. Il est raisonnable d'utiliser un critère encore moins précis, mais plus approprié pour le calcul adaptatif:

$$\hat{C}_N (\mathbf{w}_n) = \frac{1}{N+1} \sum_{m=n-N}^{n} J (\mathbf{x}_m , \mathbf{w}_n) \tag{2.18}$$

Ce critère consiste à estimer le comportement du système sur les $N+1$ derniers exemples avec le jeu de paramètres courant $\mathbf{w}_n$. Il peut être utilisé pour l'adaptation optimale ou pour l'apprentissage optimal (en terme de vitesse de convergence).

2.3.2.2 Apprentissage quasi-optimal

Étude générale

Chaque réévaluation des paramètres par descente de gradient stochastique s'écrit:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} J (\mathbf{x}_n, \mathbf{w}_n) , \tag{2.19}$$

où Γ est une matrice symétrique définie positive quelconque. La fonction de coût estimée (2-17) s'écrit alors:

$$\hat{C}(\mathbf{w}_n) = \frac{1}{n}\sum_{m=1}^{n} J\left(\mathbf{x}_m\ ,\ \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n,\mathbf{w}_n)\right) \tag{2.20}$$

L'optimalité de l'apprentissage va être obtenue en choisissant Γ_n de manière à minimiser la fonction de coût estimée. La condition d'extrémalité de cette fonctionnelle par rapport à Γ_n conduit à l'équation:

$$\mathbf{0} = \left[\frac{1}{n}\sum_{m=1}^{n} \nabla_{\mathbf{w}} J\left(\mathbf{x}_m\ ,\ \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n,\mathbf{w}_n)\right)\right] \tag{2.21}$$

Cette condition n'est pas, dans le cas général, évaluable; cependant en utilisant la matrice jacobienne de la fonction de coût J, on peut l'approcher linéairement dans le cas où Γ_n est petit:

$$\mathbf{0} = \sum_{m=1}^{n}\left[\nabla_{\mathbf{w}} J(\mathbf{x}_m,\mathbf{w}_n) - \nabla_{\mathbf{w}}^2 J(\mathbf{x}_m,\mathbf{w}_n)\,\Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n,\mathbf{w}_n)\right] + \mathcal{O}_{\Gamma\to 0}(|\Gamma|^2). \tag{2.22}$$

Nous sommes dans le contexte d'algorithmes *quasi-optimaux*. (••• références). Si les précédentes étapes de l'apprentissage ont été bien traitées, on peut supposer que:

$$\sum_{m=1}^{n-1} \nabla_{\mathbf{w}} J(\mathbf{x}_m,\mathbf{w}_n) = \mathbf{0} + \mathcal{O}_{\Gamma\to 0}(|\Gamma|^2), \tag{2.23}$$

ce qui constitue une sorte d'hypothèse de récurrence. Elle permet de simplifier l'expression (2-22) qui devient:

$$\Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n,\mathbf{w}_n) = \left(\sum_{m=1}^{n} \nabla_{\mathbf{w}}^2 J(\mathbf{x}_m,\mathbf{w}_n)\right)^{-1} \nabla_{\mathbf{w}} J(\mathbf{x}_n,\mathbf{w}_n) + \mathcal{O}_{\Gamma\to 0}(|\Gamma|^2). \tag{2.24}$$

Par la suite, nous noterons $\mathbf{K}_n$ l'inverse de la matrice jacobienne de la fonction de coût estimée $\hat{C}$:

$$\mathbf{K}_n \triangleq \left(\sum_{m=1}^{n} \nabla_{\mathbf{w}}^2 J(\mathbf{x}_m,\mathbf{w}_n)\right)^{-1} \tag{2-25}$$

L'ensemble des solutions possibles pour (2-21) est donc de la forme:

$$\Gamma_n = \Gamma_{n,0} + \Gamma_{n,v}\ ,$$

où $\Gamma_{n,0}$ est une solution particulière et où $\Gamma_{n,v}$ est une matrice symétrique (donc diagonalisable dans un repère orthonormé) annulant $\nabla_{\mathbf{W}} J(\mathbf{x}_n,\mathbf{w}_n)$ et telle que $\Gamma_{n,0} + \Gamma_{n,v}$ reste positive.

Une solution particulière simple est obtenue quand $\Gamma_{n,0}$ est diagonale. En notant $\mathbf{A}$ l'opérateur qui transforme canoniquement un vecteur en une matrice diagonale:

$$\Gamma_{n,0} = \mathbf{A}\, \gamma_n \tag{2-26}$$

et en utilisant les propriétés triviales de $\mathbf{A}$, on déduit de (2-24):

$$\gamma_n = (\mathbf{A}\, \nabla_{\mathbf{w}} J(\mathbf{x}_n,\mathbf{w}_n))^{-1} \left(\sum_{m=1}^{n} \nabla^2_{\mathbf{w}} J(\mathbf{x}_m,\mathbf{w}_n) \right)^{-1} \nabla_{\mathbf{W}} J(\mathbf{x}_n,\mathbf{w}_n) \; . \tag{2-27}$$

L'écriture de l'algorithme d'apprentissage quasi-optimal se simplifie alors pour devenir:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \left(\sum_{m=1}^{n} \nabla^2_{\mathbf{w}} J(\mathbf{x}_m,\mathbf{w}_n) \right)^{-1} \nabla_{\mathbf{w}} J\,(\,\mathbf{x}_n,\, \mathbf{w}_n\,) \; . \tag{2-28}$$

Cet algorithme est séduisant; néanmoins sous la forme (2-28), il présente en général plusieurs inconvénients importants:

- il n'est que quasi-optimal,
- il fait intervenir une inversion de matrice qui est informatiquement insupportable pour les systèmes ayant de nombreux paramètres (de l'ordre de p^3 calculs, ou p est le nombre de paramètres).
- afin de calculer $\mathbf{K}_n$ (l'inverse de la matrice jacobienne de la fonction de coût estimée $\hat{C}$) il nécessite des calculs sans cesse plus lourds et une mémoire plus grande à chaque exemple.
- il tend à devenir insensible aux changements des conditions dans lesquelles les exemples $\mathbf{x}_n$ sont engendrés.

Les deux derniers inconvénients disparaissent dès que la fonction de coût $\hat{C}_N$ est utilisée à la place de la fonction $\hat{C}$.

On peut résoudre le troisième inconvénient en réduisant la matrice $\mathbf{K}_n$ à sa diagonale. Il s'agit d'une approximation un peu sauvage mais, en pratique, elle donne d'excellents résultats. Elle permet de trouver un intermédiaire entre les méthodes quasi-newtoniennes et les simples descentes de gradient dont le pas d'apprentissage est isotrope. Une méthode sophistiquée utilisant ce principe a été mise au point par Marcquardt suite à une suggestion de Levenberg

[1].

Dans le cas linéaire (voir ci-dessous) les trois derniers inconvénients sont circonvenus.

Cas où le gradient est linéaire

Plusieurs propriétés du cas où $\nabla_{\mathbf{W}} J(\mathbf{x},\mathbf{w})$ est linéaire par rapport à $\mathbf{w}$ méritent d'être mentionnées: l'optimalité, la réévaluation adaptative et la proximité de l'optimum de la fonction de coût total.

Dans le cas linéaire, l'algorithme d'apprentissage (2-28) est vraiment optimal. Par exemple, pour:

$$J(\mathbf{x},\mathbf{w}) = \frac{1}{2} (\mathbf{y} - \mathbf{w}^T \varphi(\tilde{\mathbf{x}}))^2 \qquad (2\text{-}29)$$

où φ est une fonction vectorielle connue et où $\mathbf{x} = (\tilde{\mathbf{x}}, \mathbf{y})$, l'inverse de matrice jacobienne s'écrit:

$$\mathbf{K}_n = \left(\sum_{m=1}^{n} (\mathbf{A}\varphi(\tilde{\mathbf{x}}_m))^2 \right)^{-1} \qquad (2\text{-}30)$$

Cette matrice est diagonale et ses coefficients rendent compte de la courbure de l'hyper-surface représentant la fonction de coût quadratique.

Pour calculer $\mathbf{K}_n$ en conservant le caractère adaptatif de la descente de gradient stochastique, il faut trouver pour l'expression (2-30) une formule de réévaluation. Dans le cas linéaire cette formule existe. En remarquant que:

$$\mathbf{K}_{n+1}^{-1} = \mathbf{K}_n^{-1} + \varphi(\tilde{\mathbf{x}}_n)\, \varphi^T(\tilde{\mathbf{x}}_n) \qquad (2\text{-}31)$$

et en utilisant la formule d'inversion:

$$(A + BCB^T)^{-1} = A^{-1} - A^{-1}B(C^{-1} + B^T A^{-1} B)^{-1} B^T A \,, \qquad (2\text{-}32)$$

on déduit:

$$\mathbf{K}_{n+1} = \mathbf{K}_n - \frac{[\, \mathbf{K}_n \varphi(\tilde{\mathbf{x}}_n)\,] \,.\, [\, \mathbf{K}_n \varphi(\tilde{\mathbf{x}}_n)\,]^T}{1 + \varphi(\tilde{\mathbf{x}}_n)^T [\, \mathbf{K}_n \varphi(\tilde{\mathbf{x}}_n)\,]} \,. \qquad (2\text{-}32)$$

---

1 **Marcquardt D. W.:** in Journal of Society for Industry Applied Mathematics, vol 11, pp 431-441 (1963)

Cette formule de réévaluation permet à l'algorithme d'apprentissage (2-28) de garder son caractère adaptatif.

Dans le cas linéaire, la fonction de coût est quadratique positive. En général, elle possède un unique minimum que l'on note $\mathbf{w}^*$. Par définition, l'algorithme optimal (2-28) minimise à chaque étape $\hat{C}$; il minimise aussi l'espérance (portant sur les représentants $\mathbf{x}_n$ de la distribution $(\mathbf{X}, p)$) de la distance moyenne carrée séparant la configuration de paramètres courante $\mathbf{w}_n$ et la configuration optimale $\mathbf{w}^*$.

Enfin, il existe une variante de la formule d'évaluation (2-32) quand la fonction de coût $\hat{C}_N$ est utilisée à la place de la fonction $\hat{C}$, ce qui fournit un algorithme adaptatif optimal pour traiter le cas de conditions évolutives.

2.4 Convergence de l'algorithme de descente de gradient stochastique

2.4.1 Généralités

Pour les systèmes d'apprentissage, la propriété de convergence est fondamentale. En revanche, elle n'est pas toujours fondamentale pour les systèmes adaptatifs, qui lui préfèrent certaines propriétés de stabilité et surtout une grande promptitude d'adaptation. Il existe cependant toute une problématique de la convergence développée pour le comportement d'algorithmes adaptatifs placés en situation évolutive [1].

Les lignes qui suivent sont consacrés à la convergence d'algorithmes d'apprentissage par descente de gradient stochastique.

Il existe des théorèmes de convergence asymptotique pour les processus stochastique [2], [3],

---

1 **Macchi O., Eweda E.:** *Second order convergecne analysis of stochastic adaptive linear filtering* - IEEE ASSP Mag. pp 12-31 (1990)

2 **Fisk D.L.:** *Quasi-martingales* - Tans. Amer. Math. Soc, nb 120, pp 359-388 (1965)

3 **Neveu J.:** *Martingales à temps discret* - Masson (1972)

[1], [2]. De tels théorèmes appartiennent au domaine de l'analyse numérique. Ils reposent sur l'analyse de phénomènes stochastiques (probabilistes). Des progrès relativement récents, notamment par l'introduction des quasi martingales, facilitent grandement leur démonstration.

Dans le cas de la descente de gradient stochastique, ces théorèmes reposent sur des conditions de régularité de la fonction de coût local J ou globale C et de décroissance des valeurs propres (positives) de la matrice symétrique définie positive Γ_n, qui autorisent une accumulation arbitrairement importante des variations du jeu de paramètres et une lenteur arbitraire de l'évolution du jeu de paramètres.

Il semblerait que les démonstrations de tels théorèmes ont été publiées depuis longtemps dans la littérature spécialisée des processus d'apprentissage adaptatif, notamment dans les pays du pacte de Varsovie [3], [4]. Nous n'avons malheureusement pas pu les retrouver.

Nous n'avons pas connaissance de théorème aux hypothèses minimales. Le théorème que nous avons démontré et que nous proposons ici est à notre connaissance le moins contraignant qui ait été publié; il constitue une version un peu plus générale du théorème proposé récemment par L. Bottou [5].

2.4.2 Quasi-Martingales

i Généralités

Pour dire les choses grossièrement, les quasi-martingales sont partiellement aux processus stochastiques ce que les suites à accroissement fini (resp. séries absolument convergentes) sont pleinement aux suites (resp. séries). Elles constituent de puissants outils pour montrer la convergence de processus stochastiques (vers une variable aléatoire).

---

1 **Metivier M.:** *Martingales et convergence presque sure d'algorithmes stochatiques, in "Outils et modèles mathématiques pour l'automatique, lanalyse de système et le traitement du signal"* - Edition du CNRS, vol 1, partie 3, "Techniques probabilistes en automatique et traitement du signal", pp 529-552 (1981)

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

3 **Devyaterikev, Kaplinsky, Tsypkin:** *On the convergence of the algorithms of learning* - Automat. i Telemeh. n° 10 (1969)

4 **Braverman E.M., Litvakov B.M.:** Convergence of algorithms of learning and adaptation - congress IFAC 4th, Warsawa, (1969)

5 **Bottou L.:** *Une approche théorique de l'apprentissage connexionniste; Application à la reconnaissance de la parole* - Thèse de doctorat, §3.3.2, pp 70-74 Univerité Paris XI, Orsay (1991)

ii Notations

On considère des processus $\mathbf{h}_{n\geq 0}$ à valeur dans un espace complet, par exemple $\mathbb{R}^p$. On note $\mathbf{H}_0^n$ la suite des $n+1$ premières valeurs du processus.

On note $E_{\mathbf{h}_{n+1}}\{ f(\mathbf{h}_{n+1}) \mid \mathbf{H}_0^n\}$ l'espérance de la fonction f portant sur la variable aléatoire $\mathbf{h}_{n+1}$ sachant $\mathbf{H}_0^n$, c'est à dire l'espérance d'une fonction du futur immédiat du processus sachant son passé.

iii Définition

On appelle quasi-martingale tout processus stochastique $\mathbf{h}_{n\geq 0}$ vérifiant:

$$\sum_n E_{\mathbf{H}_0^n} | E_{\mathbf{h}_{n+1}}\{ \mathbf{h}_{n+1} - \mathbf{h}_n \mid \mathbf{H}_0^{n-1}\} | \leq \infty \qquad (2\text{-}33)$$

En d'autres termes, une quasi-martingale est un processus dont l'espérance de variation moyenne est finie. Néanmoins, cela ne suffit pas pour pouvoir affirmer que le processus converge.

Ainsi, le processus $g_{n+1} = g_n \pm (n+1)^n$ (avec équiprobabilité des deux événements) est une quasi-martingale dont l'espérance de variation moyenne est à chaque instant nulle (c'est même une martingale); pourtant ce processus diverge toujours.

iv Théorème

Une quasi-martingale vérifiant la condition:

$$\operatorname{Sup}_n E_{\mathbf{h}_{n+1}}\{ | \mathbf{h}_{n+1} | \mid \mathbf{H}_0^n\} \leq \infty , \qquad (2\text{-}34)$$

converge presque sûrement, c'est à dire qu'elle vérifie le critère de convergence:

$$\forall\, \eta, \varepsilon, \quad \exists\, n_0, \quad \forall\, n \geq n_0, k \geq 0$$

$$p(| \mathbf{h}_{n+k} - \mathbf{h}_n | < \varepsilon) \quad > 1 - \eta . \qquad (2\text{-}35)$$

La démonstration de ce théorème n'est pas simple. Le lecteur travailleur pourra la trouver dans

différents ouvrages [1], [2], [3].

v Corollaire

Un processus à valeur dans $\mathbb{R}^+$ tel que:

$$\sum_{t} E_{H_0^n}\{ \max(0 , E_{h_{n+1}}\{ h_{n+1} - h_n \mid H_0^n\} \} \leq \infty \text{ et}$$

$$\sup_n E_{h_{n+1}}\{ h_{n+1} \mid H_0^n\} \leq \infty , \qquad (2\text{-}36)$$

converge, car un tel processus est une quasi-martingale et vérifie la condition (2-34).

Ce corollaire correspond bien à notre besoin, car nous cherchons à analyser des fonctions de coûts généralement minorées (on peut donc se ramener à des fonctions de coûts positives) et pour lesquelles on se fait fort de montrer que l'espoir de croissance est le plus faible possible…

2.4.3 Un théorème de convergence de l'algorithme de descente de gradient stochastique pour un pas d'apprentissage anisotrope

Nous proposons une étude dont le plan est donné en (i), menant à un théorème de convergence énoncé en (ix) et son corollaire en (x). Les hypothèses sont détaillées et discutées en (iii). La démonstration proprement dite est composée de (iv), (v), (vi), (vii) et (viii). Les parties (iv) et (v) reprennent la première partie du théorème de Bottou en les généralisant au cas du pas d'apprentissage anisotrope. Les parties (vi), (vii) et (viii), particulièrement délicates, sont entièrement originales.

i Plan de l'étude

i Plan de l'étude

ii Généralités

iii Hypothèses commentées

iv Les variations algébriques de la fonction de coût total sont majorables en fonction d'un

---

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

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

3 **Metivier M.:** *Martingales et convergence presque sure d'algorithmes stochatiques, in "Outils et modèles mathématiques pour l'automatique, lanalyse de système et le traitement du signal"* - Edition du CNRS, vol 1, partie 3, "Techniques probabilistes en automatique et traitement du signal", pp 529-552 (1981)

majorant de sa dérivée seconde…

v … la fonction de coût total est donc une quasi-martingale convergente

vi La majoration en moyenne de $|\nabla_{\mathbf{w}} J(\mathbf{w},\mathbf{x})|$ et la réduction du pas d'apprentissage engendrent le ralentissement probable de l'évolution des paramètres.

vii La D''lipschitziennité du gradient et le ralentissement probable de l'évolution des paramètres engendrent un allongement probable de la durée de validité du gradient approché.

viii La convergence presque sûre de $C(\mathbf{w})$, l'allongement probable de la durée de validité du gradient approché et la sommation arbitrairement élevée du pas d'apprentissage impliquent la convergence presque sûre de $|\nabla_{\mathbf{w}} C(\mathbf{w})|$ vers $\mathbf{0}$.

ix Théorème

x Corollaire

i Généralités

Nous proposons ici une variante généralisant le théorème proposé par L. Bottou [1] (dont nous reprenons et adaptons la première partie de la démonstration aux (iv) et (v). Nous autorisons les pas d'apprentissage anisotropes. De plus, nous n'exigeons pas que la fonction de coût globale soit trois fois dérivable: nous exigeons simplement que son gradient soit lipschitzien, ce qui est vérifié dés que la fonction de coût globale est deux fois dérivable.

L'apprentissage par descente de gradient stochastique découle de la règle (2-5) qui s'écrit:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n) \ , \qquad (2\text{-}37)$$

où Γ_n est une matrice symétrique définie positive définissant le pas d'apprentissage anisotrope et où $\mathbf{x}_n$ est l'exemple utilisé à la $n^{ième}$ itération pour l'apprentissage. On suppose que $\mathbf{x}_n$ est tiré aléatoirement de la distribution $(\mathbf{X} , p)$ étudiée.

On cherche à montrer que le processus aléatoire des paramètres $\mathbf{w}_t$ - découlant de l'algorithme d'apprentissage et du processus aléatoire de sélection des exemples au sein de la distribution $(\mathbf{X} , p)$ - converge, c'est à dire que les paramètres vont presque sûrement converger vers une configuration stable. On attend de cette configuration qu'elle constitue un point singulier de la fonction de coût total, si possible un minimum local.

---

1 **Bottou L.:** *Une approche théorique de l'apprentissage connexionniste; Application à la reconnaissance de la parole* - Thèse de doctorat, §3.3.2, pp 70-74 Univerité Paris XI, Orsay (1991)

ii Hypothèses commentées

Nous allons utiliser (cf. § iii) un certain nombre d'hypothèses. Nous les présentons ici avec quelques commentaires qui éclaireront peut-être le lecteur sur l'esprit de la démonstration.

Hypothèse 1: minoration de la fonction de coût total C:

L'hypothèse:

$$C(\mathbf{w}) \geq C_{min} \tag{2-38}$$

permet "d'arrêter" la descente de gradient qui autrement pourrait s'éterniser dans les profondeurs d'un puits de potentiel infini.

Cette hypothèse est vérifiée en particulier quand la fonction de coût local est minorée.

Hypothèse 2: symétrie définie positive et indépendance du pas d'apprentissage anisotrope

$$\Gamma_n \text{ est une matrice symétrique définie positive, indépendante du processus stochastique } \mathbf{x}_n, \tag{2-39}$$

L'hypothèse 2 permet de qualifier les valeurs propres de Γ_n et Γ_n^2. On définit ε_n^- et ε_n^+ les valeurs propres extrêmes de Γ_n :

$$\mathbb{R}^{+*} \supset [\ \varepsilon_n^-\ ,\ \varepsilon_n^+\ ] \supset \text{Valeurs Propres}\ (\ \Gamma_n\ )\ . \tag{2-40}$$

On sait que toute matrice symétrique $\mathbf{M}$ à coefficients dans $\mathbb{R}$ est diagonalisable dans un repère orthonormé. Son carré $\mathbf{M}^2$ est également symétrique, diagonalisable dans le même repère, et ses valeurs propres sont les carrés des valeurs propres de M. On peut ainsi qualifier les valeurs propres de Γ_n^2:

$$\mathbb{R}^{+*} \supset [\ \varepsilon_n^{-2}\ ,\ \varepsilon_n^{+2}\ ] \supset \text{Valeurs Propres}\ (\ \Gamma_n^2\ )\ , \tag{2-41}$$

on en déduit, pour tout vecteur $\mathbf{y}$:

$$\mathbf{y}\ \Gamma_n^2\ \mathbf{y} \in [\ \varepsilon_n^{-2}\ |\mathbf{y}|^2,\ \varepsilon_n^{+2}\ |\mathbf{y}|^2\ ]\ , \tag{2-42}$$

Quant à l'indépendance de Γ_n et de $\mathbf{x}_n$, c'est une condition moins forte qu'il n'y paraît. En effet Γ_n peut très bien dépendre de $\mathbf{x}_1 \ldots \mathbf{x}_{n-1}$, ce qui autorise des variantes de la plupart des algorithmes du second degré.

Hypothèse 3: sommabilité des carrés des valeurs propres maximales du pas d'apprentissage

anisotrope

…c'est à dire:

$$\sum_{n=0}^{\infty} \varepsilon_n^{+2} < \infty . \qquad (2\text{-}43)$$

Avec quelques hypothèses portant sur la régularité des fonctions de coût, ceci permet de garantir que les accroissements positifs de la fonction de coût sont bornés.

Hypothèse 4: non-sommabilité des valeurs propres minimales du pas d'apprentissage anisotrope et solidarité avec les valeur propres maximales

La première partie de cette hypothèse s'écrit simplement:

$$\sum_{n=0}^{\infty} \varepsilon_n^{-} = \infty \qquad (2\text{-}44)$$

elle permet de garantir que le chemin susceptible d'être parcouru dans l'espace des paramètres, est aussi long que nécessaire pour que le gradient tende vers **0**. Sans cela, la convergence pourrait se faire vers une configuration de paramètres pour laquelle le gradient serait non-nul…

Répondant au même objectif, la deuxième partie de cette hypothèse porte sur la solidarité entre valeurs propres minimales et maximales, afin que certaines directions (en matière de variation de paramètres) ne soient pas injustement défavorisées:

$$\forall\ \varepsilon''>0,\ \exists\ \varepsilon,\ n_0'',\ \forall\ n \geq n_0'',\ \exists\ k: \quad \begin{cases} \sum_{n}^{n+k} \varepsilon_n^{+} \leq \varepsilon'' \\ \dfrac{\varepsilon}{\varepsilon''} \leq \sum_{n}^{n+k} \varepsilon_n^{-} \end{cases} \qquad (2\text{-}45)$$

Cette hypothèse est vérifié dès que le rapport entre les deux valeurs propres extrêmes est borné, i.e. $\exists\ \varepsilon_0\ , n_0 : \forall\ n \geq n_0 : \dfrac{\varepsilon_n^{-}}{\varepsilon_n^{+}} > \varepsilon_0$.

En fait, comme on va le voir, on mettra en évidence (2-77) une formule du type:

$$\forall\ \varepsilon, \exists\ n_0, \forall\ n \geq n_0, k: \quad 0 \leq \varphi_n \leq \frac{\varepsilon}{(A_{n+k} - A_n)} + (B_{n+k} - B_n) \qquad (2\text{-}46)$$

et on cherchera à obtenir:

$$\forall\, \varepsilon', \exists\, n_0', \forall\, n \geq n_0' : \quad \varphi_n \leq \varepsilon' \tag{2-47}$$

Si on veut démontrer que (2-46) implique (2-47) pour tout φ_n en respectant les contraintes:

$$\begin{cases} B_0 = A_0 = 0 \\ (B_{n+1} - B_n) \geq (A_{n+1} - A_n) \geq 0 \\ (B_{n+1} - B_n) \underset{\infty}{\rightarrow} 0 \end{cases} \tag{2-48}$$

on fait apparaître une condition nécessaire et suffisante sur A_n et B_n:

$$\forall\, \varepsilon'' > 0, \exists\, \varepsilon, n_0, \forall\, n \geq n_0, \exists\, k: \quad \begin{cases} B_{n+k} - B_n \leq \varepsilon'' \\ \dfrac{\varepsilon}{\varepsilon''} \leq A_{n+k} - A_n \end{cases} \tag{2-49}$$

Cette formulation est quelque peu abstraite. Afin de clarifier l'idée, on peut fournir une condition nécessaire et une condition suffisante plus tangibles:

Condition nécessaire:

$$\begin{cases} A_n \underset{\infty}{\rightarrow} \infty \\ \overline{\lim} \dfrac{B_n}{A_n} < \infty \end{cases} \tag{2-50}$$

Condition suffisante:

$$\begin{cases} A_n \underset{\infty}{\rightarrow} \infty \\ \exists\, n_0, k_0 : \underset{n \geq n_0, k \geq k_0}{\overline{\lim}} \dfrac{B_{n+k} - B_n}{A_{n+k} - A_n} < \infty \end{cases} \tag{2-51}$$

Hypothèse 5: le gradient **global** est lipschitzien

…c'est à dire:

$$\boxed{\exists\, D'' : \left| \nabla_{\mathbf{w}}\, C(\mathbf{w} + \mathbf{v}) - \nabla_{\mathbf{w}}\, C(\mathbf{w}) \right| \leq D'' \left| \mathbf{v} \right| .} \tag{2-52}$$

Cette hypothèse permet de majorer les accroissements positifs de la fonction de coût total. Elle est vérifiée dés que la fonction de coût total est deux fois dérivable.

Hypothèse 6: le gradient global est égal à l'espérance du gradient local

…c'est à dire:

$$\boxed{E_{\mathbf{x}}\{ \nabla_{\mathbf{w}}\, J(\mathbf{x}, \mathbf{w}) \} = \nabla_{\mathbf{w}}\, C(\mathbf{w}) .} \tag{2-53}$$

Cette hypothèse est vérifiée dés que la fonction de coût local $J(\mathbf{x}, \mathbf{w})$ est suffisamment régulière, notamment quand elle constitue une famille sommable (théorème de convergence dominée de l'intégrale d'une fonction paramétrique).

Hypothèse 7: le carré du gradient local varie en moyenne au plus linéairement avec la fonction de coût total

…c'est à dire:

$$E_{\mathbf{X}}\{ | \nabla_{\mathbf{W}} J(\mathbf{x}, \mathbf{w}) |^2 \} \leq (a\ C(\mathbf{w}) + b) . \qquad (2\text{-}54)$$

On peut alors remarquer que:

$$E_{\mathbf{X}}\{ | \nabla_{\mathbf{W}} J(\mathbf{x}, \mathbf{w}) | \} \leq E_{\mathbf{X}}\{ \max(1, | \nabla_{\mathbf{W}} J(\mathbf{x}, \mathbf{w}) |^2) \} \leq (a\ C(\mathbf{w})+1+b) \qquad (2\text{-}55)$$

D'où le corollaire de (2-54):

Hypothèse 7 (corollaire): le gradient local varie en moyenne au plus linéairement avec la fonction de coût total

$$E_{\mathbf{X}}\{ | \nabla_{\mathbf{W}} J(\mathbf{x}, \mathbf{w}) | \} \leq (a\ C(\mathbf{w}) + b') . \qquad (2\text{-}56)$$

Conclusion sur les hypothèses

iv Les variations algébriques de la fonction de coût total sont majorables en fonction d'un majorant du taux de variation du gradient…

Dans un premier temps, nous allons simplement montrer que la fonction de coût total converge. Cette fonction (2-1) s'écrit:

$$C(\mathbf{w}) \stackrel{\Delta}{=} E_{\mathbf{x}}\{ J(\mathbf{x},\mathbf{w}) \} . \qquad (2\text{-}56)$$

D'après (2-56) et l'hypothèse 5 (2-52), on a:

$$\begin{aligned} C(\mathbf{w}_{n+1}) &= C(\mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n)) , \\ &\leq C(\mathbf{w}_n) - (\Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n)) \bullet \nabla_{\mathbf{w}} C(\mathbf{w}_n) \\ &\quad + \tfrac{1}{2} D'' | \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n) |^2 . \end{aligned} \qquad (2\text{-}57)$$

L'hypothèse 1 (2-38) permet de définir:

$$h_n \stackrel{\Delta}{=} C(\mathbf{w}_n) - \min_{\mathbf{W}} C(\mathbf{w}) \stackrel{\Delta}{=} C(\mathbf{w}_n) - C_{min} \geq 0 \ , \tag{2-58}$$

d'où:

$$\begin{aligned} h_{n+1} - h_n = & - (\Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n)) \bullet \nabla_{\mathbf{w}} C(\mathbf{w}_n) \\ & + \frac{1}{2} D'' (| \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n) |^2) \end{aligned} \tag{2-59}$$

En tenant compte de l'hypothèse d'indépendance 2 (2-39) on déduit que:

$$\begin{aligned} E_{\mathbf{x}_n}\{ h_{n+1} - h_n \mid \mathbf{X}_0^{n-1}\} \quad \leq & - (\Gamma_n \nabla_{\mathbf{w}} C(\mathbf{w}_n)) \bullet \nabla_{\mathbf{w}} C(\mathbf{w}_n) \\ & + \frac{1}{2} D'' E_{\mathbf{x}_n}\{ | \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n) |^2 \mid \mathbf{X}_0^{n-1}\} \end{aligned} \tag{2-60}$$

En notant $Q_{\mathbf{M}}$ la forme quadratique découlant canoniquement d'une matrice symétrique $\mathbf{M}$, et en tenant compte de la symétrie de Γ_n on obtient:

$\exists\ n_0, K: \forall\ n \geq n_0$:

$$\begin{aligned} E_{\mathbf{x}_n}\{ h_{n+1} - h_n \mid \mathbf{X}_0^{n-1}\} \quad \leq & - Q_{\Gamma_n} (\nabla_{\mathbf{w}} C(\mathbf{w}_n)) \\ & + \frac{1}{2} D'' E_{\mathbf{x}_n}\{ Q_{\Gamma_n^2} (\nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n)) \mid \mathbf{X}_0^{n-1}\} \ . \end{aligned} \tag{2-61}$$

selon l'hypothèse d'indépendance 2 (2-39), on obtient:

$$\boxed{\begin{aligned} E_{\mathbf{x}_n}\{ h_{n+1} - h_n \mid \mathbf{X}_0^{n-1}\} \leq & - \varepsilon_n^- \ | \nabla_{\mathbf{w}} C(\mathbf{w}_n) |^2 \\ & + \frac{1}{2} D'' \varepsilon_n^{+2} E_{\mathbf{x}_n}\{ | \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n) |^2 \mid \mathbf{X}_0^{n-1} \} \end{aligned}} \tag{2-62}$$

Cette majoration a le défaut de laisser apparaître le gradient de la fonction de coût local, qui est un terme difficile à quantifier en raison de son aspect stochastique.

v la fonction de coût total est une quasi-martingale convergente

On est amené à utiliser l'hypothèse 7 (2-54), une majoration quelque peu grossière qui permet de se ramener à l'étude de termes moins aléatoires:

$$E_{\mathbf{x}_n}\{ h_{n+1} - h_n \mid \mathbf{X}_0^{n-1}\} \quad \leq \frac{1}{2} D'' \varepsilon_n^{+2} (\alpha + \beta C(\mathbf{w}_n))$$

$$E_{\mathbf{x}_n}\{ h_{n+1} \mid \mathbf{X}_0^{n-1}\} - (1+\beta\varepsilon_n^{+2}) h_n \leq \frac{1}{2} D'' \varepsilon_n^{+2} (\alpha+\beta C_{min})$$

$$\frac{E_{\mathbf{x}_n}\{\, h_{n+1}\,\}}{\prod_{i\le n}(1+\beta\varepsilon_i^{+2})} - \frac{h_n}{\prod_{i\le n-1}(1+\beta\varepsilon_i^{+2})} \le \frac{\frac{1}{2}D''\,\varepsilon_n^{+2}\,(\alpha+\beta C_{min})}{\prod_{i\le n}(1+\beta\varepsilon_i^{+2})} \tag{2-63}$$

En posant:

$$h'_n = \frac{h_n}{\prod_{i\le n}(1+\beta\varepsilon_i^{+2})} \tag{2-64}$$

on met alors en évidence un processus positif dont l'espérance d'accroissement positif est finie: d'après le corollaire (2-36), un tel processus est une quasi-martingale convergente. En effet:

$$E_{\mathbf{x}_n}\{\, h'_{n+1} \mid \mathbf{X}_0^{n-1}\} - h'_n \le \frac{\frac{1}{2}D''\,\varepsilon_n^{+2}\,(\alpha+\beta C_{min})}{\prod_{i\le n}(1+\beta\varepsilon_i^{+2})} \tag{2-65}$$

Afin de pouvoir affirmer que h' est une quasi-martingale et qu'on peut en déduire que h est également une quasi-martingale, on est amené à poser l'hypothèse 3 (2-43) sur ε_n^+. On obtient alors immédiatement:

$$E_{\mathbf{x}_n}\{\, h'_{n+1} - h'_n \mid \mathbf{X}_0^{n-1}\} \le K\,\varepsilon_n^{+2} \tag{2-66}$$

dont on déduit que h' est une quasi-martingale convergente, puis:

$$\begin{aligned} h_{n+1} - h_n &= (h'_{n+1}\,(1+\beta\,\varepsilon_n^{+2}) - h'_n)\prod_{i\le n}(1+\beta\,\varepsilon_i^{+2}) \\ &\le \mathrm{Max}(\,0\,,\, h'_{n+1}\,(1+\beta\,\varepsilon_n^{+2}) - h'_n)\;K' \\ &\le \mathrm{Max}(\,0,\,(h'_{n+1} - h'_n)\,)\,K' + h'_{n+1}K'\,\beta\,\varepsilon_n^{+2} \end{aligned} \tag{2-67}$$

Comme h' est une quasi martingale convergente, l'hypothèse 3 (2-43) nous permet une fois encore de conclure que le majorant de (2-67) est sommable. D'après le corollaire (2-36), cela signifie que le processus h est une quasi-martingale convergente.

Le théorème de convergence presque sûre des quasi-martingales nous permet de conclure que le coût total $C(\mathbf{w}_n)$ converge presque sûrement au sens des processus stochastiques, c'est-à-dire qu'il converge vers une distribution (et non vers une valeur).

Nous allons à présent montrer que le gradient de la fonction de coût total converge presque sûrement vers 0.

v i La majoration en moyenne de $|\nabla_w J(w,x)|$ et la réduction du pas d'apprentissage engendrent le ralentissement probable de l'évolution des paramètres

L'algorithme d'apprentissage s'écrit:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n) ,$$

D'où:

$$E_{\mathbf{x}_n}\{ | \mathbf{w}_{n+1} - \mathbf{w}_n | | \mathbf{X}_0^{n-1} \}$$

$$\leq \varepsilon_n^+ E_{\mathbf{x}_n}\{ | \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n) | | \mathbf{X}_0^{n-1}\} ,$$

$$\leq \varepsilon_n^+ (\alpha' + \beta' C(\mathbf{w}_n)), \qquad (2\text{-}68)$$

On peut remarquer à cette occasion que la convergence presque sûre de $C(\mathbf{w}_n)$ permettrait de conclure rapidement que $\mathbf{w}_n$ est une quasi-martingale presque sûrement convergente. Par ailleurs, on déduit de (2-68):

$$E_{\mathbf{X}_n^{n+k-1}}\{ | \mathbf{w}_{n+k} - \mathbf{w}_n | | \mathbf{X}_0^{n-1} \}$$

$$\leq \alpha' \sum_{i=0}^{k-1} \varepsilon_{n+i}^+ + \beta' \sum_{i=0}^{k-1} (\varepsilon_{n+i}^+ C(\mathbf{w}_{n+i})), \qquad (2\text{-}69)$$

Or, la convergence presque sûre de $C(\mathbf{w}_n)$ implique:

$$\forall \eta, \exists K, n_0, \forall n \geq n_0: \quad P_{\mathbf{X}_0^{n-1}} (| C(\mathbf{w}_n) | < K) \geq 1 - \eta \qquad (2\text{-}70)$$

On déduit de (2-69) et (2-70) :

$$\boxed{\begin{array}{l} \forall \eta, \exists K, n_0, \forall n \geq n_0, k \geq 0: \\ P_{\mathbf{X}_0^{n-1}} \left(E_{\mathbf{X}_n^{n+k-1}} | \mathbf{w}_{n+k} - \mathbf{w}_n | < (\alpha' + K \beta') \sum_{i=0}^{k-1} \varepsilon_{n+i}^+ \right) \geq 1 - \eta \end{array}} \qquad (2\text{-}71)$$

Dans la mesure ou ε_n^+ tend vers 0, cette expression traduit le ralentissement probable de l'évolution des paramètres.

vii La D''-lipschitziennité du gradient et le ralentissement probable de l'évolution des paramètres engendrent un allongement probable de la durée de validité du gradient approché.

L'évolution du gradient s'écrit:

$$E_{\mathbf{X}_n^{n+k-1}}\{ \| \nabla_{\mathbf{w}} C(\mathbf{w}_{n+k}) - \nabla_{\mathbf{w}} C(\mathbf{w}_n) \| \mid \mathbf{X}_0^{n-1} \}$$

$$\leq D'' \, E_{\mathbf{X}_n^{n+k-1}}\{ \| \mathbf{w}_{n+k} - \mathbf{w}_n \| \} \qquad (2\text{-}72)$$

On déduit alors de (2-71):

$$\boxed{\begin{array}{l} \forall\, \eta, \exists\, K, n_0, \forall\, n \geq n_0,\ k \geq 0: \\ P_{\mathbf{X}_0^{n-1}}\left(E_{\mathbf{X}_n^{n+k-1}}\{ |\nabla_{\mathbf{w}} C(\mathbf{w}_{n+k}) - \nabla_{\mathbf{w}} C(\mathbf{w}_n)| \mid \mathbf{X}_0^{n-1} \} < (\alpha' + K\beta') \sum_{i=0}^{k-1} \varepsilon_{n+i}^{+} \right) \geq 1-\eta \end{array}} \qquad (2\text{-}73)$$

Dans la mesure ou ε_n^+ tend vers 0, cette expression traduit l'allongement probable de la validité du gradient de la fonction de coût total.

viii La convergence presque sûre de C(w), l'allongement probable de la durée de validité du gradient approché et la sommation arbitrairement élevée du pas d'apprentissage impliquent la convergence presque sûre de $|\nabla_w C(w)|$ vers 0.

L'évolution du coût s'écrit:

$$E_{\mathbf{X}_n^{n+k-1}} \{ C(\mathbf{w}_{n+k}) - C(\mathbf{w}_n) \mid \mathbf{X}_0^{n-1} \}$$

$$= E_{\mathbf{X}_n^{n+k-1}} \{ \sum_{i=0}^{k-1} (\Gamma_{n+i} \nabla_{\mathbf{w}} J(\mathbf{x}_{n+i}, \mathbf{w}_{n+i})) \mid \mathbf{X}_0^{n-1} \}$$

$$= E_{\mathbf{X}_n^{n+k-1}} \{ \sum_{i=0}^{k-1} (\Gamma_{n+i} \nabla_{\mathbf{w}} C(\mathbf{w}_{n+i})) \mid \mathbf{X}_0^{n-1} \} \qquad (2\text{-}74)$$

D'après la convergence presque sûre de $C(\mathbf{w}_n)$ cette égalité donne:

$\forall\, \varepsilon, \eta', \exists\, n_0, \forall\, n \geq n_0,\ k \geq 0$:

$$P_{\mathbf{X}_0^n}\left(\left| E_{\mathbf{X}_n^{n+k-1}}\{ \sum_{i=1}^{k} (\Gamma_{n+i} \nabla_{\mathbf{w}} C(\mathbf{w}_{n+i})) \mid \mathbf{X}_0^{n-1} \} \right| < \varepsilon \right) \geq 1-\eta' \qquad (2\text{-}75)$$

L'allongement probable de la durée de validité du gradient approché (2-73) implique alors:

$$\forall\ \varepsilon,\eta,\eta',\ \exists\ K,n_0,\ \forall\ n\geq n_0,\ k\geq 0: \qquad (2\text{-}76)$$
$$P_{\mathbf{X}_0^{n-1}}\left((|\nabla_{\mathbf{w}} C(\mathbf{w}_n)| - (\alpha'+K\beta') \sum_{i=0}^{k-1} \varepsilon_{n+i}^{+}) \sum_{i=0}^{k-1} \varepsilon_{n+i}^{-} < \varepsilon \right) \geq 1-\eta-\eta'$$

ou encore:

$$\forall\ \varepsilon,\eta,\eta',\ \exists\ n_0,\ \forall\ n\geq n_0,\ k\geq 0:$$
$$P_{\mathbf{X}_0^{n-1}}\left(|\nabla_{\mathbf{w}} C(\mathbf{w}_n)| < \frac{\varepsilon}{\sum_{i=0}^{k-1} \varepsilon_{n+i}^{-}} + (\alpha'+K\beta') \sum_{i=0}^{k-1} \varepsilon_{n+i}^{+} \right) \geq 1-\eta-\eta' \qquad (2\text{-}77)$$

Pour conclure à la convergence de $\nabla_{\mathbf{w}} C(\mathbf{w}_n)$, il faut encore faire une hypothèse sur le pas d'apprentissage, afin que la somme partielle des ε_n^- puisse rester aussi grande que voulue par rapport à ε, alors que dans le même temps la somme partielle des ε_n^- doit rester aussi petite que voulue. On déduit de (2-77) et de l'hypothèse 4 (2-44), (2-45):

$$\forall\ \varepsilon'',\eta'',\ \exists\ n_0,\ \forall\ n\geq n_0 : P_{\mathbf{X}_0^{n-1}}\left(|\nabla_{\mathbf{w}} C(\mathbf{w}_n)| < \varepsilon'' \right) \geq 1 - \eta'' \qquad (2\text{-}78)$$

En d'autres termes, la suite $\nabla_{\mathbf{w}} C(\mathbf{w}_n)$ converge presque sûrement vers $\mathbf{0}$.

ix Théorème: moyennant quelques hypothèses de régularité et de conditionnement, l'algorithme d'apprentissage par descente de gradient stochastique fait converger presque sûrement la fonction de coût global et converger presque sûrement son gradient vers 0.

Par application de l'algorithme d'apprentissage par descente de gradient stochastique:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \Gamma_n \nabla_{\mathbf{w}} J(\mathbf{x}_n, \mathbf{w}_n)\ , \qquad (2\text{-}79)$$

le processus de fonction de coût globale $C(\mathbf{w}_n)$ converge presque sûrement vers une variable aléatoire et

le processus de gradient local $\nabla_{\mathbf{w}} C(\mathbf{w}_n)$ converge presque sûrement vers la valeur $\mathbf{0}$

quand:

- le pas d'apprentissage est convenable: Γ_n est symétrique définie positive, indépendante du processus stochastique $\mathbf{x}_n$ et ses valeurs propres minimales ε^-_n et maximales ε^+_n restent solidaires et ne sont ni trop fortes, ni trop faibles:

$$\sum_{n=0}^{\infty} \varepsilon^{+2}_n < \infty,\ \sum_{n=0}^{\infty} \varepsilon^-_n = \infty,\ \forall \varepsilon'' > 0,\ \exists \varepsilon,\ n_0,\ \forall n \geq n_0,\ \exists k : \begin{cases} \sum_{n}^{n+k} \varepsilon^+_n \leq \varepsilon'' \\ \frac{\varepsilon}{\varepsilon''} \leq \sum_{n}^{n+k} \varepsilon^-_n \end{cases} , \qquad (2\text{-}80)$$

- la fonction de coût globale C est suffisamment régulière:

$$\exists\ C_{min},\ \forall\ \mathbf{w} : C(\mathbf{w}) \stackrel{\Delta}{=} E_{\mathbf{x}}\{\ J(\mathbf{x}, \mathbf{w})\ \} \geq C_{min} \qquad (2\text{-}81a)$$

$$\exists\ D'',\ \forall\ \mathbf{w}\ ,\ \mathbf{v} : |\ \nabla_{\mathbf{w}}\ C(\mathbf{w} + \mathbf{v}) - \nabla_{\mathbf{w}}\ C(\mathbf{w})\ | \leq D''\ |\ \mathbf{v}\ |\ , \qquad (2\text{-}81b)$$

$$\forall\ \mathbf{w} : E_{\mathbf{x}}\{\ \nabla_{\mathbf{w}}\ J(\mathbf{x}, \mathbf{w})\ \} = \nabla_{\mathbf{w}}\ E_{\mathbf{x}}\{\ J(\mathbf{x}, \mathbf{w})\ \} \stackrel{\Delta}{=} \nabla_{\mathbf{w}}\ C(\mathbf{w})\ , \qquad (2\text{-}81c)$$

$$\exists\ a,b,\ \forall\ \mathbf{w} : E_{\mathbf{x}}\{\ |\ \nabla_{\mathbf{w}}\ J(\mathbf{x}, \mathbf{w})\ |^2\ \} \leq (a\ C(\mathbf{w}) + b)\ , \qquad (2\text{-}81d)$$

x *Premier corollaire: moyennant quelques hypothèses de régularité et de conditionnement, l'algorithme d'apprentissage par descente de gradient stochastique fait presque sûrement converger la configuration de paramètres vers un point singulier de la fonction de coût.*

On appelle $sing(C)$ l'ensemble des points singuliers de la fonction de coût global C caractérisés par $\nabla_{\mathbf{w}} C(\mathbf{w}) = 0$. Si en plus de (2-78) on suppose:

$$\inf_{\mathbf{w}\ :\ d(\mathbf{w}, sing(C)) > \varepsilon} |\nabla_{\mathbf{w}}\ C(\mathbf{w})\ | > 0 \qquad (2\text{-}82)$$

alors la propriété:

$$\forall\ \varepsilon'', \eta'', \exists\ n_0, \forall\ n \geq n_0 :\quad P_{\mathbf{x}_0}^{n-1}\left(\ d(\mathbf{w}_n\ ,\ sing(C)) \leq \varepsilon''\ \right) \geq 1 - \eta'' \qquad (2\text{-}83)$$

est vérifiée et le jeu de paramètres converge presque sûrement vers un point singulier de la fonction de coût.

xi *Analogie avec la mécanique des fluides et le recuit simulé*

La propriété (2-80) est encore insuffisante, car elle autorise la convergence vers les points selles

et les maxima locaux. Or, ces solutions parasites peuvent en général être écartées.

Considérons une nappe solide, à la surface de laquelle se trouve astreinte une fine lame d'un fluide compressible parfait. Ce fluide connaît, à un niveau microscopique, un mouvement brownien proportionnel à la température qui, associée aux variation de densité de la lame, engendre une pression macroscopique. Il est soumis à la gravité. Il semble clair que le fluide se concentrera dans les creux de la nappe et éventuellement sur ses plateaux (points selles) au fur et à mesure que la température s'affaiblira.

2.5 Conclusion

Il existe plusieurs familles de techniques d'optimisation qu'on peut répartir en deux catégories: l'optimisation non-dérivative (programmation linéaire, optimisation multiplement mono-dimensionelle par encadrement-dichotomie, méthode d'encadrement par simplex…) et l'optimisation dérivative (variantes de la descente de gradient et méthodes d'ordre supérieur).

Nous avons exposé dans ce chapitre les éléments principaux théoriques de l'optimisation par descente de gradient et de ses nombreuses variantes, ainsi que les méthodes quasi-newtonniennes (du second ordre). Nous nous sommes particulièrement concentrés sur le gradient stochastique, particulièrement adapté pour l'optimisation de fonctions de coûts ayant la forme d'une espérance de coûts contributifs de chacun des exemples stochastiques issus d'une distribution.

En pratique, la descente de gradient *stricto sensu* (anisotrope globale) est peu utilisée. On lui préfère certaines variantes: descente de gradient stochastique, sous contrainte, biaisée (selon des heuristiques utilisant le pas d'apprentissage anisotrope)…

Nous avons également établi un théorème original de convergence pour une large classe d'algorithmes de descente de gradient stochastique. Ce théorème reprend et étend le théorème de L. Bottou.

3 Classification

Quotidiennement, nous effectuons de nombreuses tâches de classification. Décider de la nature d'un objet ou de sa couleur, entendre et reconnaître un mot, lire une chaîne de caractères, faire un diagnostic… sont autant de tâches de classification.

Au sens large, la classification inclue également la quantification, utilisée pour la compression de données, ou encore la préclassification, qui consiste à regrouper les données en macro-classes.

L'approche que nous avons développé est basée sur la minimisation de fonctions de risques. Nous allons tout d'abord introduire la théorie de décision statistique et notamment la décision bayésienne classique (cf. § 3.1). Puis, nous discuterons des limites de cette théorie, en ce qu'elle conduit à une fonction de risque discontinue qui se prête peu à optimisation. Nous détaillerons ensuite les trois types d'approches paliatives qui visent à contourner ce problème fondamental. Nous passerons ainsi en revue la plupart des fonctions de coût utilisées pour l'optimisation de classifieurs paramètriques (cf. § 3.2, § 3.3).

La distinction des trois types d'approches paliatives (cf. § 3.1.7, § 3.1.8 et § 3.4) est, à nos yeux, un point important et relativement original; elle guide les travaux de notre équipe en matière de classification. Le premier type d'approche (cf. § 3.2), qui consiste à approcher le risque de façon différentiable est original. Il conduira dans les chapitres suivants aux fonctions de coût LVQ2+ (cf. § 4.2.3.1) et à leurs adaptations pour la programmation dynamique (cf. § 5.2.1.2).

3.1 Quelques éléments de la théorie de décision statistique

Les problèmes de décision statistique et de reconnaissance des formes ont fait depuis plusieurs décennies l'objet d'études poussées, qui ont conduit à des formalisations rigoureuses [1], [2], [3], [4], [5].

3.1.1 Notations

Considérons une distribution $(\mathbf{X}, p)$ dont les représentants $\mathbf{x} = (\tilde{\mathbf{x}}, k)$ sont constitués d'une observation $\tilde{\mathbf{x}}$ à valeurs dans $\tilde{\mathbf{X}}$ et d'une classe k parmi K classes possibles. Considérons également un système $\Lambda_{\mathbf{w}}$ dont les paramètres $\mathbf{w}$ conduisent pour une observation $\tilde{\mathbf{x}}$ à une classification m parmi M classifications possibles. Selon les problèmes, une classification représentera une classe, un amas d'exemples, une quantification ou bien une macro classe...

Nous noterons dans la suite:

- $\mathbf{X}_m$ le sous ensemble de $\mathbf{X}$ des $\mathbf{x}$ dont l'observation $\tilde{\mathbf{x}}$ est classifiées m par le système $\Lambda_{\mathbf{w}}$,
- $\tilde{\mathbf{X}}_m$ le sous ensemble de $\tilde{\mathbf{X}}$ des observations $\tilde{\mathbf{x}}$ classifiées m par le système $\Lambda_{\mathbf{w}}$,
- $\mathbf{X}^k$ le sous ensemble de $\mathbf{X}$ des $\mathbf{x}$ de classe k ,
- $\mathbf{X}_m^k$ le sous ensemble de $\mathbf{X}$ des $\mathbf{x}$ de classe k dont l'observation $\tilde{\mathbf{x}}$ est classifiées m par le système ,
- $\mathbf{p}(\tilde{\mathbf{x}}|l)$ le vecteur des fonctions probabilités conditionnelles $(p(\tilde{\mathbf{x}}|k))_{1\le k\le K}$ issues de la

---

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

2 **Nillson N.J.:** *Learning Machines: Foundation of trainable Patter-Clasiciation Systems* - MacGtraw Hill (1965)

3 **Devijer P. A., Kittler J.:** *Pattern Recognition: a Stztistical Approach* - Prentice Hall International (1982)

4 **Berger J.O.:** *Stastistical décision theory and Bayesian alalysis* - Spichoper (1985)

5 **Thodbey H.:** *Ace of Bayes: application of neural networks with pruning* - TR n° 1132E Danish meet. research institute (1993)

distribution $(\mathbf{X}, p)$ et $\mathbf{p}_\bullet$ le vecteur fonctionnel correspondant ,

- $\mathbf{p}(l\tilde{\mathbf{x}})$ le vecteur des probabilités *a posteriori* $(p(k|\tilde{\mathbf{x}}))_{1\le k\le K}$ issues de la distribution $(\mathbf{X}, p)$ et $\mathbf{p}_l$ le vecteur fonctionnel correspondant ,

- $\mathbf{P}$ le vecteur $P(k)$ des probabilité *a priori* de la classe k issues de la distribution $(\mathbf{X}, p)$,

- $\tilde{\mathbf{y}}_{sm}$ la frontière séparant $\tilde{\mathbf{X}}_s$ de $\tilde{\mathbf{X}}_m$., une telle frontière est généralement une hyper-surface (mais pas forcément connexe),

- F_{km} représente la fonction de coût propre à la classification en m d'un représentant $(\tilde{\mathbf{x}},k)$ de la distribution $(\mathbf{X},p)$.

3.1.2 Risque moyen

La classification consiste à attribuer une étiquette de classe/identité/catégorie à une forme/donnée/observation. Idéalement, les classes constituent une partition de l'ensemble des observations qu'on traite. Les systèmes de classification déterministes respectent cette hypothèse. Cependant, les données peuvent être contradictoires. Ainsi, deux personnes donnent la même image dans l'obscurité; deux mots couverts par la tempête ne pourront être distingués l'un de l'autre; deux phénomènes sociaux peuvent fort bien engendrer pour un temps les mêmes indices macro-économiques…

Quand nous manquons d'information *a priori* ou quand nous ne savons pas les exploiter intégralement, nous restons capables, dans une certaine mesure, de prendre des décisions. Sans doute, notre jugement se fonde alors inconsciemment sur la notion de risque moyen qui pondère la probabilité de nos erreurs par la gravité de leurs conséquences potentielles.

Dans sa forme la plus générale, le risque moyen s'écrit:

$$R(\mathbf{w}) \quad \stackrel{\Delta}{=} \sum_{m=1}^{M} \int_{\mathbf{X}_m} F_{km}(\tilde{\mathbf{x}},\mathbf{w})\; p(\mathbf{x})\; d\mathbf{x} \;,$$

$$\begin{cases} = \sum_{k=1}^{K} \sum_{m=1}^{M} \int_{\tilde{\mathbf{X}}_m} F_{km}(\tilde{\mathbf{x}},\mathbf{w})\; P(k|\tilde{\mathbf{x}})\; p(\tilde{\mathbf{x}})\; d\tilde{\mathbf{x}} \;, \\ = \sum_{k=1}^{K} \sum_{m=1}^{M} \int_{\tilde{\mathbf{X}}_m} F_{km}(\tilde{\mathbf{x}},\mathbf{w})\; P(k)\; p(\tilde{\mathbf{x}}|k)\; d\tilde{\mathbf{x}} \;, \end{cases} \tag{3-1}$$

où:

Cette forme est particulièrement générale. Elle fait appel à une fonction $F_{km}(\tilde{\mathbf{x}},\mathbf{w})$ et non à un simple tableau de coûts des confusions F_{km} (à partir du § 3.1.6, nous nous limiterons au cadre plus restrictif de la décision Bayésienne classique). Elle présente K situations (classes) pouvant donner lieu à M décisions (classifications),

Le système de décision optimal au sens du risque moyen propose l'association observation-classification la moins coûteuse possible. Distinguer les classes k et les classifications m permet à ce formalisme de recouvrir plusieurs cas couramment rencontrés:

- La classification simple où chaque classe k correspond à une unique classification m. Dans ce cas, $K = M$.

- La préclassification, qui consiste à associer les formes d'entrées à des macro-classes (ou typologies), en vue d'une classification ultérieure par des modules spécialisés par macro-classe. Dans ce cas, on a $K \geq M$.

- Les systèmes de quantification vectorielle (vector quantization ou clusterization), qui regroupent les formes d'entrée en plusieurs classifications. Les techniques de quantification vectorielle sont employées comme prétraitements pour des systèmes ne pouvant traiter que des entrées discrètes, comme compression (radicale !), ou bien comme aiguilleurs dans les systèmes modulaires par arbres (cf. § 4.4.3).

3.1.3 Règle de décision fondamentale

Tout système de classification cherche à minimiser un risque moyen de la forme (3-1). Avant de faire intervenir l'apprentissage des paramètres **w**, il est utile d'étudier la *règle de décision théorique* que le risque moyen induit pour le système: dans une configuration donnée **w** du système, quelle règle devrait être utilisée pour classifier optimalement une observation ? Plus formellement, quelle est la partition optimale de l'espace des observations $\tilde{\mathbf{X}}$ en domaines $\{\tilde{\mathbf{X}}_m\}_{1\leq m\leq M}$ étant donné les fonctions de coûts F_{km} et une configuration de paramètres **w** ? Il existe une réponse théorique simple à cette question.

La décision de classifier une observation $\tilde{\mathbf{x}}$ en s engendre un coût local:

$$F_s(\tilde{\mathbf{x}},\mathbf{w}) \stackrel{\Delta}{=} \sum_{k=1}^{K} F_{ks}(\tilde{\mathbf{x}},\mathbf{w})\ P(k)\ \ p(\tilde{\mathbf{x}}|k) \qquad (3\text{-}2)$$

La règle de décision:

$$m = \min_{s} \; F_s(\mathbf{x},\mathbf{w}) \; . \tag{3-3}$$

minimise partout le coût local (donc le coût total). Cette règle de décision conduit au risque moyen (minimal en terme de règle de décision) pour **w** fixé :

$$R_{min}(\mathbf{w}) = \sum_{k=1}^{K} \int_{\tilde{\mathbf{x}}} \min_{m} \sum_{m=1}^{M} F_{km}(\tilde{\mathbf{x}},\mathbf{w}) \; P(k) \; p(\tilde{\mathbf{x}}|k) \; d\tilde{\mathbf{x}} \; , \tag{3-4}$$

Le critère (3-3) est purement local, il a cependant le défaut de faire intervenir $p(\tilde{\mathbf{x}}|k)$ *qu'on ne sait en général pas évaluer*. Le risque minimal moyen (3-4) doit donc être considéré comme un risque inhérent aux données traitées, un minimum théorique en deçà duquel les autres règles de décisions, basées sur la notion de risque, ne peuvent mener.

3.1.4 Fonctions discriminantes

La règle de décision:

$$m = \min_{s} \; F_s(\tilde{\mathbf{x}},\mathbf{w}) \tag{3-5}$$

conduit à définir les fonctions discriminantes:

$$\begin{aligned} f_{sm}(\tilde{\mathbf{x}},\mathbf{w}) \quad &\stackrel{\Delta}{=} F_s(\tilde{\mathbf{x}},\mathbf{w}) - F_m(\tilde{\mathbf{x}},\mathbf{w}) \\ &= \sum_{k=1}^{K} \left(F_{ks}(\tilde{\mathbf{x}},\mathbf{w}) - F_{km}(\tilde{\mathbf{x}},\mathbf{w}) \right) P(k) \; p(\tilde{\mathbf{x}}|k) \; . \end{aligned} \tag{3-6}$$

Le signe d'une fonction discriminante permet de choisir "la moins risquée" des deux classifications qu'elle compare. En général, les fonctions discriminantes prennent des valeurs strictement positives ou négatives. Il arrive cependant qu'elles prennent des valeurs nulles à l'occasion d'observations pour lesquelles les deux classifications présentent des risques équivalents.

Ceci se produit quand on se trouve sur une frontière $\tilde{\mathbf{y}}_{sm}$, les deux classifications (distinctes) m et s ont un coût ponctuellement identique: leur fonction discriminante f_{sm} est nulle.

$$\forall \; \tilde{\mathbf{x}} \in \tilde{\mathbf{y}}_{sm} : f_{sm}(\tilde{\mathbf{x}},\mathbf{w}) = \sum_{k=1}^{K} \left(F_{ks}(\tilde{\mathbf{x}},\mathbf{w}) - F_{km}(\tilde{\mathbf{x}},\mathbf{w}) \right) P(k) \; p(\tilde{\mathbf{x}}|k) = 0 \tag{3-7}$$

La présente partie 3.1 est consacrée à la théorie de décision statistique. Aussi, les fonctions

discriminantes (3-6) font-elles références à l'information *a posteriori* k. Une fois l'apprentissage accompli, les règles de décision des systèmes opérationnels ne font bien sûr référence qu'à la forme $\tilde{\mathbf{x}}$. On parle encore de fonctions discriminantes, mais les fonctions théoriques (3-6) sont remplacées par des fonctions dont découle la règle de décision. La notion de frontière de classification reste bien entendu valable et la définition (3-7) est conservée.

3.1.5 Condition de minimalité du risque moyen

La règle de décision (3-5) nous permet théoriquement de minimiser le risque moyen dans une configuration de paramètres **w** déterminée. A présent, nous étudions l'optimisation de **w**.

Réexprimé à l'aide de $\mathbb{1}_{\tilde{\mathbf{X}}_m}$ le risque moyen s'écrit:

$$R(\mathbf{w}) = \sum_{k=1}^{K} \sum_{m=1}^{M} \int_{\tilde{\mathbf{X}}} \mathbb{1}_{\tilde{\mathbf{X}}_m}(\tilde{\mathbf{x}},\mathbf{w})\, F_{km}(\tilde{\mathbf{x}},\mathbf{w})\, P(k)\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} \,. \qquad (3\text{-}8)$$

La condition de minimalité (2-2) devient:

$$\begin{aligned} \mathbf{0} &= \nabla R(\mathbf{w}) \\ &= \sum_{k=1}^{K} \sum_{m=1}^{M} \left(\int_{\tilde{\mathbf{X}}} \mathbb{1}_{\tilde{\mathbf{X}}_m}(\tilde{\mathbf{x}},\mathbf{w})\, \nabla_{\mathbf{w}} F_{km}(\tilde{\mathbf{x}},\mathbf{w})\, P(k)\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} \right. \\ &\qquad \left. + \int_{\tilde{\mathbf{X}}} \nabla_{\mathbf{w}} \mathbb{1}_{\tilde{\mathbf{X}}_m}(\tilde{\mathbf{x}},\mathbf{w})\, F_{km}(\tilde{\mathbf{x}},\mathbf{w})\, P(k)\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} \right) \end{aligned} \qquad (3\text{-}9)$$

Le gradient de la fonction caractéristique présent dans la deuxième intégrale constitue une masse de Dirach. Il est possible d'écrire cette expression sous une forme plus classique:

$$\begin{aligned} \mathbf{0} &= \sum_{k=1}^{K} \sum_{m=1}^{M} \left(\int_{\tilde{\mathbf{X}}} \mathbb{1}_{\tilde{\mathbf{X}}_m}(\tilde{\mathbf{x}},\mathbf{w})\, \nabla_{\mathbf{w}} F_{km}(\tilde{\mathbf{x}},\mathbf{w})\, P(k)\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} \right. \\ &\qquad \left. + \sum_{s=1}^{M} \int_{\tilde{\mathfrak{Y}}_{sm}} \left[F_{ks}(\tilde{\mathbf{x}},\mathbf{w}) - F_{km}(\tilde{\mathbf{x}},\mathbf{w}) \right] P(k)\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} \right) \\ &= \sum_{k=1}^{K} \sum_{m=1}^{M} \int_{\tilde{\mathbf{X}}} \mathbb{1}_{\tilde{\mathbf{X}}_m}(\tilde{\mathbf{x}},\mathbf{w})\, \nabla_{\mathbf{w}} F_{km}(\tilde{\mathbf{x}},\mathbf{w})\, P(k)\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} \end{aligned}$$

$$+ \sum_{m=1}^{M} \sum_{s=1}^{M} \int_{\tilde{\mathbf{y}}_{sm}} \sum_{k=1}^{K} \left[F_{ks}(\tilde{\mathbf{x}},\mathbf{w}) - F_{km}(\tilde{\mathbf{x}},\mathbf{w}) \right] P(k) p(\tilde{\mathbf{x}}|k) d\tilde{\mathbf{x}} \qquad (3\text{-}10)$$

Chacune des deux intégrales de cette somme a un sens intuitif. La première représente l'évolution locale de chaque fonction de coût. La deuxième représente les flux engendrés par les fonctions discriminantes (3-6) aux frontières $\tilde{\mathbf{y}}_{sm}$. Or d'après (3-7), aux frontières $\tilde{\mathbf{y}}_{sm}$ les fonctions discriminantes sont nulles. On en déduit que la deuxième intégrale est nulle. La condition de minimalité du risque moyen se simplifie donc pour devenir:

$$\begin{aligned} \mathbf{0} &= \sum_{k=1}^{K} \sum_{m=1}^{M} \int_{\tilde{\mathbf{X}}} \mathbb{1}_{\tilde{\mathbf{X}}_m}(\tilde{\mathbf{x}},\mathbf{w}) \, \nabla_{\mathbf{w}} F_{km}(\tilde{\mathbf{x}},\mathbf{w}) \, P(k) \, p(\tilde{\mathbf{x}}|k) \, d\tilde{\mathbf{x}} \\ &= E_{\mathbf{x}} \{ \mathbb{1}_{\tilde{\mathbf{X}}_m}(\tilde{\mathbf{x}},\mathbf{w}) \, \nabla_{\mathbf{w}} F_m(\tilde{\mathbf{x}},\mathbf{w}) \} \end{aligned} \qquad (3\text{-}11)$$

ce qui est de la forme:

$$\mathbf{0} = E_{\mathbf{x}} \{ \phi(\mathbf{x},\mathbf{w}) \} \qquad (3\text{-}12)$$

Cette condition est un cas particulier de (2-1) et les remarques formulées au chapitre précédent restent valables. En particulier, l'expression (3-12) est une condition nécessaire mais pas suffisante de minimalité; de plus, elle est bien souvent analytiquement inconnue: on en est réduit à utiliser des algorithmes d'apprentissage.

Il faut également souligner que l'expression (3-12) fait appel aux fonctions de coût F_{km}. Or, la définition en (3-2) de ces fonctions fait appel aux termes $P(k)$ et $p(\tilde{\mathbf{x}}|k)$ qui constituent une connaissance complète de la distribution. En pratique, on ne dispose pas de cette connaissance, sinon pour une distribution empirique presque toujours trop pauvre pour autoriser une estimation de bonne qualité par simple comptage.

Néanmoins, pour de nombreux système de classification, la condition de minimalité du risque moyen prend la forme (3-12). En effet, tout système de classification $\Lambda_{\mathbf{w}}$ repose sur une règle de décision. Bien souvent, cette règle de décision est basée sur des fonctions discriminantes de la forme (3-6a) faisant appel à des fonctions paramètriques $F_m(\tilde{\mathbf{x}},\mathbf{w})$, dont la forme est déterminée *a priori* et fait partie de la définition du système $\Lambda_{\mathbf{w}}$. L'optimisation d'un tel système de classification conduit alors à une condition de minimalité du risque moyen de la forme (3-12).

3.1.6 Décision bayésienne classique

Très souvent, le risque moyen local lié à une décision de classification ne dépend que de la classe de l'exemple et de la classification effectuée. En d'autres termes:

$$F_{km}(\tilde{\mathbf{x}},\mathbf{w}) \equiv F_{km}$$

Ceci constitue le cadre de la décision bayésienne classique. Le risque moyen total (3-1) se simplifie alors pour devenir:

$$R = \sum_{k=1}^{K} \sum_{m=1}^{M} \int_{\tilde{\mathbf{X}}_m} F_{km}\, P(k)\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}}\ , \qquad (3\text{-}13)$$

et la règle théorique de décision bayésienne (3-3) s'écrit:

$$m = \min_s\ F_{ks}\, P(k)\ \ p(\tilde{\mathbf{x}}|k) \qquad (3\text{-}14)$$

On pourrait croire à la lecture de ces deux expressions que le problème de la classification est pour ainsi dire résolu. Comme l'avons laissé entendre en commentant (3-12), il n'en est rien. Nous allons à présent détailler les limites de la théorie de décision statistique.

3.1.7 Limites de la théorie de décision statistique

En général, on ne sait pas exprimer la quantité $p(\tilde{\mathbf{x}}|k)$. En effet, pour ce faire, il faut pouvoir formuler et résoudre le problème de façon analytique, ou bien disposer d'informations *a posteriori* complétant toutes les lacunes de l'information *a priori*. Sans cela, on en est réduit faire des approximations probabilistes.

A plus forte raison, on se trouve généralement dans l'*incapacité d'utiliser la règle de décision* (3-3) et de minimiser directement (3-1). On en est réduit à faire appel à des techniques de classifications *a posteriori* telles que les méthodes de plus proche voisinage (k plus proches voisins, fenêtres de Parzen…) et/ou d'apprentissage paramétrique (perceptrons multicouches, modèles de Markov cachés…) pour obtenir une approximation de la règle de décision théorique.

Nous nous plaçons désormais dans le cadre de l'approche bayésienne classique. En notant:

- $J(i,j) = F_{ij}$ et
- $o(\tilde{\mathbf{x}},\mathbf{w})$ la classification que le système effectue pour une observation $\tilde{\mathbf{x}}$ quand il a $\mathbf{w}$ pour configuration de paramètres,

le risque moyen du système s'écrit:

$$R(\mathbf{w}) = \sum_{k=1}^{K} \int_{\tilde{\mathbf{X}}} J(k,o(\tilde{\mathbf{x}},\mathbf{w}))\, P(k)\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} \,. \qquad (3\text{-}15)$$

Fondamentalement, on cherche à minimiser le risque moyen de (mauvaise) classification. Or, pour un exemple donné le risque local ne peut prendre, dans le cadre restrictif de l'approche bayésienne, que les valeurs discrètes F_{ij}. L'estimation $\hat{R}(\mathbf{w})$ (nommée risque global ou risque empirique) du risque moyen (3-13) à l'aide d'un nombre fini d'exemples, est donc une fonction constante par morceaux dans l'espace des paramètres **w**. Cette fonction ne se prête pas à une optimisation par descente de gradient.

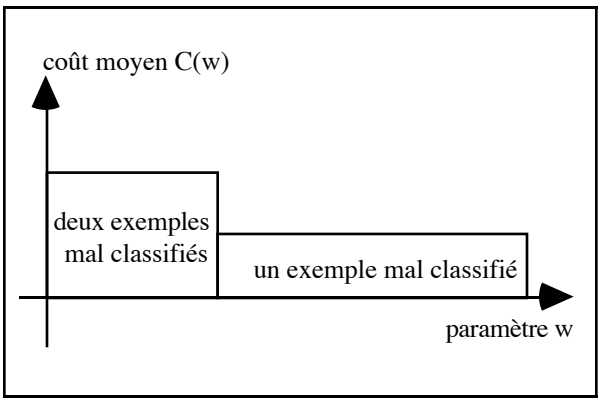


Figure 3.1
Discontinuité du risque empirique

Quand la distribution empirique est finie, et que la fonction de coût est à valeurs discrètes, le risque empirique est une fonction constante par morceaux.
Dans la pratique, ces conditions sont vérifiés notamment pour les problèmes de classification.

On en déduit que les systèmes d'apprentissage fondés sur la descente de gradient ne peuvent être utilisés aussi directement pour résoudre les problèmes de classification.

3.1.8 Approches palliatives

A ce stade, en matière d'apprentissage paramétrique de la règle de décision théorique bayésienne, la situation semble compromise:

- en général, on ne sait pas exprimer la quantité $p(\tilde{\mathbf{x}}|k)$ et à plus forte raison, on se trouve dans l'incapacité d'utiliser la règle de décision (3-3);
- de plus, la distribution empirique provient généralement d'une collection finie d'exemples; les densités de probabilités résultantes sont nulles presque partout et infinies aux emplacements des exemples (masses de Dirach);
- enfin, le risque local $J(k,o(\tilde{\mathbf{x}},\mathbf{w}))$ et le risque global (ou risque empirique) $\hat{R}(\mathbf{w})$ sont constants par morceaux dans l'espace des paramètres **w** de tout système de classification $\Lambda_{\mathbf{w}}$. On ne peut donc directement minimiser le risque empirique $\hat{R}(\mathbf{w})$ par descente de

gradient.

On est donc amené à définir une procédure de conception de systèmes paramétriques de classification. Une telle procédure peut se décomposer en deux phases. La première consiste à décider de la nature du système; elle conduit à la définition d'un critère de décision dépendant de paramètres ajustables. La deuxième phase consiste à exploiter une stratégie d'ajustement de ces paramètres afin de minimiser le risque moyen induit par la distribution empirique et par le critère de décision.

Pour cela, on dispose fondamentalement de trois approches:

- l'optimisation directe d'une approximation dérivable du risque moyen R, pour laquelle on sait exprimer une règle de décision,
- la modélisation de densités de probabilités conditionnelles $p(\tilde{\mathbf{x}}|k)$,
- l'estimation directe des probabilités a posteriori $P(k|\tilde{\mathbf{x}})$.

Ces trois approches conduisent à des systèmes de classification sensiblement différents.

3.2 Optimisation directe d'une approximation du risque bayésien

Nous étudions à présent la première approche palliative proposée au § 3.1.8.

Considérons un système paramètré, pour lequel un critère de décision a été fixé. A défaut d'effectuer une descente de gradient pour minimiser le risque empirique, il est en général possible de faire "évoluer dans le bon sens" les paramètres affectant directement le critère de décision, jusqu'à obtenir un minimum local strict, car:

- il existe des techniques d'optimisation ne faisant pas appel à la descente de gradient et susceptibles de traiter des fonctions discontinues; ces techniques se heurtent néanmoins à la très grande difficulté d'optimiser des fonction de coût trop irrégulières dans des espaces de grande dimension;
- il existe des approximations différentiables de ce risque autorisant les méthodes d'optimisation fondées sur la descente de gradient; ce mémoire porte justement sur l'utilisation de telles techniques, aussi nous allons privilégier cette voie.

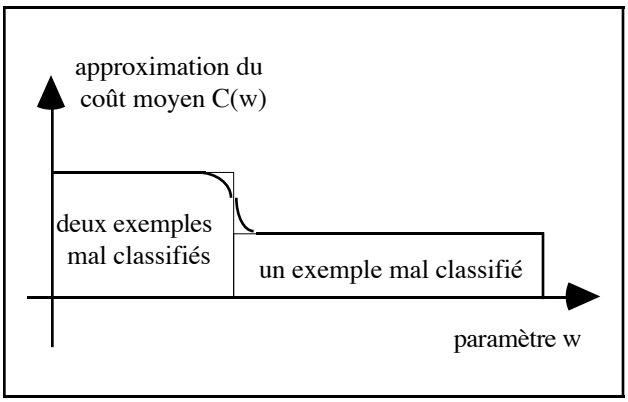


Figure 3.2
Approximation différentiable du risque empirique

Quand bien même le risque moyen est discontinu, il est possible d'en fournir des approximations différentiables.

Nous étudions dans les lignes qui suivent la possibilité d'obtenir des approximations différentiables du risque local et du risque moyen.

i Comparaison de scores

Un système de classification $\Lambda_{\mathbf{w}}$ repose sur une règle de décision. Toute règle de décision peut s'exprimer sous la forme de comparaison des scores (fonction à valeur dans $\mathbb{R}^+$) de chaque classification. Plus formellement pour toute règle de décision s on peut définir des scores s_l tels que:

$$s(\tilde{\mathbf{x}},\mathbf{w}) \triangleq \underset{l}{\operatorname{argmin}}\ s_l(\tilde{\mathbf{x}},\mathbf{w}) \tag{3-16}$$

où s_l représente le score de la classification l. Par analogie avec la minimisation du risque, nous employons la convention selon laquelle plus le score s_l est faible, plus la classification l est jugée adéquate par le système. Quand le critère de décision est suffisamment régulier en termes topologiques, on peut imposer que les scores soient des fonctions multiplement différentiables.

La comparaison de scores constitue le formalisme naturel de nombreux systèmes statistiques, markoviens ou neuronaux.

On peut réécrire le risque bayésien local à l'aide de ce formalisme:

$$J(\tilde{\mathbf{x}},\mathbf{w}) = J(k,s(\tilde{\mathbf{x}},\mathbf{w})) = J(k, \underset{l}{\operatorname{argmin}}\ s_l(\tilde{\mathbf{x}},\mathbf{w})) \tag{3-17}$$

Avant d'utiliser le formalisme de la comparaison de scores pour construire des approximations différentiables du risque moyen, il nous faut l'enrichir de quelques concepts supplémentaires.

ii Classifications gagnantes, correctes et désirées

Pour une forme $\tilde{\mathbf{x}}$ et un jeu de fonctions-scores (s_l), nous appellerons classifications gagnantes

$G(\tilde{\mathbf{x}},\mathbf{w})$ les classifications de moindres scores. En général, il existe une unique classification gagnante $g(\tilde{\mathbf{x}},\mathbf{w})$:

$$g(\tilde{\mathbf{x}},\mathbf{w}) \triangleq s(\tilde{\mathbf{x}},\mathbf{w}) = \underset{l}{\operatorname{argmin}}\ s_l(\tilde{\mathbf{x}},\mathbf{w}) \tag{3-18}$$

Les classifications correctes $E(\tilde{\mathbf{x}}, k,\mathbf{w})$ pour un exemple $(\tilde{\mathbf{x}},k)$ sont celles qui minimisent le coût local (en principe à valeurs discrètes):

$$E(\tilde{\mathbf{x}}, k,\mathbf{w}) \triangleq \underset{l}{\operatorname{argmin}}\ J(k,l)$$

La classification gagnante (de moindre score) se détermine *a priori*. Elle n'est pas nécessairement parmi les classifications correctes (de moindre coût local) qui se déterminent *a posteriori*.

Prenons l'exemple d'un système de classification dont la règle de décision est celle du plus proche voisin. Chaque référence constitue une classification possible. Si chaque classe est représentée par plusieurs références, il y aura plusieurs classifications correctes pour chaque exemple $(\tilde{\mathbf{x}}, k)$. En revanche, il n'y aura généralement qu'une seule classification gagnante: la plus proche référence, éventuellemnt incorrecte.

Nous appellerons classifications désirées les classifications correctes qui, en sus, vérifient des conditions supplémentaires, permettant de penser qu'une modification minime des paramètres permettrait de traiter correctement l'exemple. Le plus souvent, par soucis de simplicité, on choisit pour classifications désirées les classifications correctes dont les scores sont les meilleurs (i.e. les plus faibles). En général, il existe alors une unique classification désirée $e(\tilde{\mathbf{x}}, k, \mathbf{w})$:

$$e(\tilde{\mathbf{x}}, k,\mathbf{w}) \triangleq \underset{l \in E(\tilde{\mathbf{x}}, k,\mathbf{w})}{\operatorname{argmin}}\ s_l(\tilde{\mathbf{x}},\mathbf{w}) \tag{3-19}$$

Pour des raisons de commodité, nous introduisons les notations:

$$s_g(\tilde{\mathbf{x}},\mathbf{w}) \triangleq s_{g(\tilde{\mathbf{x}},\mathbf{w})}(\tilde{\mathbf{x}},\mathbf{w})$$

et

$$s_e(\tilde{\mathbf{x}}, k,\mathbf{w}) \triangleq s_{e(\tilde{\mathbf{x}}, k,\mathbf{w})}(\tilde{\mathbf{x}},\mathbf{w})$$

Nous venons d'introduire ces quelques concepts (scores, classifications gagnantes correctes et désirées) dans le but de construire des approximations différentiables du risque local.

iii approximation différentiable du risque local

Il s'agit de lisser le risque local afin d'en obtenir des approximations différentiables. Or, le risque local est une fonction constante par palier, où la hauteur de chaque palier est déterminée par le coût $J(k,g)$ de la classification gagnante g et la classe k de l'exemple $\mathbf{x}$. Quand les paramètres du système évoluent (lors de l'apprentissage), pour un même exemple, la classification gagnante peut changer. L'objet du lissage du risque local est de permettre à un système d'apprentissage par descente de gradient d'évoluer vers des classifications moins coûteuses (et donc plus en adéquation avec la classe de chaque exemple).

Il est généralement possible - en cas de classification insatisfaisante - de faire évoluer les paramètres $\mathbf{w}$ de manière à augmenter les scores des classifications gagnantes mais usurpatrices et réduire les scores des classifications désirées. On espère que, progressivement, les classifications usurpatrices seront plus rares.

De nombreux systèmes d'apprentissage utilisent de manière plus ou moins convenable ce principe: en fait, un grand nombre d'entre eux "oublient" le but fondamental qui est d'optimiser le risque moyen. On confond ainsi la notion de score qui détermine le critère de décision et celle de risque, qui détermine la performance du système de classification.

Cependant, il n'y a pas de méthode permettant d'obtenir une approximation continue simple dans le cas général. En effet, il existe des points de l'espace des paramètres qui constituent des frontières entre plusieurs plateaux d'altitudes différentes: autour de ces configurations de paramètres, les classifications gagnantes sont multiples. Pour assurer la continuité de l'approximation, on est obligé de tenir compte de toutes les altitudes des plateaux proches.

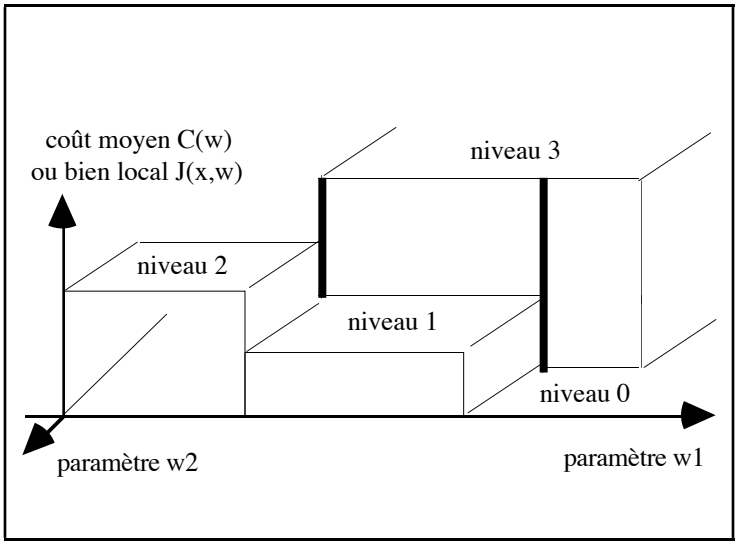


Figure 3.3
Discontinuité complexe du risque moyen

Le risque moyen (de même que le risque local) est une fonction constante par paliers. Il arrive qu'une configuration de paramètres mette en contact plusieurs de ces paliers.
Dans cet exemple, ces configurations de paramètres sont visualisées par des segments de droites épais.

Bien que cela ne soit en aucune façon un obstacle théorique, prendre en compte toutes les

altitudes des plateaux proches rend malaisée l'expression et l'optimisation d'une telle fonction. On se retrouve donc devant un choix difficile entre:

- un système discontinu; or, on connait la difficulté de l'optimisation en grande dimension,
- un système dont l'expression analytique est malaisée et calculatoirement lourde (multi-coûts) qui prend en compte toutes les altitudes des plateaux proches,
- un système simplifié où il n'existe pour chaque classe que deux altitudes de plateaux (coûts simplifiés).

On est tenté de rejeter la première alternative pour des raisons théoriques. La deuxième alternative, souffre de sa lourdeur. Néanmoins, elle a déjà été traitée d'une façon relativement élégante [1]. La troisième alternative offre un cadre restrictif mais cohérent; de plus, la grande majorité des problèmes de classifications considèrent une valeur de bonne réponse et une valeur de mauvaise réponse.

Cas multi-coûts

La technique parfois nommée "Empirical Risk Minimization" est basée sur la formulation de fonctions de coût approchant le risque moyen. Ces fonctions prennent la forme:

$$J(\mathbf{x}, \mathbf{w}) = J(\tilde{\mathbf{x}}, k, \mathbf{w}) = j_k(s_1(\tilde{\mathbf{x}},\mathbf{w}),\ldots s_K(\tilde{\mathbf{x}},\mathbf{w})) \qquad (3\text{-}20a)$$

Par exemple [1]:

$$J(\mathbf{x}, \mathbf{w}) = J(\tilde{\mathbf{x}}, k, \mathbf{w}) = j (-s_k(\tilde{\mathbf{x}},\mathbf{w}) + \left(\frac{1}{K\text{-}1} \sum_{l \neq k} s_l(\tilde{\mathbf{x}},\mathbf{w})^{-\eta} \right)^{-\frac{1}{\eta}}) \qquad (3\text{-}20b)$$

Quand η tend vers l'infini, $J(\tilde{\mathbf{x}}, k, \mathbf{w})$ tend vers $j(-s_k(\tilde{\mathbf{x}},\mathbf{w}) + s_g(\tilde{\mathbf{x}},\mathbf{w}))$.

Coûts simplifiés

Quand chaque classe ne peut donner lieu qu'à deux coûts (i.e. un coût J_{k+} pour les classifications correctes et un coût J_{k-} pour les classifications incorrectes), il est facile de fournir une expression de fonction de coût local continue. On pose:

---

1 **Katagari S., Lee C.H., Huang B.H.:** *New discriminative training algorithms based on the generalized probabilistic decscent method* - NNSP, pp 299-308 (1991)

$$J_k \triangleq (J_{k+} - J_{k-}) \quad (3\text{-}21)$$

l'écart entre le coût d'une classification correcte J_{k+} et J_{k-} celui d'une classification incorrecte.

On peut alors proposer des fonctions continues de la forme:

$$J_\delta(k,\tilde{\mathbf{x}},\mathbf{w})) = J_k \bullet t_\delta (s_e(\tilde{\mathbf{x}},\mathbf{w}) , s_g(\tilde{\mathbf{x}},\mathbf{w})) \quad (3\text{-}22)$$

où t est une fonction arbitraire "à seuil mou réglable". On exige simplement que t vérifie certaines contraintes, par exemple:

$t_\delta(s_e,s_g)$ définie pour $s_e \geq s_g$ (3-23a)

$t_\delta(s_e,s_g) \in [0,1]$ (3-23b)

$\Delta_\delta(s_g) \triangleq \{s_e \text{ tq: } 0< t_\delta(s_e,s_g) <1\} : \forall\, s_g : \Delta_\delta(s_g)$ borné et $\Delta_\delta(s_g) \underset{\delta \to 0}{\to} \varnothing$ (3-23c)

$t(s_e,s_g)$ est croissante avec s_e et décroissante avec s_g. (3-23d)

Plus le coefficient δ est petit, plus l'approximation est fine mais moins d'exemples mal classifiés engendreront une adaptation. Quand les scores sont proches le coût est faible. Lors de l'apprentissage, au fur et à mesure de la convergence, on fera décroître le paramètre contrôlé δ vers 0. Ainsi, dans un premier temps un grand nombre d'exemples engendreront une adaptation. Au fur et à mesure que celle-ci s'avérera en opposition avec les autres adaptations, seuls les exemples semblant demander une adaptation plus restreinte pour être bien classifiés seront pris en considération.

iv Applications

Nombreuses sont les règles de décisions qui prennent naturellement la forme (3-16). Il est alors possible d'employer des fonctions de coût de la forme (3-22) qui approche le risque local.

Les fonctions de coût globales qui découlent sont différentiables et approchent le risque moyen. Elles restent cependant irrégulières et elles nécessitent quelques précautions d'utilisation, sous peine que les procédures d'optimisations ne conduisent à des minima locaux de mauvaise qualité.

Nous décrivons succintement dans les lignes qui suivent quelques formes de règles de décision dont nous détaillerons les représentants au § 4 et § 5.

Règle du plus proche voisin

Quand chaque score s_l est de la forme:

$$s_l(\tilde{\mathbf{x}},\mathbf{W}) = s_l(\tilde{\mathbf{x}},\tilde{\mathbf{w}}_1,\ldots,\tilde{\mathbf{w}}_M) = d(\tilde{\mathbf{w}}_l,\tilde{\mathbf{x}}) \quad (3\text{-}24)$$

on parle de références pour les paramètres $\tilde{\mathbf{w}}_l$. La fonction d est typiquement une métrique.

Ainsi, la règle de décision du plus proche voisin (cf. § 4.2.2) est de la forme (3-24). Comme nous le verrons, elle peut être associée à des fonctions de coûts de la forme (3-22) telles que LVQ2+ (cf. § 4.2.3.1) et DP-LVQ2+ (cf. § 5.2.1.2).

Références et codages

Quand chaque score s_l est de la forme:

$$s_l(\tilde{\mathbf{x}},\mathbf{W}) = s_l(\tilde{\mathbf{x}},\mathbf{w}',\tilde{\mathbf{w}}_1,\ldots,\tilde{\mathbf{w}}_M) = d(\tilde{\mathbf{w}}_l,f(\tilde{\mathbf{x}},\mathbf{w}'))) \qquad (3\text{-}24)$$

on parle de codage pour la fonction f et de références pour les paramètres $\tilde{\mathbf{w}}_l$. La fonction d est typiquement une métrique.

En exigeant que d soit continue, différentiable et qu'elle possède au plus un extremum pour chaque $f(\tilde{\mathbf{x}},\mathbf{w}_i)$, on a la garantie qu'un chemin continu existe permettant d'apprendre par descente de gradient à classifier correctement un exemple fixé.

Ainsi, la règle de décision des systèmes modulaires MLP- plus proche voisin (cf. § 4.4.2) est de la forme (3-24). Comme nous le verrons, elle peut être associée à des fonctions de coûts de la forme (3-22) telles que MLP-LVQ2+ (cf. § 4.4.2.3) et MLP-DP-LVQ2+ (cf. § 5.5.1).

v Conclusion

Nous avons proposé une méthode permettant d'approcher le risque local et donc le risque moyen dans le cas où le coût d'une classification ne prend que deux valeurs pour une classe donnée. Il existe une méthode similaire dans le cas multicoût, plus coûteuse algorithmiquement et plus lourde analytiquement.

Pour retrouver le cadre rassurant de la théorie de l'optimisation on est amené à se restreindre au cas où le coût d'une classification ne prend que deux valeurs pour une classe donnée. On est ainsi capable de rendre compte de l'importance de bien reconnaître certaines classes, sans avoir la possibilité de nuancer le coût de la classification autrement que par une valeur booléenne correct / incorrect.

Ceci conclu notre étude de l'optimisation directe d'une approximation différentiable du risque moyen. Une autre technique d'apprentissage paramétrique de système de classification de système de classification est celles des modèles probabilistes.

3.3 Systèmes probabilistes

3.3.1 Estimation des vraisemblances conditionnelles

3.3.1.1 Généralités

i Apprentissage des modèles producteurs par estimation des vraisemblances conditionnelles

A présent, nous cherchons à estimer les vraisemblances conditionnelles (ou densités de probabilités conditionnelles) $\mathbf{p}(\tilde{\mathbf{x}}|) = (p(\tilde{\mathbf{x}}|k))_{1 \le k \le K}$ et les probabilités *a priori* $\mathbf{P} = P(k)_{1 \le k \le K}$.

Pour cela, on peut faire appel à des fonctions paramétriques. Par analogie avec les quantités estimées, nous les noterons $\mathbf{p_w}(\tilde{\mathbf{x}}|) = (p_{\mathbf{w}}(\tilde{\mathbf{x}}|k))_{1 \le k \le K}$ et $\mathbf{P} = P_{\mathbf{w}}(k)$. Pour leur apprentissage, on emploie souvent des techniques d'optimisation.

Ces systèmes sont généralement des modèles producteurs, en ce qu'ils peuvent être ensuite utilisés pour des tâches de production (fabrication de données synthétiques). Cependant, leur principal usage porte sur les tâches de reconnaissance (identification du modèle ayant la plus forte probabilité de produire la forme étudiée).

Le plus souvent, on impose que les fonctions paramétriques vérifient:

$$\int_{\mathbf{X}_k} p_{\mathbf{w}}(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} = 1 \quad \text{et} \quad \sum_{k=1}^{K} P_{\mathbf{w}}(k) = 1 \tag{3-25}$$

Comme nous l'avons évoqué précédemment (cf. § 2.2.5), on peut imposer de tels jeux de contraintes à chaque étape de l'apprentissage ou bien se contenter de l'imposer pour la configuration obtenue à l'issue de l'apprentissage, par exemple à l'aide de la méthode des multiplicateurs lagrangiens.

Bien des problèmes de classification permettent une évaluation satisfaisante des probabilités *a priori* $P(k)$ par simple comptage. Ceci explique que certaines méthodes portent l'exclusivité de leur attention sur l'estimation des vraisemblances conditionnelles p. C'est le cas de la majorité des méthodes de reconnaissance des formes statistiques.

Cependant, l'estimation des probabilités a priori $P(k)$ n'est pas toujours triviale, notamment quand il y a un nombre considérable de classes. La reconnaissance de la parole fait parfois

intervenir l'ensemble des séquences de phonèmes possibles: il n'est plus question d'utiliser directement des méthodes de comptage. On emploiera alors d'autres techniques qui estiment à la fois les probabilités *a priori* et les vraisemblances conditionnelles.

Plusieurs méthodes existent pour estimer les vraisemblances conditionnelles [1], [2].

ii Problématique de la généralisation

Un système probabiliste, et notamment un modèle producteur ne doit surtout pas approcher trop finement la distribution empirique, qui, issue généralement d'une collection finie d'exemples, a pour densités de probabilités des vides et des masses de Dirach.

Comme le montre la figure 3.4, utiliser des modèles producteurs à des fins de classification conduit à combiner les erreurs d'estimation des différents modèles.

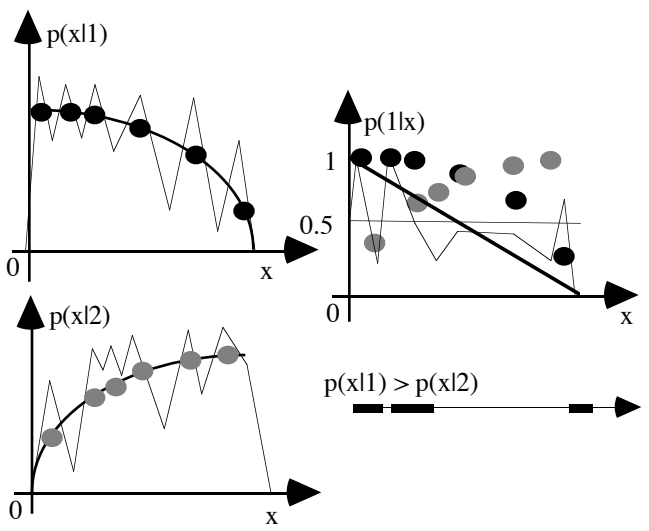


Figure 3.4
Exemple d'utilisation d'un modèle producteur à des fins de classification

A gauche: les vraisemblances conditionnelles (théorique: ligne courbe, estimée: ligne brisée en pointillé). Distribution empirique: points.
A droite: les probabilités a posteriori *(théorique: ligne droite, estimée: ligne brisée).*

Les erreurs d'estimation des distributions se combinent .

Il est intéressant d'utiliser des modèles producteurs, donc de supporter les conséquences de ce biais fâcheux, quand:

- on désire effectuer un apprentissage non-discriminant. Un tel apprentissage permet l'ajout incrémental de classes. Il permet parfois également des tâches autres que la classification, telles que des compressions…

---

1 **Csizar I., Tusnady G.:** *Information geometry and alternating minimization procedure* - Preprint n° 35, Math-Inst. Hungar, Acad. Sci., Budapest (1983)

2 **Ephraim Y., Rabiner L.R.:** *On the Relation Between Modeling Approaches for Information Sources* - S1.2 pp 24-27, proceedings of IEEE-ICASSP (1988)

- l'utilisation de modèles producteurs permet de mettre en valeur des informations *a priori* constituant un cadre utilement restrictif. C'est le cas des modèles de Markov cachés, dédiés à la modélisation de phénomènes temporels. D'autres modèles tels les fonctions à bases radiales (radial basis functions ou RBF) ou les modèles gaussiens (approximation des densités de probabilités par des sommes finies de gaussiennes) sont moins spécifiques; cependant, on peut réduire ce problème en contrôlant leur capacité d'apprentissage par le nombre de gaussiennes, réglé par exemple en fonction de tests de validation croisés.

Des techniques d'appauvrissement de la capacité des systèmes d'apprentissage permettent également de traiter la problématique de la généralisation. C'est le cas notamment de techniques inspirées de la théorie de la régularisation (cf. § 4.1.2.3).

iii Unicité de la décomposition des densités de probabilités en probabilités a priori *et vraisemblances conditionnelles*

Les propriétés:

$$\int_{\mathbf{X}^k} p_{\mathbf{w}}(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} = 1 \quad \text{et} \quad \int_{\mathbf{X}^k} p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} = 1 \tag{3-26}$$

nous permettent de déduire que pour toute classe k, on a l'équivalence:

$$P(k)\, p(\tilde{\mathbf{x}}|k) \equiv P_{\mathbf{w}}(k)\, p_{\mathbf{w}}(\tilde{\mathbf{x}}|k) \quad \Leftrightarrow \quad \begin{cases} P(k) = P_{\mathbf{w}}(k) \\ p(\tilde{\mathbf{x}}|k) \equiv p_{\mathbf{w}}(\tilde{\mathbf{x}}|k) \end{cases} . \tag{3-27}$$

En d'autres termes, il y a unicité de la décomposition des probabilités conjonctives en probabilités *a priori* et probabilités conditionnelles.

iv Conséquence de la continuité des critères optimisés

Les procédés visant à approcher les vraisemblances conditionnelles $\mathbf{p}(\tilde{\mathbf{x}}|)$ et probabilités *a priori* $\mathbf{P}$ consistent en général à optimiser un critère qu'elles seules minimisent absolument: un tel critère ne sera minimisé que par un modèle produisant exactement les vraisemblance conditionnelles $\mathbf{p}(\tilde{\mathbf{x}}|)$ et probabilités *a priori* $\mathbf{P}$. On espère alors obtenir en l'optimisant ce critère un minimum local proche du minimum absolu.

En effet, la famille paramétrique d'estimateurs qu'on utilise ne contient pas, en général, les vraisemblances conditionnelles. La question que l'on peut se poser est de savoir si une minimisation dans une telle famille de fonctions conduit à une approximation des

vraisemblances conditionnelles. En pratique, la réponse à cette question est souvent positive: une approximation suffisamment fine de la valeur minimale du critère provient alors d'une approximation fine des probabilités conditionnelles. Cette propriété est fausse dans le cas général; la figure 3.5 en montre un contre-exemple.

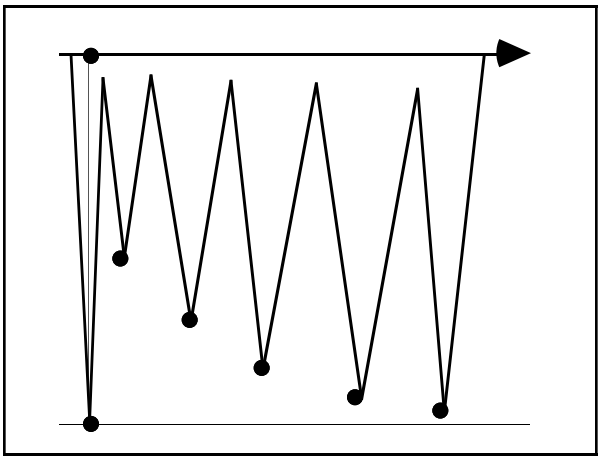


Figure 3.5
Approche d'un minimum global par une suite de minima locaux

Bien qu'en pratique ce ne soit pas un phénomène fréquent, l'optimisation d'un critère peut conduire à une configuration de paramètres fort éloignée de la configuration donnant l'optimum absolu. Cette propriété théorique reste gênante quelque soit la qualité du minimum local.

L'approximation fine des vraisemblances est théoriquement possible si (1) la famille paramétrique est suffisamment riche, et si (2) la méthode d'optimisation est sûre. Hélas, en général, les méthodes d'optimisation dont on dispose sont des algorithmes de descente de gradient qui ne permettent d'obtenir que des minima locaux. En général, la famille de fonctions paramétriques que l'on choisi possède des minima locaux: l'algorithme d'optimisation n'approche plus véritablement les probabilités *a priori*; or, la classification qui s'ensuit est en principe fondée sur cet espoir.

On peut alors se demander quelle confiance il faut accorder à des optima locaux (au sein d'un ensemble de fonctions) d'un critère dont on sait simplement que son optimum absolu (dans l'espace de toutes les fonctions) correspond aux vraisemblances de la distribution empirique…

A défaut de réponse théorique, l'expérience montre que l'optimisation de ces critères conduit à des solutions de bonne qualité. Les méthodes de classification basées sur l'estimation des vraisemblances conditionnelles sont parmi les plus efficaces et les plus populaires, notamment dans le traitement de données temporelles avec les modèles de Markov cachés (reconnaissance de la parole, de signal sonar,…).

3.3.1.2 Estimateur maximisant la vraisemblance globale (MLE)

La méthode la plus connue et la plus employée pour estimer les vraisemblances conditionnelles consiste à maximiser au sein d'une famille d'estimateurs la vraisemblance de l'échantillon

global (constitué d'un ensemble fini d'exemples) dont on dispose. Elle est généralement connue sous son nom anglo-saxon: Maximum Likelihood Estimator (MLE).

i Vraisemblance globale

Quand on dispose d'un échantillon constitué d'un ensemble fini d'exemples indépendants $(\tilde{\mathbf{x}}_i, k_i)$, on peut tenter d'en maximiser la vraisemblance estimée $p_{\mathbf{w}}$:

$$p_{\mathbf{w}}(\mathbb{X}) = p_{\mathbf{w}}(\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_I) = \prod_{i=1}^{I} p_{\mathbf{w}}(\tilde{\mathbf{x}}_i, k_i) = \prod_{i=1}^{I} p_{\mathbf{w}}(\tilde{\mathbf{x}}_i | k_i)\, P_{\mathbf{w}}(k_i) \tag{3-28}$$

On peut indifféremment optimiser cette expression ou son logarithme. On pose alors comme critère global:

$$\begin{aligned} \hat{C}^{mle}(\mathbf{p}_{\mathbf{w}}, \mathbf{P}_{\mathbf{w}}) &= \ln p_{\mathbf{w}}(\mathbb{X}) = \ln \prod_{i=1}^{I} p_{\mathbf{w}}(\tilde{\mathbf{x}}_i | k_i)\,)\, P_{\mathbf{w}}(k_i) \\ &= \sum_{i=1}^{I} \ln P_{k_i}(\mathbf{w}) + \sum_{i=1}^{I} \ln p_{\mathbf{w}}(\tilde{\mathbf{x}}_i | k_i) \end{aligned} \tag{3-29}$$

Ce critère global est en fait l'estimation du critère total:

$$\begin{aligned} C^{mle}(\mathbf{p}_{\bullet \mathbf{w}}, \mathbf{P}_{\mathbf{w}}) &= \int_{\mathbb{X}} \ln p_{\mathbf{w}}(\tilde{\mathbf{x}}, k)\, p(\mathbf{x})\, d\mathbf{x} \\ &= \sum_{k=1}^{K} P(k) \int_{\mathbb{X}_k} \ln p_{\mathbf{w}}(\tilde{\mathbf{x}} | k)\, p(\tilde{\mathbf{x}} | k)\, d\tilde{\mathbf{x}} \\ &+ \sum_{k=1}^{K} P(k) \ln P_{\mathbf{w}}(k) \end{aligned} \tag{3-30}$$

ii Le critère C^{mle} est maximisé par les densités de probabilités conditionnelles

En utilisant les propriétés:

$$\int \ln \frac{g(x)}{f(x)} f(x)\, dx \;\leq\; \int \left(\frac{g(x)}{f(x)} - 1 \right) f(x)\, dx \;=\; \int g(x)dx - \int f(x)dx \;, \tag{3-31a}$$

$$1 = \int_{\mathbb{X}} p_{\mathbf{w}}(\tilde{\mathbf{x}}, k)\, d\mathbf{x} \;=\; \int_{\mathbb{X}} p(\tilde{\mathbf{x}}, k)\, d\mathbf{x} \tag{3-31a}$$

on déduit que:

$$\int_{\mathcal{X}} \ln\, p_{\mathbf{w}}(\tilde{\mathbf{x}},k)\; p(\tilde{\mathbf{x}},k)\; d\mathbf{x} \;\le\; \int_{\mathcal{X}} \ln\, p(\tilde{\mathbf{x}},k)\; p(\tilde{\mathbf{x}},k)\; d\mathbf{x}\ . \qquad (3\text{-}32)$$

En d'autre termes, le critère $C^{mle}(\mathbf{p}_{\bullet\mathbf{w}},\mathbf{P}_{\mathbf{w}})$ est majoré par $C^{mle}(\mathbf{p}_{\bullet},\mathbf{P})$.

Quand $\mathbf{p}$ se trouve dans $\mathbf{p}_{\bullet\mathbf{w}}$, il existe une configuration de paramètres $\mathbf{w}$ permettant d'atteindre ce maximum absolu. En général, cela n'est pas le cas.

Cependant l'inégalité (3-32) est stricte quand $\mathbf{p}_{\bullet\mathbf{w}}$ est distinct de $\mathbf{p}_{\bullet}$. La propriété du § 3.3.2.ii permet alors de déduire que si des approximations suffisamment fines de $\mathbf{p}_{\bullet}$ se trouvent dans $\mathbf{p}_{\mathbf{w}}$, les optimum globaux du critère MLE se trouveront parmi elles.

iii Apprentissage stochastique et global du critère C^{mle}

La fonctionnelle (qu'on cherche à maximiser) s'écrit:

$$C^{mle}(\mathbf{p}_{\bullet\mathbf{w}},\mathbf{P}_{\mathbf{w}}) \;=\; \int_{\mathcal{X}} \ln\, p_{\mathbf{w}}(\tilde{\mathbf{x}}|k)\; p(\mathbf{x})\; d\mathbf{x}\ ; \qquad (3\text{-}33)$$

on en déduit l'algorithme d'apprentissage stochastique:

$$\mathbf{w}_{n+1} \;=\; \mathbf{w}_n + \Gamma_n\, \nabla_{\mathbf{w}} \ln\, p_{\mathbf{w}}(\tilde{\mathbf{x}}|k) \;=\; \mathbf{w}_n + \Gamma_n \frac{\nabla_{\mathbf{w}_n}\, p_{\mathbf{w}_n}(\tilde{\mathbf{x}}|k)}{p_{\mathbf{w}_n}(\tilde{\mathbf{x}}|k)} \qquad (3\text{-}34)$$

et l'algorithme d'apprentissage global:

$$\mathbf{w}_{n+1} \;=\; \mathbf{w}_n + \Gamma_n \sum \nabla_{\mathbf{w}_n} \ln\, p_{\mathbf{w}_n}(\tilde{\mathbf{x}}|k) \;=\; \mathbf{w}_n + \Gamma_n \sum \frac{\nabla_{\mathbf{w}_n}\, p_{\mathbf{w}_n}(\tilde{\mathbf{x}}|k)}{p_{\mathbf{w}_n}(\tilde{\mathbf{x}}|k)} \qquad (3\text{-}35)$$

Quand les fonctions paramétriques dépendent chacune d'une partie attitrée des paramètres, l'apprentissage peut se faire isolément pour chaque classe. Le système bénéficie d'une certaine souplesse, puisqu'on peut y ajouter des classifications au fur et à mesure des besoins sans devoir le réentrainer. En revanche, l'apprentissage est alors non-discriminant.

Quand les paramètres sont partagés entre les classes, l'apprentissage reste peu discriminant. C'est là un des principaux griefs mentionnés contre cette méthode.

iv Contraintes de normalisation

Il est fréquent que les paramètres soient astreints à respecter certaines contraintes de normalisation. C'est le cas en particulier des probabilités de transitions des modèles de Markov,

dont la somme est astreinte à rester égale à un.

En présence d'une contrainte de normalisation de la forme:

$$\forall\ n : \mathrm{mat}^m(\mathbf{w}_{n+1}) \bullet \mathbf{1} = \mathbf{1} \tag{3-36}$$

où m divise la dimension de $\mathbf{w}_n$. La formule d'adaptation 3-34 devient par projection de $\mathbf{w}_n$ sur l'espace des contraintes:

$$\mathbf{w}_{n+1} = \frac{\mathbf{w}_n + \Gamma_n \sum_{\mathbf{x}} \nabla_{\mathbf{w}_n} \ln p_{\mathbf{w}_n}(\tilde{\mathbf{x}}|k)}{\mathbf{1}_1^m \otimes \left(\mathbf{1} + \mathrm{mat}^m(\Gamma_n \sum_{\mathbf{x}} \nabla_{\mathbf{w}_n} \ln p_{\mathbf{w}_n}(\tilde{\mathbf{x}}|k)) \bullet \mathbf{1}\right)} , \tag{3-37}$$

Cette formule un peu abstraite admet une interprétation simple: chaque contrainte de normalisation élémentaire ne concerne que quelques coefficients de $\mathbf{w}$; on découpe donc les vecteurs en tronçons sur chacun desquels s'applique une normalisation élémentaire; puis on reconstitue les vecteurs selon leur dimension d'origine.

L'équation (3-37) est de la forme:

$$\mathbf{w}_{n+1} = \frac{\mathbf{w}_n + \Gamma_n \nabla_{\mathbf{w}_n} l(\mathbf{w}_n)}{\mathbf{1}_1^m \otimes \left(\mathbf{1} + \mathrm{mat}^m(\Gamma_n \nabla_{\mathbf{w}_n} l(\mathbf{w}_n)) \bullet \mathbf{1}\right)} . \tag{3-38}$$

Si l est une fonction strictement croissante d'un polynôme à coefficient positifs et si le vecteur Γ_n est colinéaire avec $\mathbf{w}_n$, alors on a la garantie que $l(\mathbf{w}_{n+1}) < l(\mathbf{w}_n)$.

Lorsque la norme du vecteur Γ_n tend vers l'infini, cette inégalité reste stricte [1], [2]. La formule (3-37), qui se simplifie, représente alors une forme générale de l'algorithme de Baum [3], qui est un des algorithmes les plus populaires pour optimiser les modèles de Markov cachés.

---

1 **Soules G.W.:** *A gradient derivation of PTAH* - IDA-CRD Tech Report N°80328, (1977)

2 **Bottou L.:** *Une approche théorique de l'Apprentissage Connexionniste: Applications à la reconnaissance de la Parole* -Thèse de doctorat, Université Paris XI, Orsay , pp 217 (1991)

3 **Baum L. E., Petrie T., Soules G., Weiss N.:** *A maximisation technique occuring in the stastistical analysis of probabilistic functions of Markov Chains* - Annals of mathematic statistics, vol 41, n°1, pp 164-171 (1970).

v Minimisation d'une forme simple de l'information discriminante (MDI-)

Un forme simple de l'information discriminante permettant de distinguer un jeu de densités de probabilités conditionnelles $\mathbf{p}_\bullet$ de celui de la distribution $(\mathbf{X},p)$ s'écrit [1]:

$$C^{mdi-}(\mathbf{p}_{\bullet\mathbf{w}},\mathbf{P}_\mathbf{w}) \quad = \quad \int_{\mathbf{X}} \ln \frac{P(k)\ p(\tilde{\mathbf{x}}|k)}{P_\mathbf{w}(k)\ p_\mathbf{w}(\tilde{\mathbf{x}}|k)}\ p(\mathbf{x})\ d\mathbf{x} \tag{3-39}$$

ou encore:

$$C^{mdi-}(\mathbf{p}_{\bullet\mathbf{w}},\mathbf{P}_\mathbf{w}) \ = \ C^{mle}(\mathbf{p}_\bullet,\mathbf{P}) - C^{mle}(\mathbf{p}_{\bullet\mathbf{w}},\mathbf{P}_\mathbf{w}) \tag{3-40}$$

Cette expression indique clairement que ce critère MDI- est équivalent au critère MLE.

3.3.2 Estimation directe des probabilités *a posteriori*

L'estimation des probabilités *a posteriori* peut se faire de manière indirecte à l'aide de modèles producteurs. Cependant, comme le montre la figure 3.4, ceci conduit à combiner les approximations. De plus, l'obtention d'une estimation de $p(\tilde{\mathbf{x}})$ est le plus souvent inutile. On est donc tenté d'estimer directement les probabilités *a posteriori*.

Plusieurs critères populaires permettent d'optimiser directement les probabilités *a posteriori*.

3.3.2.1 Moindre erreur quadratique L_2^2 ou MSE

i Définition

Le critère d'erreur quadratique, ou moindre carré (MSE ou Mean Square Error estimator) est la fonction d'erreur la plus employée pour optimiser les associateurs paramétriques. Pour un exemple $\mathbf{x} = (\tilde{\mathbf{x}},\mathbf{y})$, la fonction de coût qu'elle engendre s'écrit:

$$J(\mathbf{x}) = J(\tilde{\mathbf{x}},\mathbf{y}) = |\ \mathbf{f}(\tilde{\mathbf{x}})-\mathbf{y}\ |^2 \tag{3-41}$$

La fonction de coût globale s'écrit:

$$C^{mse}(\mathbf{f}) = E_{\tilde{\mathbf{x}},\mathbf{y}} \left(|\ \mathbf{f}(\tilde{\mathbf{x}})-\mathbf{y}\ |^2 \right) \tag{3-42}$$

---

1 **Devijner P.A., Kittler J.:** *Pattern recognition: A Statistical Approach* - Printice-Hall International Inc., London (1982)

Ce critère est très général. Il est destiné à traiter des problèmes d'approximation. Il n'est pas spécialement destiné aux problèmes de classification mais l'approximation permet de faire de l'estimation.

Nous allons d'abord étudier la propriété remarquable du critère MSE, qui lui permet d'être très adapté aux problèmes de classification.

ii Les espérances a posteriori *minimisent le critère MSE*

Quand la distribution est sans ambiguïtés, cette fonction de coût est optimisée par l'association parfaite entre les arguments (entrées) $\mathbf{x}$ et les résultats (sorties) $\mathbf{y}$. Quand la distribution est ambiguë (plusieurs résultats possibles pour un même argument), il est intéressant d'étudier quel associateur minimise cette fonction d'erreur.

La fonction de coût globale s'écrit:

$$\begin{aligned} C^{mse}(\mathbf{f}) &= E_{\tilde{\mathbf{x}},\mathbf{y}} \left\{ \mid \mathbf{f}(\tilde{\mathbf{x}}) - \mathbf{y} \mid^2 \right\} \\ &= E_{\tilde{\mathbf{x}}} \left\{ \mathbf{f}^2(\tilde{\mathbf{x}}) \right\} - 2\, E_{\tilde{\mathbf{x}}} \left\{ \mathbf{f}(\tilde{\mathbf{x}})\, E_{\mathbf{y}}\{\mathbf{y}|\tilde{\mathbf{x}}\} \right\} + E_{\tilde{\mathbf{x}}} \left\{ E_{\mathbf{y}}\{\mathbf{y}^2|\tilde{\mathbf{x}}\} \right\} \\ &= E_{\tilde{\mathbf{x}}} \left\{ \mathbf{f}(\tilde{\mathbf{x}}) - E_{\mathbf{y}}\{\mathbf{y}|\tilde{\mathbf{x}}\} \right\}^2 - E_{\tilde{\mathbf{x}}} \left\{ E_{\mathbf{y}}(\mathbf{y}|\tilde{\mathbf{x}})^2 \right\} + E_{\tilde{\mathbf{x}}} \left\{ E_{\mathbf{y}}(\mathbf{y}^2|\tilde{\mathbf{x}}) \right\} \\ &= E_{\tilde{\mathbf{x}}} \left\{ \mathbf{f}(\tilde{\mathbf{x}}) - E_{\mathbf{y}}\{\mathbf{y}|\tilde{\mathbf{x}}\} \right\}^2 + \text{terme indépendant de } \mathbf{f} \end{aligned} \qquad (3\text{-}43)$$

D'où, finalement le seul extremum de l'expression (3-42) :

$$\mathbf{f}^*(\tilde{\mathbf{x}}) = E_{\mathbf{y}}\{ \mathbf{y} \mid \tilde{\mathbf{x}} \} = \bar{\mathbf{y}}_{\tilde{\mathbf{x}}} \qquad (3\text{-}44)$$

Cette propriété remarquable reste cependant théorique. En effet, rien ne prouve que l'associateur optimal peut être engendré par l'associateur paramétrique avec lequel on travaille. De plus, rien ne prouve que l'algorithme d'optimisation que l'on utilise va faire converger l'associateur vers un minimum proche du minimum absolu plutôt que vers un minimum local fort différent.

iii Application du MSE à la classification: les probabilités a posteriori *minimisent le critère MSE*

Dans le cas particulier de la classification, il faut utiliser un codage des sorties qui doivent représenter la classe désirée. Pour cela, on peut utiliser la convention: si $\mathbf{x}$ est de la classe k, alors: $y_k = 1$, et pour $l \neq k$: $y_l = 0$ sinon. La propriété (3-44) s'écrit alors:

$$f_k^*(\tilde{\mathbf{x}}) = p(k|\tilde{\mathbf{x}}) \qquad (3\text{-}45)$$

En d'autres termes, la minimisation de l'erreur quadratique conduit à l'approximation des probabilités *a posteriori*: Ceci conduit à la notation:

$$p_{\mathbf{w}}(k|\tilde{\mathbf{x}}) \triangleq f_k(\tilde{\mathbf{x}})$$

Le critère MSE, appliqué à la classification, prend la forme:

$$C^{mse}(\mathbf{p}_{|\mathbf{w}}) = E_{\tilde{\mathbf{x}},k} \left\{ \; | \; \mathbf{p}_{|\mathbf{w}}(\tilde{\mathbf{x}}) \; - \; \mathbf{1}^k \; |^2 \; \right\} \tag{3-46}$$

Comme nous l'avons déjà dit, cette propriété remarquable reste cependant théorique car l'associateur paramétrique contient rarement les probabilités *a posteriori* et la convergence de l'algorithme d'apprentissage peut fort bien se faire vers un minimum local fort différent. Ce handicap théorique semble être de moindre importance que le handicap similaire du critère MLE-MDI. En effet, l'utilisation des modèles producteurs pour des tâches de classification combine les erreurs d'estimation des différents modèles.

iv Apprentissage du critère MSE

L'algorithme de descente de gradient stochastique du critère MSE s'écrit, dans le cas général:

$$\mathbf{w}_{n+1} \;=\; \mathbf{w}_n - 2\,\Gamma_n \left(\nabla_{\mathbf{w}}\, \mathbf{f}(\tilde{\mathbf{x}})\right) * | \, \mathbf{f}(\tilde{\mathbf{x}}) - \mathbf{y} \, | \tag{3-47a}$$

et dans le cas de la classification évoqué au § iii:

$$\mathbf{w}_{n+1} \;=\; \mathbf{w}_n + 2\,\Gamma_n \left(\nabla_{\mathbf{w}}\, \mathbf{p}_{|\mathbf{w}}(\tilde{\mathbf{x}})\right) * | \, \mathbf{p}_{|\mathbf{w}}(\tilde{\mathbf{x}}) - \mathbf{1}^k \, | \tag{3-47b}$$

L'erreur quadratique est un des critères d'optimisation les plus populaires et efficaces pour les problèmes de classification.

3.3.2.2 Maximum de Probabilité *a Posteriori* MAP

i Définition

Le critère MAP s'écrit

$$C^{map}(\mathbf{p}_{|\mathbf{w}}) = E\{ \log(p_{\mathbf{w}}(k|\tilde{\mathbf{x}}) \} \tag{3-48}$$

Ce critère ressemble beaucoup au critère MMI. Dans le cas simple où les probabilités *a priori* sont toutes égales, ces critères sont équivalents.

ii Le critère MAP est maximisé par les probabilités a posteriori

De même que pour le critère MLE, on peut établir une inégalité de la forme (3-31):

$$\sum_k \ln \frac{p_{\mathbf{w}}(k|\tilde{\mathbf{x}})}{p(k|\tilde{\mathbf{x}})} p(k|\tilde{\mathbf{x}}) \leq \sum_k p_{\mathbf{w}}(k|\tilde{\mathbf{x}}) - \sum_k p(k|\tilde{\mathbf{x}}) = 0 \qquad (3\text{-}49a)$$

Cette inégalité est stricte dés que $\mathbf{p}_{|\mathbf{w}}$ et $\mathbf{p}_{|}$ diffèrent. Sur la distribution empirique, cette inégalité devient:

$$\int \ln p_{\mathbf{w}}(k|\tilde{\mathbf{x}})\, p(k|\tilde{\mathbf{x}})\, d\mathbf{x} \leq \int \ln p(k|\tilde{\mathbf{x}})\, p(k|\tilde{\mathbf{x}})\, d\mathbf{x} \qquad (3\text{-}49b)$$

Cette inégalité est stricte dés que $\mathbf{p}_{|}$ et $\mathbf{p}_{|\mathbf{w}}$ ne sont pas égales presque partout. En d'autres termes $C^{map}(\mathbf{p}_{|\mathbf{w}})$ est majoré par $C^{map}(\mathbf{p}_{|})$.

iii Apprentissage du critère MAP

L'algorithme d'apprentissage du critère MAP s'écrit:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \Gamma_n \frac{\nabla_{\mathbf{w}}\, p_{\mathbf{w}}(\, k \mid \tilde{\mathbf{x}}\,)}{p_{\mathbf{w}}(\, k \mid \tilde{\mathbf{x}}\,)} \qquad (3\text{-}50)$$

3.3.2.3 Information mutuelle maximale (MMI)

i Information mutuelle

L'information mutuelle liant une classe k et une observation $\tilde{\mathbf{x}}$ s'écrit:

$$\ln \frac{p(k|\tilde{\mathbf{x}})}{P(k)} = \ln \frac{p(\tilde{\mathbf{x}},k)}{P(k)\, p(\tilde{\mathbf{x}})} = \ln \frac{p(\tilde{\mathbf{x}}|k)}{p(\tilde{\mathbf{x}})} = \ln \frac{p(\tilde{\mathbf{x}}|k)}{\sum_{s=1}^{K} P(s)\, p(\tilde{\mathbf{x}}|s)} \qquad (3\text{-}51)$$

Comme on peut le constater, le critère MMI est très proche du critère MAP.

L'estimation du critère d'information mutuelle relatif à la distribution s'écrit alors:

$$C^{mmi}(\mathbf{p}_{|\mathbf{w}}, \mathbf{P}_{\mathbf{w}}) = \sum_{k=1}^{K} P(k) \int_{\mathbf{X}^k} \ln \frac{p_{\mathbf{w}}(\tilde{\mathbf{x}}|k)}{\sum_{s=1}^{K} p_{\mathbf{w}}(\tilde{\mathbf{x}}|s) P_{\mathbf{w}}(s)}\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}} \qquad (3\text{-}52)$$

Ce critère est plus complexe que le critère MLE. Il a été introduit pour ses qualités discriminantes: comme on peut le voir, l'estimateur de densité de probabilité de chaque classe y est systématiquement rapporté à la somme des estimateurs de densités de probabilités. En fait,

le processus d'optimisation va chercher à maximiser, pour chaque exemple $(\tilde{\mathbf{x}},k)$ l'estimation des probabilité *a posteriori* de la classe k.

Le critère MMI est donc un critère intermédiaire entre l'univers des modèles producteurs et l'univers des estimateurs directs de probabilité *a posteriori*. Dans le cas simple où les probabilités *a priori* sont toutes égales, ce critère est équivalent au critère MAP (cf. § 3.3.2.2).

ii Les probabilités a posteriori *maximisent le critère MMI*

Nous proposons ici une ébauche de démonstration de la maximisation du critère MMI par $(\mathbf{p}_{|},\mathbf{P})$. La méthode que nous employons est lourde. Nous la proposons pour des raisons pédagogiques car elle a un caractère très général. Elle repose sur une formulation particulière de la condition de minimalité de (3-52).

i Reformulation du critère d'optimalité

Nous cherchons à montrer que $(\mathbf{p}_{\bullet},\mathbf{P})$ maximise le critère MMI au sein de l'ensemble des couples $(\hat{\mathbf{p}}_{\bullet},\hat{\mathbf{P}})$ possibles (cette notation ne fait pas référence à l'estimation au sens strict).

On ne peut facilement engendrer ou même approcher paramétriquement l'ensemble des $(\hat{\mathbf{p}}_{\bullet},\hat{\mathbf{P}})$ possibles tout en respectant leurs caractéristiques topologiques. En revanche il est possible de mettre en évidence des transformations aussi locales et infinitésimales que voulu, respectant les contraintes propres aux probabilités (sommes égales à 1), suffisamment régulières et susceptibles d'engendrer l'ensemble des $(\hat{\mathbf{p}}_{\bullet},\hat{\mathbf{P}})$.

Nous proposons donc de transformer une configuration $(\hat{\mathbf{p}}_{\bullet},\hat{\mathbf{P}})$ à l'aide des deux transformations élémentaires suivantes:

- ajouter-retrancher à $\hat{p}(\bullet|k)$ (k arbitrairement fixé) deux bosses paramétriques positives $b_1(\bullet)$ et $b_2(\bullet)$, de diamètres, de formes et de centres arbitraires, et dont les intégrales sont égales à 1, en les pondérant par un coefficient w_1;

$$\begin{cases} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|l) = \hat{p}_0(\tilde{\mathbf{x}}|l) + w_1\, b_1(\tilde{\mathbf{x}}) - w_1\, b_2(\tilde{\mathbf{x}}) \\ \forall\ m \neq l : \quad \hat{p}_{w_1}(\tilde{\mathbf{x}}|m) = \hat{p}_0(\tilde{\mathbf{x}}|m) \end{cases} \tag{3-53}$$

Dans la suite, on notera:

$$b_{\mathbf{w}}(\tilde{\mathbf{x}}) = w_1\, b_1(\tilde{\mathbf{x}}) - w_1\, b_2(\tilde{\mathbf{x}}) \tag{3-54}$$

Les densités de probabilités doivent rester entre 0 et 1 malgré cette transformation. Ceci est la seule contrainte supplémentaire restreignant l'arbitraire du choix de w_1, $b_1(\bullet)$ et

$b_2(\bullet)$.

- ajouter w_2 à $\hat{P}_i$ et retrancher w_2 à $\hat{P}_j$ (i et j arbitrairement fixés). On notera

$$\hat{\mathbf{P}}_{\mathbf{w}} = \hat{\mathbf{P}} + w_2\, \mathbf{1}^i - w_2\, \mathbf{1}^j$$

Les probabilités doivent rester entre 0 et 1 malgré cette transformation. Ceci est la seule contrainte restreignant l'arbitraire du choix de w_2, i et j.

Il est aisé de vérifier que les combinaisons de transformations élémentaires de ce type permettent d'approcher (et en réalité elles engendrent) l'ensemble des configurations $(\hat{\mathbf{p}}_\bullet, \hat{\mathbf{P}})$.

Dans l'espace des estimateurs les extrema locaux de C^{mmi} sont caractérisés par la conjonction des deux propriétés relatives à ces transformations élémentaires autour de $\mathbf{w} = (w1,w2) = \mathbf{0}$.

- une dérivée de C^{mmi} par rapport à w_1 nulle en $\mathbf{w} = \mathbf{0}$ quelle que soit les centres des bosses, leur diamètre et la classe l dont la densité de probabilité estimée reçoit l'ajout des bosses [première condition],
- une dérivée de C^{mmi} par rapport à w_2 nulle en $\mathbf{w} = \mathbf{0}$ [deuxième condition].

Nous allons à présent examiner successivement ces deux conditions.

ii Analyse de la première condition

Dans ce cas, la première condition de maximalité 3-52 (portant sur w_1) s'écrit:

$$\nabla_{w_1} C^{mmi}(\hat{\mathbf{p}}_{\bullet\mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}})_{\mathbf{w}=0} = \mathbf{0} \qquad (3\text{-}55)$$

$$= \sum_{k=1}^{K} P(k) \int_{\mathbf{X}^k} (\nabla_{w_1} \ln \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|k))\, p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}}$$

$$- \sum_{k=1}^{K} P(k) \int_{\mathbf{X}^k} (\nabla_{w_1} \ln \sum_{s=1}^{K} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|s) \hat{P}_{\mathbf{w}}(s)) p(\tilde{\mathbf{x}}|k)\, d\tilde{\mathbf{x}}$$

$$= P(l) \int_{\mathbf{X}^l} \frac{b(\tilde{\mathbf{x}})}{\hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|l)}\, p(\tilde{\mathbf{x}}|l)\, d\tilde{\mathbf{x}}$$

$$- \sum_{k=1}^{K} P(k) \int_{\mathbf{X}_k} \frac{b(\tilde{\mathbf{x}})\hat{P}_{\mathbf{w}}(l)}{\sum_{s=1}^{K} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|s)\hat{P}_{\mathbf{w}}(s)} p(\tilde{\mathbf{x}}|k)d\tilde{\mathbf{x}}$$

$$\nabla_{w_1} C^{mmi}(\hat{\mathbf{p}}_{\bullet \mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}})_{\mathbf{w}=0} = \mathbf{0} = P(l) \int_{\mathbf{X}_l} \left(\frac{p(\tilde{\mathbf{x}}|l)}{\hat{p}(\tilde{\mathbf{x}}|l)} - \frac{\sum_{k=1}^{K} p(\tilde{\mathbf{x}}|k)P(k)}{\sum_{s=1}^{K} \hat{p}(\tilde{\mathbf{x}}|s)\hat{P}(s)} \right) b(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \qquad (3\text{-}56)$$

Comme on peut choisir la classe l, le centre et la forme des bosses, et comme les probabilités conditionnelles sont supposées suffisamment régulières (continues), les bosses peuvent tendre à jouer vis-à-vis du terme principal de (3-46) le rôle de masses de Dirach arbitraires. On en déduit:

$$\forall\, l : \quad \frac{p(\tilde{\mathbf{x}}|l)}{\hat{p}(\tilde{\mathbf{x}}|l)} \equiv \frac{\sum_{k=1}^{K} p(\tilde{\mathbf{x}}|k)P(k)}{\sum_{s=1}^{K} \hat{p}(\tilde{\mathbf{x}}|s)\hat{P}(s)} + D \qquad (3\text{-}57)$$

où D est une constante. On obtient alors un système d'équations caractérisant la première condition de minimalité:

$$\begin{cases} \forall\ i,j : \quad \dfrac{p(\tilde{\mathbf{x}}|i)}{\hat{p}(\tilde{\mathbf{x}}|i)} \equiv \dfrac{p(\tilde{\mathbf{x}}|j)}{\hat{p}(\tilde{\mathbf{x}}|j)} \triangleq f(\tilde{\mathbf{x}}) \\ \dfrac{\sum_{k=1}^{K} p(\tilde{\mathbf{x}}|k)P(k)}{\sum_{s=1}^{K} \hat{p}(\tilde{\mathbf{x}}|s)\hat{P}(s)} = \dfrac{p(\tilde{\mathbf{x}})}{\hat{p}(\tilde{\mathbf{x}})} = f(\tilde{\mathbf{x}}) - D \end{cases} \qquad (3\text{-}58)$$

Ce système d'équation peut se simplifier; en effet:

$$\int_{\mathbf{X}} p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} = \int_{\mathbf{X}} \hat{p}(\tilde{\mathbf{x}})\, (\, f(\tilde{\mathbf{x}}) - D\,)\, d\tilde{\mathbf{x}} \qquad (3\text{-}59)$$

$$1 + D = \int_{\mathbf{X}} \hat{p}(\tilde{\mathbf{x}})\, f(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} = \int_{\mathbf{X}} \sum_{s=1}^{K} \hat{p}(\tilde{\mathbf{x}}|s)\hat{P}(s)\, f(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \qquad (3\text{-}60)$$

$$= \sum_{s=1}^{K} \hat{P}(s) \int_{\mathbf{X}} p(\tilde{\mathbf{x}}|s)\, d\tilde{\mathbf{x}} \qquad = 1$$

D'où la nullité de D. Alors les équations 3-58 impliquent:

$$\sum_{k=1}^{K} p(\tilde{\mathbf{x}}|k)P(k) = \sum_{s=1}^{K} \left(\hat{p}(\tilde{\mathbf{x}}|s)\, \hat{P}(s)\, f(\tilde{\mathbf{x}}) \right) = \sum_{s=1}^{K} p(\tilde{\mathbf{x}}|s)\, \hat{P}(s) \tag{3-61}$$

$$\sum_{k=1}^{K} p(\tilde{\mathbf{x}}|k)\, (P(k) - \hat{P}(k)) = 0$$

Sauf cas de dégénérescence de la distribution, les vraisemblances conditionnelles ne sont pas uniformément colinéaires sur tout $\mathbf{X}$. On en déduit l'égalité des probabilités *a priori*:

$$\forall\, k :\ P(k) = \hat{P}(k) \tag{3-62a}$$

Alors:

$$\forall\, i,j :\ \frac{p(\tilde{\mathbf{x}}|i)}{\hat{p}(\tilde{\mathbf{x}}|i)} \equiv \frac{p(\tilde{\mathbf{x}}|j)}{\hat{p}(\tilde{\mathbf{x}}|j)} \triangleq f(\tilde{\mathbf{x}}) \tag{3-62b}$$

La première condition de minimalité se réduit alors à l'équation:

$$\boxed{\forall\ k :\quad P(k) = \hat{P}(k) \qquad \forall\ i :\quad p(i \mid \tilde{\mathbf{x}}) = \hat{p}(i \mid \tilde{\mathbf{x}})} \tag{3-63}$$

En d'autres termes, seul le respect des probabilités *a posteriori* permet de vérifier la première condition.

iii Analyse de la deuxième condition

La deuxième condition de minimalité (portant sur w_2) s'écrit:

$$\nabla_{w_2} C^{mmi}(\hat{\mathbf{p}}_{\bullet\mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}})_{\mathbf{w}=0} \qquad = \mathbf{0} \tag{3-64}$$

$$= \nabla_{P_i} C^{mmi}(\hat{\mathbf{p}}_{\mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}}) - \nabla_{P_j} C^{mmi}(\hat{\mathbf{p}}_{\mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}})$$

or:

$$\nabla_{P_i} C^{mmi}(\hat{\mathbf{p}}_{\bullet\mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}})_{\mathbf{w}=0} = - \sum_{k=1}^{K} P(k) \int_{\mathbf{X}^k} \frac{\hat{p}(\tilde{\mathbf{x}}|i)}{\sum_{s=1}^{K} \hat{p}(\tilde{\mathbf{x}}|s)\hat{P}(s)}\, p(\tilde{\mathbf{x}}|k) d\tilde{\mathbf{x}} \tag{3-65}$$

$$= - \int_{\mathbf{X}_k} \hat{p}(\tilde{\mathbf{x}}|i) \frac{\sum_{k=1}^{K} p(\tilde{\mathbf{x}}|k)P(k)}{\sum_{s=1}^{K} \hat{p}(\tilde{\mathbf{x}}|s)\hat{P}(s)} \, d\tilde{\mathbf{x}}$$

quand la première condition de maximalité (3-52) est vérifiée, on obtient:

$$\nabla_{P_i} C^{mmi}(\hat{\mathbf{p}}_{\bullet\mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}})_{\mathbf{w}=0} = - \int_{\mathbf{X}_k} \hat{p}(\tilde{\mathbf{x}}|i) \, f(\tilde{\mathbf{x}}) \, d\tilde{\mathbf{x}}) = - \int_{\mathbf{X}_k} p(\tilde{\mathbf{x}}|i) \, d\tilde{\mathbf{x}} = -1 \qquad (3\text{-}66)$$

Cette égalité est valable pour j comme pour i, d'où

$$\nabla_{w_2} C^{mmi}(\hat{\mathbf{p}}_{\bullet\mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}}) = \nabla_{P_i} C^{mmi}(\hat{\mathbf{p}}_{\bullet}, \hat{\mathbf{P}}) - \nabla_{P_j} C^{mmi}(\hat{\mathbf{p}}_{\bullet}, \hat{\mathbf{P}}) = 0 \qquad (3\text{-}67)$$

En d'autres termes, la première condition d'extrémalité du critère MDI implique la seconde.

iv Synthèse

On obtient finalement le critère nécessaire et suffisant d'optimalité de C^{mmi}:

$$\boxed{\forall\ k : \quad P(k) = \hat{P}(k) \qquad \forall\ \tilde{\mathbf{x}} : \quad p(k \mid \tilde{\mathbf{x}}) = \hat{p}(k \mid \tilde{\mathbf{x}})} \qquad (3\text{-}68)$$

On voit que le critère MMI n'a pas pour objectif de conduire à une estimation complète des vraisemblances conditionnelles contrairement au critère MLE. En fait, de même que le critère MAP, le critère MMI est optimisé par les probabilités *a posteriori*. C'est pourquoi le couple $(\mathbf{p}_{\bullet},\mathbf{P})$ optimise le critère C^{mmi} mais n'est pas le seul.

iii Apprentissage stochastique du critère MMI

Ce critère ressemble beaucoup au critère MAP. Il est beaucoup plus discriminant les critères MLE et MDI. Tous les paramètres interagissent à chaque étape de l'apprentissage.

Le gradient du critère local MMI pour un exemple $(\tilde{\mathbf{x}},k)$ s'écrit:

$$\nabla_{\mathbf{w}} J^{mmi}(\hat{\mathbf{p}}_{\mathbf{w}}, \hat{\mathbf{P}}_{\mathbf{w}}) = \nabla_{\mathbf{w}} \ln \frac{\hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|k)}{\sum_{s=1}^{K} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|s)\hat{P}_{\mathbf{w}}(s)} \qquad (3\text{-}69)$$

$$= \frac{\nabla_{\mathbf{w}} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|k)}{\hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|k)} - \frac{\sum_{s=1}^{K} \nabla_{\mathbf{w}} (\hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|s) \hat{P}_{\mathbf{w}}(s))}{\sum_{s=1}^{K} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|s) \hat{P}_{\mathbf{w}}(s)}$$

d'où l'algorithme d'apprentissage:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \Gamma_n \nabla_{\mathbf{w}} \ln \frac{\hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|k)}{\sum_{s=1}^{K} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|s) \hat{P}_{\mathbf{w}}(s)} \tag{3-70}$$

$$= \mathbf{w}_n + \Gamma_n \left(\frac{\nabla_{\mathbf{w}} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|k)}{\hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|k)} - \frac{\sum_{s=1}^{K} \nabla_{\mathbf{w}} (\hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|s) \hat{P}_{\mathbf{w}}(s))}{\sum_{s=1}^{K} \hat{p}_{\mathbf{w}}(\tilde{\mathbf{x}}|s) \hat{P}_{\mathbf{w}}(s)} \right)$$

Pour ce critère, on ne connaît pas d'algorithme aussi efficace que l'algorithme d'estimation-maximisation pour le critère MLE-MDI, dont l'algorithme de Baum est une version particulière. On en est réduit à utiliser la descente de gradient sous ses formes les plus classiques.

3.3.2.4 Comparaison des critères d'estimation de probabilité *a posteriori*

Les critères MAP et MMI insistent particulièrement sur la bonne estimation de probabilités faibles (faibles dans l'absolu pour MAP, faibles par rapport à la normale *a priori* pour MMI). En effet, pour une probabilité faible, une petite erreur d'estimation est proportionnellement plus importante, et par passage au logarithme de la probabilité estimée, une telle erreur va engendrer un grand coût.

Or l'estimation très précise des probabilités faibles n'est d'aucune utilité pour la classification, qui se heurte à des ambiguïtés entre probabilités fortes.

3.3.2.5 Utilisation de critères d'estimation des probabilités *a posteriori* pour l'optimisation de modèles producteurs

Les modèles producteurs, par nature, permettent d'engendrer une distribution artificielle supposée proche de la distribution empirique ou de la distribution totale. Il est donc naturel de leur appliquer des critères visant à optimiser l'estimation des vraisemblances conditionnelles.

Néanmoins, les critères visant à optimiser l'estimation des probabilités *a posteriori* tels que

MSE, MAP et MMI sont souvent employés pour entraîner des systèmes capables d'engendrer des distributions artificielles, c'est-à-dire des modèles producteurs.

Bien sûr, ces critères conduisent à des systèmes tendant à estimer convenablement les probabilités *a posteriori* $p(k \mid \tilde{\mathbf{x}})$ et non les vraisemblances conditionnelles $p(\tilde{\mathbf{x}} \mid k)$. Les distributions artificielles qu'ils engendrent sont généralement fort éloignées de la distribution empirique ou de la distribution totale. Pour retrouver des modèles producteurs fiables, il est nécessaire d'ajouter une information: la vraissemblance globale $p(\tilde{\mathbf{x}})$.

Aussi, ces critères ne sont utilisés que dans le contexte d'un apprentissage discriminant, c'est-à-dire pour des problèmes de classification où les classes sont déterminées de façon définitives. Ainsi, ils conviennent mal pour entraîner des systèmes d'identification de locuteurs si on désire pouvoir ajouter facilement et rapidement de nouveaux locuteurs au sein du système.

Les critères MMI et MAP sont moins populaires que le critère MLE en raison de la lourdeur des calculs qu'ils imposent: réajustement de tous les paramètres à chaque étape de l'apprentissage, pas d'algorithme d'apprentissage aussi efficace que l'algorithme d'estimation-maximisation. En revanche, ils permettent de traiter de façon discriminante la distribution empirique, ce qui conduit en principe à un surcroît de performance. De plus, seuls certains critères discriminants peuvent entraîner certains systèmes modulaires qui sont susceptibles de dégénérer lors de l'apprentissage.

3.3.3 Conclusion sur les méthodes probabilistes

Plusieurs critères visent à permettre l'estimation des vraisemblances conditionnelles.

Le critère MLE conduit à une modélisation complète de la distribution étudiée. Quand les paramètres appris ne sont pas partagés entre les modèles, ce critère n'est absolument pas discriminant. Le critère MLE peut être optimisé à l'aide de l'algorithme d'estimation-maximisation qui est un cas limite de l'apprentissage par descente de gradient.

Les critères MSE, MMI et MAP sont discriminants. Ils visent à estimer correctement les probabilités *a posteriori*. Ils ne disposent pas d'algorithmes d'optimisation comparables à l'algorithme d'estimation-maximisation utilisés par le MLE et doivent se contenter de descentes de gradient plus ordinaires, fussent-elles stochastiques.

L'optimisation de ces critères est bien évidemment sujette aux phénomènes de minima locaux. On peut se demander quelle confiance il faut accorder à des simili-vraisemblances conditionnelles optima locales (au sein d'un ensemble de fonctions) d'un critère dont on sait simplement que son optimum absolu (dans l'espace de toutes les fonctions) correspond aux

vraisemblances de la distribution empirique… A défaut de réponse théorique, l'expérience montre que l'optimisation de ces critères conduit à des solutions de bonne qualité.

3.4 Conclusion sur les critères d'optimisation pour systèmes de classification

Le risque empirique est une fonction constante par morceaux. Il ne se prête donc pas à optimisation dérivative. Or, la plupart des systèmes d'apprentissage reposent sur des techniques d'optimisation.

Il faut donc employer des techniques alternatives. Les techniques d'optimisation pour systèmes de classification peuvent se ranger grossièrement dans trois catégories: l'optimisation directe d'une approximation différentiable du risque moyen, l'estimation des probabilités *a posteriori*, l'estimation des vraisemblances conditionnelles. Le tableau suivant en résume les principales caractéristiques.

FAMILLE DE METHODES	EXEMPLES DE CRITERES	QUALITES	DEFAUTS
Approximation différentiable du risque empirique	LVQ2+, ERM	• Discriminant • Concentre la capacité du système sur les ambiguïtés	• Irrégularité des fonctions de coût • Restriction à deux coûts par classe pour LVQ2+ • Lourdeur calculatoire des fonctions de coût ERM
Estimation des probabilités *a posteriori*	MSE, MAP, MMI	• Concentre la capacité du système sur la discrimination	• Concentre la capacité du système sur les fortes densités, même sans ambiguïté
Estimation des vraisemblances conditionnelles	MLE	• Non discriminant • Facilité de prise en compte des connaissances *a priori*	• Combinaison des erreurs d'estimation • Concentre la capacité du système sur la modélisation totale de la distribution

Cependant, un critère d'optimisation n'est rien sans son application à l'optimisation d'une famille paramétrique de fonctions. C'est ce que nous étudions dans le chapitre suivant.

4 Quelques systemes utilisant l'information *a posteriori*

4.1 Introduction

A l'occasion du chapitre d'introduction, nous avons brièvement évoqué la grande diversité des méthodes d'exploitation de l'information *a posteriori*. Ce chapitre a pour objet d'en exposer un grand nombre, notamment celles utilisant pour leur ajustement des techniques de descente de gradient, à l'exclusion des méthodes spécialement dédiées au traitement de données temporelles qui seront exposées au chapitre suivant. Ce chapitre accorde une grande importance aux techniques de classification et s'inscrit de façon importante dans la perspective du chapitre précédent qui portait sur les critères d'optimisation pour systèmes de classification.

Le § 4.1 décrit quelques notions fondamentales: supervision, discrimination, adéquation, apprentissage, généralisation, systèmes modulaires et calcul du gradient par rétropropagation. Les § 4.2 et 4.3 et 4.4 font exposent une multitude de systèmes. L'originalité de notre approche nous a conduit à réécrire systématiquement chaque technique d'apprentissage (quitte à les modifier légèrement) sous la forme d'une descente de gradient stochastique. Cette approche procure une vision unifiée, cohérente et synthétique. De plus, elle nous permet d'établir la convergence d'un grand nombre de ces algorithmes, en corollaire du théorème établi au deuxième chapitre.

Le § 4.2 est consacré à la famille des systèmes de plus proches voisinage, en distinguant les métriques, les stratégies de vote et les techniques d'apprentissage. Nous y évoquerons kNN, fenêtres de Parzen, cartes topologiques, distance tangente, kMeans, LVQ…. La fonction de coût LVQ2+ est originale. Le § 4.3 est consacré aux associateurs paramétriques. Nous y évoquerons MLP et RBF. Le § 4.4 porte sur les systèmes modulaires. Le § 4.4.3, est lui aussi original. Il

démontre le phénomène de codage ponctuel qui fait dégénérer notamment de nombreux systèmes MLP-plus proches voisins. Il expose un système modulaire original et efficace: le système MLP-LVQ2+.

4.1.1 Supervision et discrimination

4.1.1.1 Supervision

Superviser un système consiste à lui fournir lors de son développement une information *a posteriori* plus riche que l'information disponible lors de son utilisation opérationnelle. Fournir l'information lexicale associée à l'image d'un texte, fournir la valeur effective d'une consommation qu'il s'agit de prévoir, fournir la direction du vent que l'on cherche à déterminer à partir de données satellites, sont autant d'exemples de supervisions.

Dans la mesure du possible, les systèmes mis au point à l'aide d'informations *a posteriori* sont supervisés. Lors d'un apprentissage, la supervision d'un système se fait en principe par l'intermédiaire de sa fonction d'erreur locale (par exemple les contributions locales aux critères MSE, MAP, MLE étudiés au chapitre précédent), ce qui, à l'issue de l'apprentissage, permet d'extraire la partie opérationnelle du système. La figure 4.1 représente ce principe.

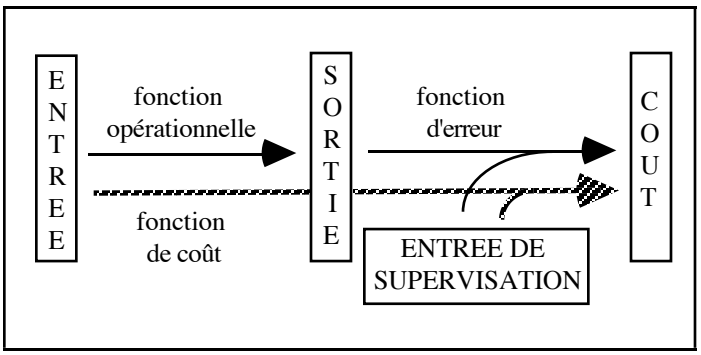


Figure 4.1
Supervision

La supervision consiste à entraîner un système sur des données de nature plus complète que les données opérationnelles.

Il existe des systèmes non supervisés. Ceux-ci ont simplement pour vocation de modéliser certains aspects des données. Leur utilisation est donc plus restreinte. Ils interviennent quand la supervision est trop difficile ou quand elle n'a pas de sens. Ainsi les systèmes de compression d'image sont en principe entraînés de façon non supervisée.

Il est généralement possible bâtir un système de classification supervisé à l'aide d'un algorithme d'apprentissage non-supervisé. Pour cela, il suffit que l'algorithme optimise de façon non-supervisée une mesure d'adéquation avec un modèle paramétrique (ex: optimisation non supervisée des scores du § 3.2). Il suffit de séparer les exemples selon leur classe et de concevoir à l'aide de l'algorithme un modèle par classe. Lors de l'utilisation opérationnelle du

système, on compare les différentes mesures d'adéquation des modèles afin d'effectuer la classification.

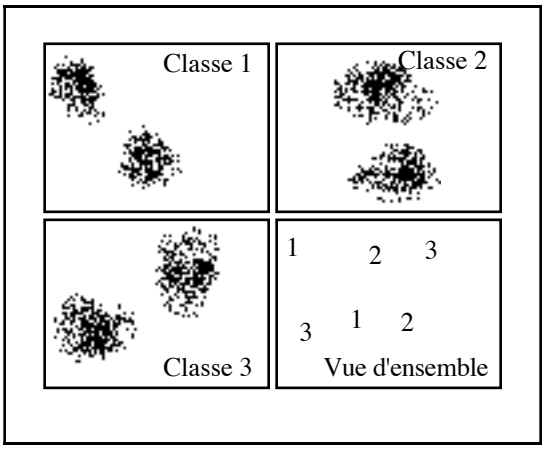


Figure 4.2:
Apprentissage Supervisé non Discriminant

Une technique d'optimisation non supervisée peut être très simplement transformée en une technique de classification supervisée.
Pour cela, il suffit d'appliquer la technique non-supervisée sur les données de chaque classe. La classification opérationnelle consiste à choisir la classe dont le critère d'apprentissage est minimal.

Comme nous allons le voir, un tel système est non-discriminant.

4.1.1.2 Discrimination

La discrimination consiste, dans un problème de classification, à apprendre les classes les unes par opposition aux autres. *A contrario*, modéliser chaque classe sans tenir compte des données des autres classes est non-discriminant.

L'apprentissage discriminant permet de mieux déterminer les frontières de classification optimales. En revanche, l'apprentissage non-discriminant permet l'ajout incrémental de classes.

4.1.2 Les trois problématique des systèmes utilisant l'information *a posteriori*

4.1.2.1 Problématique de l'adéquation: prise en compte de l'information *a priori*

Comme nous l'avons déjà évoqué, la prise en compte de l'information *a posteriori* est un pis-aller, destiné en principe à pallier à un manque de connaissance *a priori*. Le plus souvent, on dispose tout de même de précieuses informations *a priori*. Or, ces deux types d'informations revêtent des formes très différentes. Les prendre en compte au sein d'un même système est donc une tâche essentielle mais difficile, voire intrinsèquement conflictuelle.

Plusieurs constituants d'un système d'apprentissage sont susceptibles d'incorporer des connaissances *a priori*:

- les prétraitements, par exemple les bancs de filtres et les cepstres en parole, les moments en vision, les bruitages et décalages aléatoires…

- la nature même du système, par exemple la métrique d'un système de plus proche voisinage,

- l'architecture du système, par exemple la connectivité d'un Perceptron multicouche (MLP) ou d'un modèle de Markov caché et le partage de paramètres au sein d'un TDNN,

- la fonction de coût de l'apprentissage (i.e. critère d'optimisation), par exemple les normes L1, L2,… en compression de données, critère MAP ou MSE pour un système d'estimation de probabilités *a posteriori*…

- les post-traitements, par exemple la prise en compte de probabilités *a priori*, normalisation des sorties…

4.1.2.2 Problématique de l'apprentissage: optimisation sur les données *a posteriori*

L'apprentissage d'un système est la définition puis l'exécution d'un ensemble de procédures qui vont permettre un comportement optimal au regard de l'information *a posteriori* disponible.

Dans le cadre de l'optimisation, la problématique de l'apprentissage porte donc sur la définition d'une fonction d'erreur, puis d'une procédure d'optimisation. Nous avons déjà évoqué la multiplicité des procédures d'optimisation disponibles. Aujourd'hui, la problématique de l'apprentissage est largement popularisée et dispose d'outils efficaces.

Nous avons déjà largement évoqué la problématique de l'apprentissage lors des chapitres précédents, dans le cadre de la descente de gradient stochastique et notamment avec les classifieurs.

4.1.2.3 Problématique de la généralisation: meta-apprentissage

L'apprentissage se fait sur la distribution empirique, c'est à dire sur un nombre fini d'exemples. Aussi, un système ayant une grande capacité d'apprentissage serait à même d'effectuer un apprentissage "par cœur", produisant une procédure minimisant parfaitement le risque empirique mais se comportant d'une façon aléatoire sur des formes non présentes dans la distribution empirique utilisée lors de l'apprentissage, d'où de piètres performances sur toute autre distribution empirique. La problématique est délicate: dans quelle mesure faut-il être synthétique, dans quelle mesure faut-il être pointilliste ? Ces deux stratégies sont antinomiques

et selon les tâches à résoudre, on serait tenté de privilégier l'une ou l'autre de ces deux approches.

Des méthodes existent visant à assurer une généralisation correcte. Fondamentalement, elles consistent à ajuster la finesse du système d'apprentissage à la quantité d'information *a posteriori* disponible [1]. Quand on dispose d'informations *a priori* sur la tâche à effectuer (invariances spatiales, restriction de l'espace de sortie,...) il est généralement possible d'augmenter la variété des exemples de la distribution empirique (bruitage, décalages spatio-temporels...). A défaut, il faut restreindre la capacité du système d'apprentissage, opération délicate qui peut être réglée grâce à des tests statistiques croisés [2]. Ce principe ressemble beaucoup à la problématique de la régularisation, destinée à rendre plus robuste les problèmes d'inversion [3].

Une idée intéressante consiste à effectuer un réglage de façon locale sur des systèmes de faible capacité produisant par exemple des fonctions intermédiaires entre la fonction localement linéaire et la fonction localement constante [4]. La localité de l'apprentissage peut rendre ces systèmes très précis. Ils sont cependant souvent lourds d'utilisation, car un apprentissage local y est en principe requis pour chaque exemple traité. L'introduction de connaissances *a priori* au sein de tels systèmes n'est pas impossible, mais on manque encore d'expérience en la matière.

4.1.3 Descente de gradient, rétropropagation du gradient, systèmes modulaires

4.1.3.1 Généralités

Les techniques de descente de gradient exposées aux chapitres précédents permettent d'ajuster les paramètres de systèmes de traitement de l'information.

Le traitement de tâches complexes conduit le plus souvent à des systèmes modulaires. Ajuster successivement les modules ne permet d'obtenir qu'un système sous-optimal au sens de la descente de gradient: il est généralement possible de réduire l'erreur (coût empirique) du système

---

1 **Bottou L., Cortes C., Vapnik V. N.:** *On the mesure of the Effective VC Dimension* - Submited to Neural Computation

2 **Guyon I., Vapnick V., Boser B. E., Bottou L., Solla S. A.:** *Structural Risk Minimization for Character Recognition* - Advances in Neural Information Processing Systems 4, Morgan Kaufman, Denver (1992)

3 **Huynh C., Bourrely J., Bottou L.:** *Sensor Model Inversion and Regularization with Neural Network* - Submited to NIPS 93

4 **Bottou L., Vapnik V. N.:** *Local Learning Algorithms* - Neural Computation Vol 3, Nb 6, pp 888-900 (1992)

par un ajustement simultané des paramètres des différents modules. Comme nous le verrons (cf. § 4.4.1) à propos du MLP-LVQ, il n'est pas toujours possible d'effectuer un apprentissage optimal de façon aussi simple qu'un apprentissage sous-optimal, où la fixité de certains paramètres suffit parfois à empêcher l'apparition de phénomènes aberrants.

La technique de rétropropagation du gradient (utilisée par exemple pour entraîner les différentes couches d'un Perceptron multicouche) constitue une solution algorithmique calculatoirement efficace pour ajuster simultanément par descente de gradient les paramètres de différents modules d'un système.

4.1.3.2 Formalisation

La technique de rétropropagation du gradient est fondamentalement conçue pour calculer le gradient des systèmes modulaires par encahînement. Ces systèmes sont en fait équivalents à des systèmes modulaires à propagation en treillis.

Considérons un système de modules enchaînés:

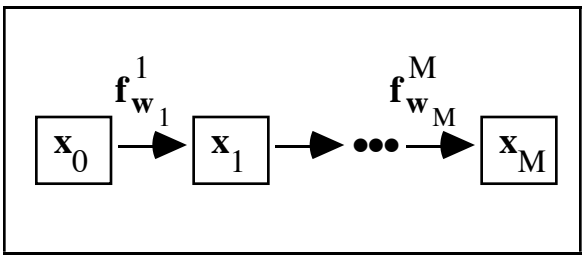


Figure 4.3
Propagation au sein d'un enchaînement de modules

L'enchaînement de modules est la forme la plus simple de système modulaire.

Chaque module constitue une fonction de transfert $\mathbf{f}^m_{\mathbf{w}_m}(\mathbf{x}_{m-1}) = \mathbf{x}_m$ qui prend comme argument la sortie $\mathbf{x}_{m-1}$ du précédent module comme entrée et produit la sortie $\mathbf{x}_m$. On pose:

$$\mathbf{F}^j_i = \mathbf{f}^j_{\mathbf{w}_j} \circ \ldots \circ \mathbf{f}^i_{\mathbf{w}_i} \quad \text{et} \quad \mathbf{W}^i_j = (\mathbf{w}_i \ldots \mathbf{w}_j) \tag{4-1}$$

L'ensemble du système est une fonction de transfert $f(\mathbf{W}^M_1, \mathbf{x}_0) = \mathbf{f}(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M, \mathbf{x}_0) = \mathbf{f}^M_{\mathbf{w}_M} \circ \mathbf{f}^{M-1}_{\mathbf{w}_{M-1}} \circ \ldots \circ \mathbf{f}^2_{\mathbf{w}_2} \circ \mathbf{f}^1_{\mathbf{w}_1}(\mathbf{x}_0) = \mathbf{F}^M_1(\mathbf{x}_0)$. La sortie du dernier module est un argument pour une fonction d'erreur $\mathbf{f}^{M+1}$ que l'on cherche à minimiser.

L'entrée du premier module est l'entrée du système. Par soucis d'homogénéité, on peut poser $\mathbf{F}^m_0 = \mathbf{F}^m_1(\mathbf{x}_0) = \mathbf{x}_m$. La fonction de coût s'écrit alors:

$$J(\mathbf{x}_0, \mathbf{W}^M_1) = \mathbf{F}^M_1(\mathbf{x}_0) = \mathbf{f}^{M+1} \circ \mathbf{F}^M_1(\mathbf{x}_0) = \mathbf{F}^{M+1}_0 \tag{4.2}$$

Pourvu que les différentes fonctions de transfert de chaque module soient suffisamment

régulières, le système modulaire ainsi constitué bénéficie des propriétés de convergence énoncées au chapitre 2.

4.1.3.3 Rétropropagation du gradient

Un système de la forme (4-1) peut être optimisé par descente de gradient. L'algorithme de rétropropagation du gradient est un algorithme efficace pour calculer le gradient de l'erreur du système selon les paramètres de chaque module.

Cet algorithme est fondé sur la très classique dérivation des fonctions composées :

$$(g \circ h)' = h' \bullet (g' \circ h) \tag{4-3}$$

Alors:

$$\nabla_{\mathbf{w}_m} J(\mathbf{x}_0, \mathbf{W}_1^M) = \nabla_{\mathbf{w}_m} \mathbf{F}_{m+1}^{M+1} \circ \mathbf{f}_{\mathbf{w}_m}^m (\mathbf{x}_{m-1})$$

$$= \left(\nabla^T_{\mathbf{w}_m} \mathbf{f}_{\mathbf{w}_m}^m (\mathbf{x}_{m-1}) \right)^T \bullet \nabla_{\mathbf{x}_m} J(\mathbf{x}_0, \mathbf{W}_1^M) \tag{4-4}$$

Or:

$$\nabla_{\mathbf{x}_m} J(\mathbf{x}_0, \mathbf{W}_1^M) = \nabla_{\mathbf{x}_m} \mathbf{F}_{m+2}^{M+1} \circ \mathbf{f}_{\mathbf{w}_{m+1}}^{m+1} (\mathbf{x}_m)$$

$$= \left(\nabla^T_{\mathbf{x}_m} \mathbf{f}_{\mathbf{w}_{m+1}}^{m+1} (\mathbf{x}_m) \right)^T \bullet \nabla_{\mathbf{x}_{m+1}} \mathbf{F}_{m+2}^{M+1} (\mathbf{x}_{m+1})$$

$$= \left(\nabla^T_{\mathbf{x}_m} \mathbf{f}_{\mathbf{w}_{m+1}}^{m+1} (\mathbf{x}_m) \right)^T \bullet \nabla_{\mathbf{x}_{m+1}} J(\mathbf{x}_0, \mathbf{W}_1^M) \tag{4-5}$$

Nous sommes donc en présence d'une relation de récurrence. La relation d'initialisation de cette récurrence s'écrit:

$$\nabla_{\mathbf{x}_{M+1}} J(\mathbf{x}_0, \mathbf{W}_1^M) = \nabla_{\mathbf{x}_{M+1}} \mathbf{x}_{M+1} = 1 \tag{4-6}$$

Ceci nous conduit à un algorithme de récurrence descendante nommé *rétropropagation du gradient*. Cet algorithme suppose que les résultats intermédiaires $\mathbf{x}_m$ calculés lors de la propagation ont étés mémorisés.

Initialisation:

$$\nabla_{\mathbf{x}_{M+1}} J(\mathbf{x}_0, \mathbf{W}_1^M) = 1 \tag{4-7a}$$

Itération descendante:

$$\nabla_{\mathbf{x}_m} J\,(\mathbf{x}_0, \mathbf{W}_1^M) = \left(\nabla^T_{\mathbf{x}_m} \mathbf{f}^{m+1}_{\mathbf{w}_{m+1}}(\mathbf{x}_m) \right)^T \bullet \nabla_{\mathbf{x}_{m+1}} J\,(\mathbf{x}_0, \mathbf{W}_1^M) \qquad (4\text{-}7b)$$

Calcul du gradient:

$$\nabla_{\mathbf{w}_{m+1}} J\,(\mathbf{x}_0, \mathbf{W}_1^M) = \left(\nabla^T_{\mathbf{w}_{m+1}} \mathbf{f}^{m+1}_{\mathbf{w}_{m+1}}(\mathbf{x}_m) \right)^T \bullet \nabla_{\mathbf{x}_{m+1}} J\,(\mathbf{x}_0, \mathbf{W}_1^M) \qquad (4\text{-}7c)$$

La figure 4.3 bis représente cet algorithme.

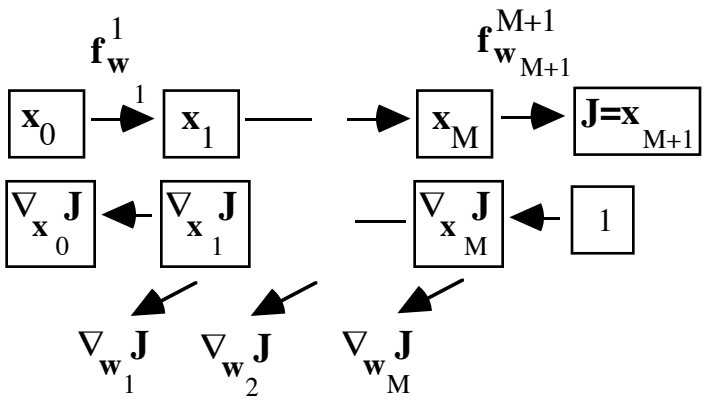


Figure 4.3 bis
Rétropropagation du gradient au sein d'un système modulaire par enchaînement

Une technique de rétropropagation permet de calculer le gradient de l'erreur J relativement aux poids W, au sein de systèmes modulaires par enchaînement.

Cet algorithme de rétropropagation permet ainsi de calculer efficacement le gradient au sein de systèmes modulaires. Il est notamment employé au sein de systèmes intrinsèquement modulaires comme les MLP (cf. § 4.3.2.1) ou les RBF (cf. § 4.3.2.2).

4.2 Plus proches voisinages

4.2.1 Généralités

La notion de plus proche voisinage permet de concevoir des systèmes de classification ou de quantification. Pour chaque exemple traité, on évalue sa "proximité" avec les références du système afin de prendre la décision de classification ou de quantification.

Un tel système est caractérisé par:

- plusieurs références, constituant l'essentiel de ses paramètres,

- une métrique permettant d'évaluer la "proximité" d'un exemple avec les différentes références et
- une stratégie de vote parmi les différentes références en fonction de leur ressemblance avec l'exemple qu'on désire classifier.

Les références peuvent être de simples exemples issus de distributions empiriques ou bien être obtenues par apprentissage. Nous donnons donc un sens large à la notion de plus proche voisinage. Dans les lignes qui suivent, nous étudions successivement les critères de décision (métriques et stratégies de vote) et les techniques d'apprentissage (kMeans, LVQ…).

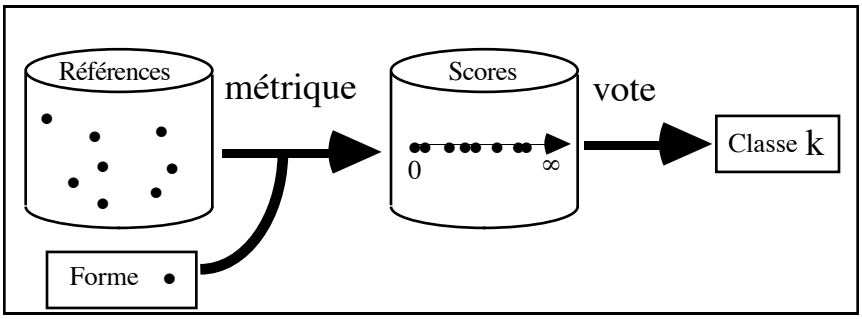


Figure 4-4 Reconnaissance des formes par plus proche voisinage

La classification par plus proche voisinage se fait à l'aide d'une métrique et d'une stratégie de vote.

De tels systèmes incorporent facilement de nombreux types de connaissances *a priori*, en particulier à travers le choix de la métrique. Ainsi, la programmation dynamique (présentée dans le chapitre suivant) permet de contrôler l'effet des distorsions strictement temporelles. Ainsi la distance tangente , permet de rendre compte d'une approximation linéaire d'invariances spatiales et morphologiques (cf. § 4.2.2.1.i). En matière de plus proche voisinage, on fait rarement appel aux partage de paramètres utilisé pour les approximateurs (cf. § 4.3.1.3), coûteux calculatoirement et presque toujours remplaçable par une modification de la métrique.

4.2.2 Critères de décision

La notion de voisinage peut reposer sur de nombreux critères de décision. On peut notamment faire appel à différentes distances, voire à de simples métriques. On peut également recourir à un vote parmi les références les plus proches de chaque exemple traité.

Bien entendu, quand la distribution empirique est très riche, ces différents critères de décision tendent à fournir des réponses identiques.

4.2.2.1 Métriques

Un critère de décision par plus proche voisinage repose tout d'abord sur une métrique.

i Distances classiques: L1, L2, L3, Mahalanobis,...

Les premières métriques auxquelles on pense sont les distances. Les distances les plus simples sont les distances basées sur les normes L_p éventuellement pondérées:

$$d(x,y) = \left(\sum_i \lambda_i (x_i - y_i)^p \right)^{1/p} \tag{4-8}$$

On retrouve souvent dans la littérature des distances visant à corriger certains biais statistiques. Ainsi, la distance Mahalanobis:

$$d(\mathbf{x},\mathbf{y}) = {}^t(\mathbf{x}-\mathbf{y})\ \mathbf{M}^{-1}\ (\mathbf{x}-\mathbf{y}) \tag{4-9}$$

où $\mathbf{M}$ est la matrice de corrélation de la distribution empirique.

La distance de Chernoff est une généralisation de la distance de Mahalanobis. Elle permet de prendre en compte des comportement statistiques différents pour $\mathbf{x}$ et $\mathbf{y}$:

$$d(\mathbf{x},\mathbf{y}) = \frac{s\,(1-s)}{2}\ {}^t(\mathbf{x}-\mathbf{y}) \left(s\ \mathbf{M}_\mathbf{x} + (1-s)\ \mathbf{M}_\mathbf{y} \right)^{-1} (\mathbf{x}-\mathbf{y}) + \frac{1}{2}\ \ln \frac{| s\ \mathbf{M}_\mathbf{x} + (1-s)\ \mathbf{M}_\mathbf{y} |}{| \mathbf{M}_\mathbf{x} |^s\ | \mathbf{M}_y |^{(1-s)}} \tag{4-10}$$

où $\mathbf{M}_\mathbf{x}$ et $\mathbf{M}_\mathbf{y}$ sont les matrices de covariances des variables aléatoires $\mathbf{x}$ et $\mathbf{y}$ et où s est un paramètre scalaire. quand $s = \frac{1}{2}$, on obtient la distance de Bhattacharyya. Une telle distance peut s'avérer utile quand l'utilisation opérationnelle d'un système de plus proche voisinage se fait dans des conditions différentes de celles de son apprentissage. Il arrive qu'on sache exprimer *a priori* $\mathbf{M}_\mathbf{y}$ ou les diférences entre $\mathbf{M}_\mathbf{x}$ et $\mathbf{M}_\mathbf{y}$.

ii Autres métriques - Exemple de la distance tangente

Cependant, les trois axiomes des distances sont contraignants et on est parfois tenté de ne pas les respecter. C'est notamment le cas de la métrique que permet de calculer l'algorithme de programmation dynamique que nous étudions dans le chapitre suivant (cf. § 5.2.1.1).

C'est également le cas de la distance tangente [1]. Pour de nombreux problèmes de classification, on dispose d'une certaine information *a priori* sur les distorsions des formes laissant invariantes leur classe.

---

1 **Simard P., Le Cun Y., Denker J.:** *Efficient pattern recognition using a new transformation distance* - Advances in Neural Information Processing Systems 5 (1993)

Par exemple, dans le cas de la reconnaissance de caractères (pour la distance tangente a initialement été mise au point), on sait que certaines transformations géométriques simples dans la topologie des images analogiques telles les translations, rotations, affinités et épaississements laissent les classes invariantes. Cependant, dans la topologie des images digitalisées sous forme de pixels, ces transformations sont hautement non-linéaires.

Il est donc difficile de déterminer une distance prenant en compte de telles invariances, et encore plus difficile de concevoir un algorithme utilisable pour calculer un telle distance. La technique de la distance tangente a été développée pour traiter cette difficulté.

Considérons des formes qu'on cherche à classifier. Les transformations géométriques laissant invariantes les classes permettent d'engendrer une nappe à partir de chaque forme. Il en va de même pour l'exemple. La distance tangente entre deux formes est la distance entre les deux hyperplans tangents chacun à une des deux nappes.

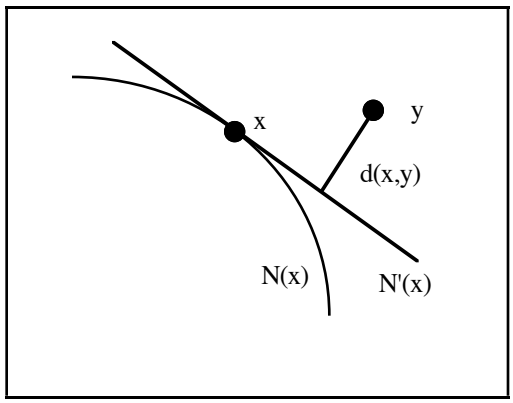


Figure 4-5
Distance tangente

La distance tangente permet de rendre compte de propriétés d'invariance a priori*, sous forme de transformation non-linéaires qui font d'un exemple x une nappe d'exemples virtuels N(x). Cette nappe est approchée localement par un hyperplan tangent d'exemples virtuels N'(x).*

Bien sûr, la distance tangente n'est pertinente qu'au voisinage de chaque forme. C'est pourquoi on l'utilise en conjonction avec des distances plus classiques.

Le principe de la distance tangente peut également être utilisé pour spécifier des invariances de traitement d'un système paramètrique (contraintes) ou bien pour modifier aléatoirement des formes selon des transformation sans changer leur nature [1], ce qui permet d'enrichir la distribution empirique.

4.2.2.2 Stratégies de votes et techniques de rejet

Un critère de décision par plus proche voisinage repose également sur une stratégie de vote au

1 **Simard P.:** *Tangent prop: a formalism for specifying selected invariances in adaptive networks* - Advances in Neural Information Processing Systems 4 (1992)

sein des références les plus proches de chaque exemple traité. Rappelons que pour cela, chaque référence est associée à une classification, ou bien un jeu de probabilité de classification.

NN, KNN

La stratégie de vote la plus simple consiste à sélectionner pour chaque exemple traité la ou les k références les plus proches (stratégie des K plus Proches Voisins ou K Nearest Neighbours).

Une variante consiste à pondérer le vote (la plus proche référence ayant plus de poids que la dernière).

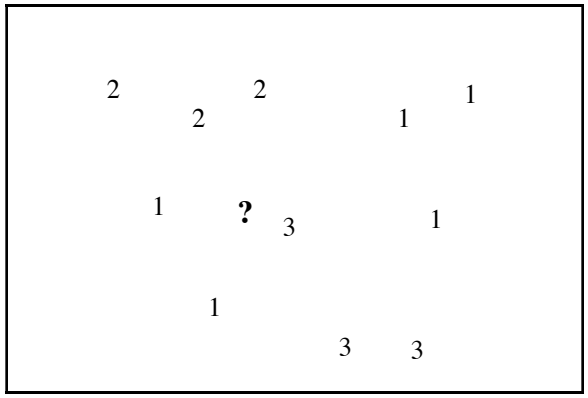


Figure 4-6
Stratégie de vote KNN

Dans cet espace à deux dimensions, le point d'interrogation est classé :

3 par un 1-NN, 1 par un 3-NN,
2 par un 6-NN, 1 par les 10+NN

Fixer le nombre k de références à faire voter peut s'avérer délicat. Quand k est grand, le système tend à produire une classification uniforme. Quand k est petit, le système tend à faire la part belle aux exemples anachroniques et à rendre les frontières de classification irrégulières. Le nombre de références à faire voter est parfois un savant dosage, directement lié à la densité de probabilité des références aux frontières de classifications (zones où la décision est incertaine).

En principe le nombre k est fixé *a priori*. Cependant, une tendance moderne consiste à optimiser k localement, en faisant abstraction tour à tour d'une référence locale pour prendre des mesures du comportement du système en fonction du nombre k [1].

Par ailleurs, on montre que la méthode kNN fournit une approximation arbitrairement fine du risque moyen quand k est suffisamment grand et que les exemples sont tirés aléatoirement de la distribution empirique, en nombre suffisamment grand au regard de k [2].

---

1 **Bottou L., Vapnik V. N.:** *Local Learning Algorithms* - Neural Computation Vol 3, Nb 6, pp 888-900 (1992)

2 **Devijer P. A., Kitter J.:** *Pattern Recognition: A statistical Approach* - Prentice Hall International (1982)

Dans le cas particulier du 1-plus proche voisin, la partition de l'espace est un diagramme de Voronoï:

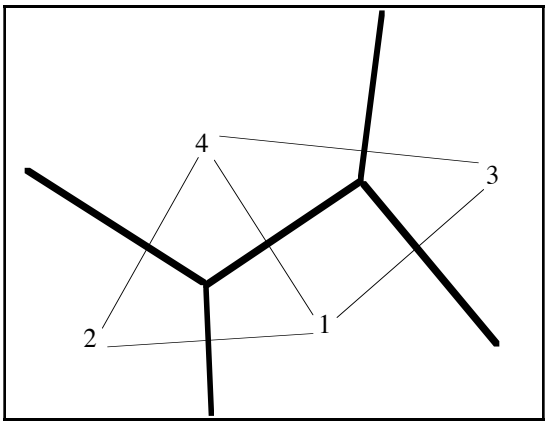


Figure 4-7
Diagramme de Voronoï

Considérons un ensemble de références (ici numérotés de 1 à 4). La division de l'espace selon l'identité de la plus proche référence conduit à un diagramme de Voronoï. Les frontières des domaines d'influence sont des tronçons de médiatrices.
Le diagramme de Delaunay est le diagramme dual, qui relie les références contiguës dans le diagramme de Voronoï.

Fenêtres de Parzen

Une stratégie de vote alternative consiste à fixer un seuil de dissemblance s au delà duquel les références sont considérées comme sans rapport avec l'exemple traité. Le vote se fait alors au sein des références suffisamment ressemblantes, quel que soit leur nombre. Quand aucune référence ne satisfait à la condition de ressemblance, l'exemple ne peut être classifié et le système le rejette.

Une variante consiste à pondérer le vote des références en fonction de leur proximité avec l'exemple.

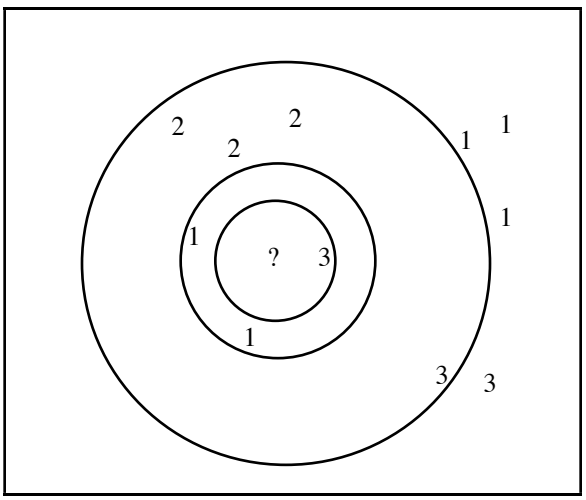


Figure 4-8
Stratégie de vote Fenêtres de Parzen

Dans cet espace à deux dimensions, le point d'interrogation est classé :

3 par une fenêtre de très petit rayon,
1 par une fenêtre de petit rayon,
3 par une fenêtre de moyen rayon,
2 par une fenêtre de grand rayon
1 par une fenêtre de très grand rayon et
n'est pas classé quand la fenêtre est trop petite.

Fixer le seuil de dissemblance peut s'avérer délicat. Quand il est grand, le système tend à rendre une classification peu évolutive sur l'espace des exemples. Quand il est petit, le système tend à faire la part belle aux exemples anachroniques et à rendre une classification erratique aux

frontières des zones de classification. Le nombre de références à faire voter est donc un savant dosage, directement lié à la densité spatiale de références aux frontières de classifications (zones où la décision est incertaine).

En principe, le seuil est fixé *a priori*. Cependant, une tendance moderne consiste à optimiser le seuil *a posteriori* en faisant abstraction tour à tour d'une ou plusieurs références locales pour prendre des mesures du comportement du système en fonction du seuil [1].

Par ailleurs, on montre que la méthode fenêtres de Parzen fournit une approximation arbitrairement fine du risque moyen quand les fenêtres sont suffisamment petites et que les exemples sont tirés aléatoirement de la distribution empirique, en nombre suffisamment grand au regard de la petitesse des fenêtres [2].

Cartes topologiques de Kohonen

Les cartes topologiques de Kohonen permettent des stratégies de votes spéciales. Ces cartes sont constituées par un maillage virtuel de références. Une règle d'apprentissage spéciale (cf. § 4.2.3.2) permet en principe à ce maillage virtuel d'acquérir une certaine cohérence: deux références proches dans la topologie du maillage seront également proches dans la topologie de l'espace des exemples.

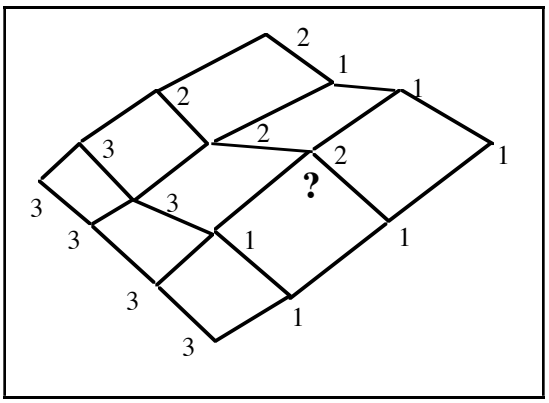


Figure 4-9
Cartes topologiques

Sans tenir compte d'éventuelles pondérations en fonction de l'éloignement des nœuds dans la topologie du maillage, le point d'interrogation sera classé 2 par plus proche voisinage, 1 par les 5 plus proches nœuds et 3 par les 16 plus proches nœuds.

Quand un exemple est traité, on recherche la plus semblable des références. Le maillage de la carte permet alors sélectionner pour le vote non pas les références les plus semblables dans l'espace des exemples, mais les références les plus proches dans la topologie du maillage. On peut également pondérer l'importance de chaque vote en fonction de sa proximité dans le

---

1 **Bottou L., Vapnik V. N.:** *Local Learning Algorithms* - Neural Computation Vol 3, Nb 6, pp 888-900 (1992)

2 **Devijer P. A., Kitter J.:** *Pattern Recognition: A statistical Approach* - Prentice Hall International (1982)

maillage avec la plus semblable des références.

4.2.3 Méthodes d'apprentissage et d'adaptation

Le critère de décision suppose un jeu cohérent de références. Ces références peuvent être obtenues par simple mémorisation de nombreux exemples réels, tirés de la distribution empirique.

Cependant, il apparaît avantageux voire nécessaire de concentrer la connaissance *a posteriori* en quelques références artificielles, dont on a ajusté préalablement les coordonnées pour optimiser le comportement du système. Cette concentration permet de réduire considérablement le temps de calcul et la mémoire requise. Mais elle permet également de rendre le système plus robuste en atténuant l'effet des exemples anachroniques et en "lissant" le comportement du système aux frontières de classification.

4.2.3.1 Méthodes pour plus proche voisin euclidien

Le plus naturel des critères de décision par plus proche voisinage est sans conteste le 1-plus proche voisin euclidien. Ce critère a donc donné lieu à de nombreuses méthodes d'apprentissage qui peuvent souvent être étendues aux autres systèmes de plus proche voisinage. Nous exposons ici quelques méthodes parmi les plus connues. Ces méthodes peuvent généralement s'adapter à d'autres critères de décision, en particulier à d'autres métriques.

KMoyennes ou kMeans

Les techniques de plus proche voisinage peuvent être indifféremment exploitées en tant que techniques de quantification ou de classification.

La méthode des kMoyennes s'inscrit parfaitement dans ce cadre, puisqu'elle garantit qu'à la fin de l'apprentissage, chaque référence du système est le centre des données empiriques pour lesquelles elle est la plus proche des références.

Cette méthode déplace chaque référence vers le centre de gravité des donnés empiriques qu'elle quantifie.

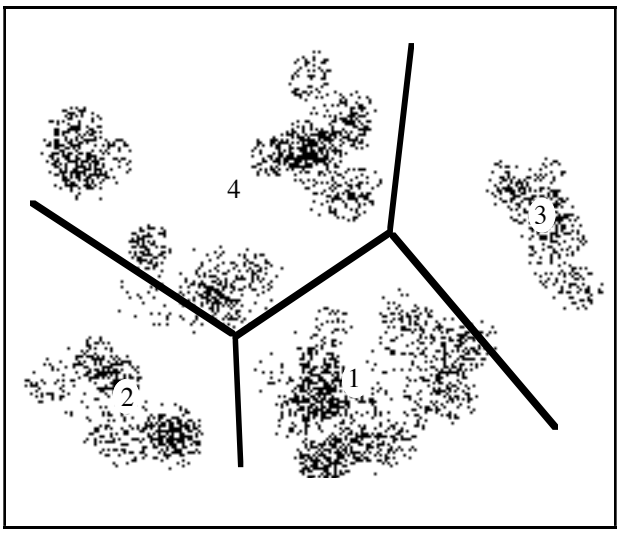


Figure 4-10
Quantification vectorielle par kMeans

A l'issue de l'apprentissage, chaque référence est le centre des données empiriques qu'elle quantifie.

Cette technique de quantification n'est pas parfaite mais elle est simple, rapide et efficace. Elle est très utilisée, notamment pour initialiser d'autres techniques plus lourdes et plus irrégulières, telles que les LVQ

L'algorithme original est itératif; il consiste à calculer pour chaque référence le centre de gravité des données empiriques qu'elle quantifie, puis à assigner ce centre comme nouvelles coordonnées de la référence:

$$\mathbf{w}_{i,n+1} = E_{\mathbf{X}_i}\{ \tilde{\mathbf{x}} \mid \mathbf{W}_n \} , \tag{4-11}$$

où les références $\mathbf{w}_{i,n}$ constituent le jeu de paramètres $\mathbf{W}_n$.

En fait, cet algorithme n'est qu'une version brutale de l'algorithme de descente de gradient global:

$$\mathbf{w}_{i,n+1} = (Id-\Gamma_n)\, \mathbf{w}_{i,n} + \Gamma_n\, E_{\mathbf{X}_i}\{ \tilde{\mathbf{x}} \} = \mathbf{w}_{i,n} + \Gamma_n \left(E_{\mathbf{X}_i}\{ \tilde{\mathbf{x}} \} - \mathbf{w}_{i,n} \right) \tag{4-12}$$

Considérons un exemple $\tilde{\mathbf{x}}_n$. On désigne par i^*_n l'indice de la plus proche référence. L'algorithme stochastique de kMeans s'écrit:

$$\mathbf{w}_{i^*_n,n+1} = (1-\varepsilon_n)\, \mathbf{w}_{i^*_n,n} + \varepsilon_n\, \tilde{\mathbf{x}}_n \tag{4-13}$$

les autres références $\mathbf{w}_{i,n}$ de $\mathbf{W}_n$ restant inchangées. Ces algorithmes découlent de la fonction de coût local:

$$J^{KMeans}(\mathbf{x},\mathbf{W}) = \min_j \frac{1}{2} \mid \tilde{\mathbf{x}}-\mathbf{w}_j \mid^2 \tag{4-14}$$

La convergence de ces algorithmes de descente de gradient est assurée dès que (ε_n) vérifie les conditions (2-80), si la distribution est continue.

L'algorithme des kMoyennes est un algorithme de quantification. Il se prête au formalisme des

fonctions discriminantes (3-6) avec:

$$f_{sm}(\mathbf{x},\mathbf{w}) = (\mathbf{x}-\mathbf{w}_s)^2 - (\mathbf{x}-\mathbf{w}_m)^2 \qquad (4\text{-}15)$$

La méthode des kMoyennes est en principe non-supervisée. Cependant, conformément au § 4.1.1, il est possible de la rendre supervisée en l'appliquant successivement aux données des différentes classes.

LVQ

La méthode LVQ ressemble beaucoup à la méthode kMeans, dont elle vise à pallier l'absence de discrimination. L'algorithme kMeans stochastique supervisé (i.e. par apprentissage séparé pour chaque classe, cf. § 4.1.1 et figure 4.2) consiste à attirer la plus proches références de bonne classe. La méthode LVQ consiste, pour chaque exemple, à repousser les plus proches références de mauvaise classe et attirer les plus proches références de bonne classe. Ceci conduit, en principe, à des algorithmes supervisés discriminants. Il existe cependant une variante non supervisée (et donc non discriminante) de LVQ. La méthode LVQ connaît plusieurs variantes discriminantes, constituant autant d'améliorations successives.

Les algorithmes de la famille LVQ sont typiquement des algorithmes adaptatifs; en effet, ils ont été introduits sous la forme de règles d'adaptation des paramètres (2-5). A chaque étape n, tous tirent un exemple $\mathbf{x}_n$ au hasard puis adaptent une ou plusieurs références $\mathbf{w}_{i,n}$ du système.

<u>LVQ non supervisé</u>

Cet algorithme non supervisé consiste à tirer un exemple au hasard, puis à rapprocher la plus proche référence et à éloigner la seconde plus proche référence.

Considérons un exemple $\mathbf{x}_n$. En appelant i^*_n (resp. i^{**}_n) l'indice de la plus proche (resp. seconde plus proche) référence. La règle de réévaluation de LVQ non-supervisé s'écrit:

$$\mathbf{w}_{i^*_n,n+1} = (1-\varepsilon_n)\,\mathbf{w}_{i^*_n,n} + \varepsilon_n\,\mathbf{x}_n$$

$$\mathbf{w}_{i^{**}_n,n+1} = (1+\varepsilon_n)\,\mathbf{w}_{i^{**}_n,n} - \varepsilon_n\,\mathbf{x}_n \qquad (4\text{-}16)$$

Ceci revient à optimiser la fonction de coût:

$$J^{LVQ}(\mathbf{x},\mathbf{w}) = \frac{1}{2}\min_i 2\,|\mathbf{x}-\mathbf{w}_i|^2 - \frac{1}{2}\min_i |\mathbf{x}-\mathbf{w}_i|^2 \qquad (4\text{-}17)$$

La convergence de cet algorithme de descente de gradient est assurée dès que (ε_n) vérifie les

conditions (2-80), si la distribution est continue.

LVQ1

L'algorithme LVQ1 est un algorithme supervisé. Il consiste à tirer un exemple au hasard, puis à rapprocher la plus proche référence si elle est de la bonne classe ou à l'éloigner si elle est de la mauvaise classe.

Considérons un exemple $\mathbf{x}_n = (\tilde{\mathbf{x}}_n, k_n)$ On désigne par i_n^* l'indice de la plus proche des références $\mathbf{w}_i = (\tilde{\mathbf{w}}_i, l_i)$. La règle de réévaluation de LVQ1 s'écrit:

$$\tilde{\mathbf{w}}_{i_n^*,n} = \tilde{\mathbf{w}}_{i_n^*,n-1} + \sigma_n \varepsilon_n (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^*,n-1}) \qquad (4\text{-}18)$$

où $\sigma_n = 1 \Leftrightarrow k_n = l_{i_n^*}$ (classification correcte); sinon $\sigma_n = -1$

Cette règle d'adaptation ne découle pas d'une fonction de coût continue. En effet, on pourrait croire qu'elle vise à optimiser la fonction de coût:

$$J^{LVQ1}(\mathbf{x},\mathbf{w}) = - \sigma_n \frac{1}{2} \min_i 2 \lceil \tilde{\mathbf{x}} - \tilde{\mathbf{w}}_i \rceil^2 \qquad (4\text{-}19)$$

où $\sigma_n = 1 \Leftrightarrow k_n = l_{i_n^*}$ (classification correcte); sinon $\sigma_n = -1$

Or, dans l'espace des paramètres, aux frontières des domaines de classification, une variation infinitésimale des paramètres suffit à rendre un exemple bien ou mal classifié et σ_n change de signe. On en déduit que la fonction de coût 4-19 est discontinue aux frontières de classifications. La règle d'adaptation ne se déduit de la fonction de coût que dans la mesure ou on ignore les masses de Dirach du gradient engendrées par les discontinuités de la fonction de coût aux frontières de classification.

Cette fonction de coût irrégulière et non minorée est problématique; en particulier, les algorithmes de descente de gradient 4-18 ne bénéficient pas du théorème de convergence du chapitre 2.

LVQ2

L'algorithme LVQ2 est un algorithme supervisé. Il consiste à tirer un exemple au hasard, puis à éloigner la plus proche référence si elle est de la mauvaise classe et à rapprocher la seconde plus proche référence. Si la plus proche référence de bonne classe est trop nettement mal classifiée, on ne fait rien.

L'algorithme LVQ2 (supervisé) diffère de l'algorithme LVQ non-supervisé simplement par le sens de modification des références et par l'absence de modification des paramètres en cas de bonne classification ou en cas de "trop nette mauvaise classification".

La notion de "trop nette mauvaise classification" est une des originalités des algorithmes de la famille LVQ2. Elle vise à éviter que les exemples trop nettement mal classifiés ne déstabilisent les références avoisinantes, engendrant d'autres mauvaises classifications. Les exemples trop mals classifiés sont donc laissés à leur triste sort.

Plus formellement, considérons un exemple $\mathbf{x}_n = (\tilde{\mathbf{x}}_n, k_n)$. On désigne par i^*_n (resp. i^{**}_n) l'indice de la plus proche (resp. seconde plus proche) référence $\mathbf{w}_i = (\tilde{\mathbf{w}}_i, l_i)$. La règle de réévaluation de LVQ2 s'écrit:

$$\tilde{\mathbf{w}}_{i^*_n,n+1} = (1+\tau_n\tau'_n\varepsilon_n)\,\tilde{\mathbf{w}}_{i^*_n,n} - \tau_n\tau'_n\varepsilon_n\,\tilde{\mathbf{x}}_n$$

$$\tilde{\mathbf{w}}_{i^{**}_n,n+1} = (1-\tau_n\tau'_n\varepsilon_n)\,\tilde{\mathbf{w}}_{i^{**}_n,n} + \tau_n\tau'_n\varepsilon_n\,\tilde{\mathbf{x}}_n \qquad (4\text{-}20)$$

où $\tau_n = 0 \Leftrightarrow k_n = l_{i^*_n}$ (classification correcte) ; sinon $\tau_n = 1$

et $\tau'_n = 0 \Leftrightarrow \left|(\tilde{\mathbf{w}}_{i^{**}_n,n} - \tilde{\mathbf{w}}_{i^*_n,n}) \bullet \left(\tilde{\mathbf{x}}_n - \frac{\tilde{\mathbf{w}}_{i^{**}_n,n} + \tilde{\mathbf{w}}_{i^*_n,n}}{2} \right)\right| > \frac{\delta}{2}(\tilde{\mathbf{w}}_{i^{**}_n,n} - \tilde{\mathbf{w}}_{i^*_n,n})^2$

(classification trop nette) ; sinon $\tau'_n = 1$

Le nombre logique τ_n (resp. τ'_n) dit si la clasification effectuée est correcte (resp. trop nette). La figure 4.11 met en valeur les domaines spatiaux correspondant aux différentes valeurs logiques de τ_n et τ'_n qui déterminent si les paramètres vont être modifiés.

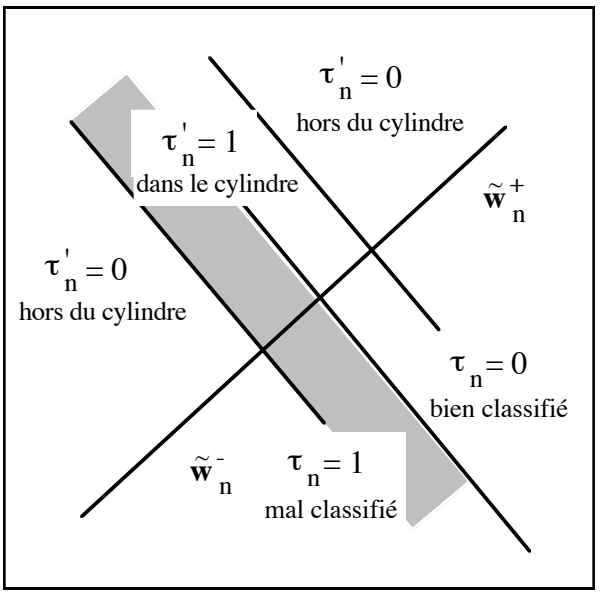


Figure 4.11
Conditions de réajustement des références par LVQ2

Considérons les deux références les plus proches d'un exemple. L'algorithme LVQ2 réajuste les références quand l'exemple est situé dans la zone ici en grise , c'est-à-dire s'il est mal classifié et situé dans un cylindre (volume de révolution engendrée par une droite) encadrant la médiatrice des deux références.
Sinon, l'exemple est considéré comme bien classifié ou trop nettement mal classifié.

La lecture de cet algorithme engendre immédiatement une question: pourquoi rapprocher la seconde meilleure référence si elle non plus n'est pas de la bonne classe ?

Cette règle correspond à l'algorithme de descente de gradient stochastique issue d'une fonction de coût de la forme:

$$J^{LVQ2}(\mathbf{x}_n,\mathbf{W}_n) = -\tau_n\tau_n' \left(\frac{1}{2} \min_i 2\, |\tilde{\mathbf{x}}-\tilde{\mathbf{w}}_i|^2 - \frac{1}{2} \min_i |\tilde{\mathbf{x}}-\tilde{\mathbf{w}}_i|^2 \right) + f(\tau_n,\tau_n') \qquad (4\text{-}21)$$

où $\tau_n = 0 \Leftrightarrow k_n = l^*_i$ (classification correcte) ; sinon $\tau_n = 1$

et $\tau'_n = 0 \Leftrightarrow \left| (\tilde{\mathbf{w}}_{i^{**}_n,n} - \tilde{\mathbf{w}}_{i^*_n,n}) \bullet \left(\tilde{\mathbf{x}}_n - \frac{\tilde{\mathbf{w}}_{i^{**}_n,n} + \tilde{\mathbf{w}}_{i^*_n,n}}{2} \right) \right| > \frac{\delta}{2} (\tilde{\mathbf{w}}_{i^{**}_n,n} - \tilde{\mathbf{w}}_{i^*_n,n})^2$

(classification trop nette) ; sinon $\tau'_n = 1$

On observe alors le second défaut inexplicable la fonction de coût LVQ2: elle est continue sauf aux limites du domaine de "trop nette mauvaise classification".

On peut cependant remarquer que, contrairement à l'algorithme LVQ, elle est constante aux frontières de classification.

LVQ2-bis

Il existe une variante très répandue visant à corriger les inexplicables défauts de LVQ2. Elle est tellement naturelle que des lecteurs trop hâtifs l'emploient spontanément au lieu de l'algorithme LVQ2 d'origine. Aussi, il est bien difficile de lui attribuer une paternité unique. Nous la

désignons par LVQ2-bis.

L'algorithme LVQ2-bis est un algorithme supervisé. Il consiste à tirer un exemple au hasard, puis, si la plus proche référence est de la mauvaise classe, à l'éloigner et à rapprocher la plus proche référence de bonne classe. Si la plus proche référence de bonne classe est "trop largement mal classifiée", on ne fait rien.

L'algorithme LVQ2-bis diffère de l'algorithme LVQ2 simplement par la prise en considération de la meilleure référence de bonne classe au lieu de la seconde meilleure référence et par son critère de "trop large mauvaise classification".

Plus formellement, considérons un exemple $\mathbf{x}_n = (\tilde{\mathbf{x}}_n, k_n)$. On appelle i_n^* (resp. i_n^+) l'indice de la plus proche (resp. plus proche de bonne classe) référence $\mathbf{w}_i = (\tilde{\mathbf{w}}_i, l_i)$. En cas de classification correcte, ces deux indices sont confondus. La règle de réévaluation de LVQ2-bis s'écrit:

$$\tilde{\mathbf{w}}_{i_n^*,n+1} = (1+\tau_n\tau_n'\varepsilon_n)\,\tilde{\mathbf{w}}_{i_n^*,n} - \tau_n\tau_n'\varepsilon_n\,\tilde{\mathbf{x}}_n$$

$$\tilde{\mathbf{w}}_{i_n^+,n+1} = (1-\tau_n\tau_n'\varepsilon_n)\,\tilde{\mathbf{w}}_{i_n^+,n} + \tau_n\tau_n'\varepsilon_n\,\tilde{\mathbf{x}}_n \qquad (4\text{-}22)$$

où $\tau_n = 0 \Leftrightarrow k_n = l_{i_n^*}$ (classification correcte); sinon $\tau_n = 1$

et $\tau_n' = 0 \Leftrightarrow \lceil\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^+,n}\rceil^2 - \lceil\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^*,n}\rceil^2 > \delta$ (classif. trop nette) ; sinon $\tau_n' = 1$

Le nombre logique τ_n (resp. τ_n') dit si la clasification effectuée est correcte (resp. trop nette). Les algorithmes LVQ2 et LVQ2bis difèrent dans leur définition de τ_n', c'est-à-dire leur critère de trop nette classification. Ceci permet à la fonction de coût de LVQ2bis d'être continue et de prendre une forme plus synthétique (4-23).

La figure 4.12 met en valeur les domaines spatiaux correspondant aux différentes valeurs logiques de τ_n et τ_n' qui déterminent si les paramètres vont être modifiés. Elle peut être comparée avec la figure 4.11.

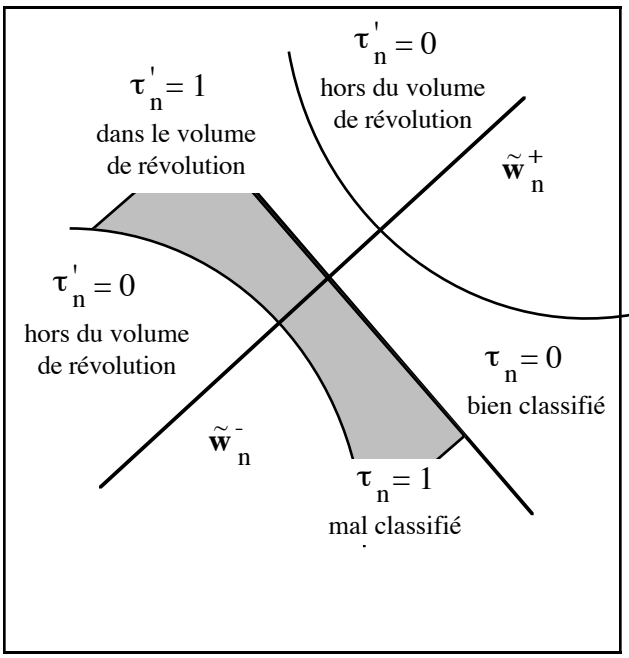


Figure 4.12
Conditions de réajustement des références par LVQ2-bis

Considérons les deux références les plus proches d'un exemple. L'algorithme LVQ2 -bis réajuste les références quand l'exemple est situé dans la zone en grise sur la figure, c'est-à-dire si il est mal classifié et situé dans un volume de révolution engendrée par une hyperbole encadrant la médiatrice des deux références. Sinon, l'exemple est considéré comme bien ou trop nettement mal classifié.

Cette règle correspond à l'algorithme de descente de gradient stochastique issue de la fonction de coût:

$$J^{\text{LVQ2-bis}}(\mathbf{x}_n,\mathbf{W}_n) = \frac{1}{2}\min\left(\delta, \min_{i\in C(k)} |\tilde{\mathbf{x}}-\tilde{\mathbf{w}}_i|^2 - \min_i |\tilde{\mathbf{x}}-\tilde{\mathbf{w}}_i|^2\right) \qquad (4\text{-}23)$$

En notant $\mathbf{w}_n^* = (\tilde{\mathbf{w}}_n^*, l_n^*) = \mathbf{w}_{i_n^*}$ et $\mathbf{w}_n^+ = (\tilde{\mathbf{w}}_n^+, l_n^+) = \mathbf{w}_{i_n^+}$, cette fonction peut également s'écrire:

$$J^{\text{LVQ2-bis}}(\mathbf{x}_n,\mathbf{W}_n) = \frac{1}{2}\min\left(\delta, |\tilde{\mathbf{x}}-\tilde{\mathbf{w}}_n^+|^2 - |\tilde{\mathbf{x}}-\tilde{\mathbf{w}}_n^*|^2\right)$$

Cette fonction de coût est continue partout. Elle est nulle aux frontières des classifications correctes. Elle est constante (égale à δ) en cas de très nette mauvaise classification.

La convergence de cet algorithme de descente de gradient est assurée dès que (ε_n) vérifie les conditions (2-80), si la distribution est continue.

On peut remarquer qu'elle ressemble beaucoup au risque local: nulle en cas de classification correcte, constante en cas de nette mauvaise classification. Deux choses l'en différentient: la progressivité du passage du coût nul au coût de mauvaise classification et la valeur du coût de classification incorrecte (ici égal à 1 pour toute nette mauvaise classification).

LVQ2-ter

Il est facile de transformer la constante du coût de classification incorrecte. Ceci permet de dériver de LVQ2-bis une fonction qui approche le risque local:

$$J^{\text{LVQ2-ter}}(\mathbf{x}_n,\mathbf{W}_n) = J_{k_n}\ \frac{2}{\delta} J^{\text{LVQ2-bis}}(\mathbf{x}_n,\mathbf{W}_n) = J_{k_n}\ \min\left(1, \frac{|\tilde{\mathbf{x}}-\tilde{\mathbf{w}}_n^+|^2-|\tilde{\mathbf{x}}-\tilde{\mathbf{w}}_n^*|^2}{\delta}\right) \qquad (4\text{-}24)$$

Cette approximation du risque local est d'autant plus fine (mais d'autant plus irrégulière) que δ est faible. Cependant, il ne s'agit pas d'une approximation uniforme mais d'une simple approximation locale dans l'espace des formes.

Le coefficient J_k permet d'associer un coût de mauvaise classification propre à chaque classe k.

L'algorithme de descente de gradient stochastique qui en découle s'écrit:

$$\tilde{\mathbf{w}}_{i_n^*,n+1} = \tilde{\mathbf{w}}_{i_n^*,n} - \tau_n\tau_n'\varepsilon_n \frac{2}{\delta} \bullet (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^*,n})$$

$$\tilde{\mathbf{w}}_{i_n^+,n+1} = \tilde{\mathbf{w}}_{i_n^+,n} + \tau_n\tau_n'\varepsilon_n \frac{2}{\delta} \bullet (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^+,n}) \qquad (4\text{-}25)$$

où $\tau_n = 0 \Leftrightarrow k_n = l_{i_n^*}$ (classification correcte); sinon $\tau_n = 1$

et $\tau_n' = 0 \Leftrightarrow |\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^+,n}|^2 - |\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^*,n}|^2 > \delta$ (classif. trop nette) ; sinon $\tau_n' = 1$

Le nombre logique τ_n (resp. τ_n') dit si la clasification effectuée est correcte (resp. trop nette). Sur le plan calculatoire, chaque étape de cet algorithme demande 2 • m multiplications, 4 • m additions et environ 6 • m accès mémoire où m est la dimension des formes $\tilde{\mathbf{x}}$.

La convergence de cet algorithme de descente de gradient est assurée dès que (ε_n) vérifie les conditions (2-80), si la distribution est continue.

LVQ2+

L'algorithme LVQ2+ est un algorithme supervisé. Il consiste à tirer un exemple au hasard, puis à éloigner la plus proche référence si elle est de la mauvaise classe et à rapprocher la plus proche référence de bonne classe. Si la plus proche référence de bonne classe est trop loin, on ne fait rien.

L'algorithme LVQ2+ diffère des algorithmes LVQ2-bis/LVQ2-ter par une normalisation.

Plus formellement, considérons un exemple $\mathbf{x}_n = (\tilde{\mathbf{x}}_n, k_n)$. On appelle i_n^* (resp. i_n^+) l'indice de la plus proche (resp. plus proche de bonne classe) référence $\mathbf{w}_i = (\tilde{\mathbf{w}}_i, l_i)$. En cas de classification correcte, ces deux indices sont confondus. On note $\mathbf{w}_n^* = (\tilde{\mathbf{w}}_n^*, l_n^*) = \mathbf{w}_{i_n^*}$ et

$\mathbf{w}_n^+ = (\tilde{\mathbf{w}}_n^+, l_n^+) = \mathbf{w}_{i_n^+}$. La fonction de coût LVQ2+ s'écrit:

$$J^{LVQ2+}(\mathbf{x}_n, \mathbf{W}_n) = \min(1, \frac{1}{\delta}\left(\frac{|\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_n^+|^2}{|\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_n^*|^2} - 1\right)) \tag{4-26}$$

Cette fonction de coût est continue. En particulier, on peut remarquer qu'elle est nulle aux frontières du domaine de classification correcte. La convergence de cet algorithme de descente de gradient est assurée dès que (ε_n) vérifie les conditions (2-80), si la distribution est continue.

Cette fonction est dotée en outre de deux propriétés remarquables. De même que la fonction de coût LVQ2-ter, elle constitue une approximation du risque local. De plus, elle est sans dimension, ce qui est une propriété très utile pour l'utiliser dans le cadre de certains systèmes modulaires tels que les MLP-LVQ (cf. § 4.4.2).

L'algorithme de descente de gradient stochastique qui s'en déduit s'écrit:

$$\begin{aligned}
\tilde{\mathbf{w}}_{i_n^*,n+1} &= \tilde{\mathbf{w}}_n^* - \tau_n \tau_n' \varepsilon_n \frac{2}{\delta} \bullet \frac{|\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_n^+|^2}{|\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_n^*|^4} \bullet (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_n^*) \\
\tilde{\mathbf{w}}_{i_n^+,n+1} &= \tilde{\mathbf{w}}_n^+ + \tau_n \tau_n' \varepsilon_n \frac{2}{\delta} \bullet \frac{1}{|\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_n^*|^2} \bullet (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_n^+)
\end{aligned} \tag{4-27}$$

où $\tau_n = 0 \Leftrightarrow k_n = l_{i_n^*}$ (classification correcte); sinon $\tau_n = -1$

et $\tau_n' = 0 \Leftrightarrow \frac{|\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^+,n}|^2}{|\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^*,n}|^2} > 1+\delta$ (classification très nette); sinon $\tau_n' = 1$

Le nombre logique τ_n (resp. τ_n') dit si la clasification effectuée est correcte (resp. trop nette). On peut remarquer la grande analogie avec l'algorithme LVQ2-bis. Sur le plan calculatoire, chaque étape de cet algorithme demande 4 • m multiplications, 4 • m additions et environ 10 • m accès mémoire, où m est la dimension des formes $\tilde{\mathbf{x}}$. Sur le plan calculatoire, cet algorithme est donc presque aussi léger que l'algorithme de descente de gradient stochastique LVQ2.

4.2.3.2 Cartes topologiques

Nous avons déjà évoqué (cf. § 4.2.2.2) l'utilisation des cartes topologiques comme technique de vote alternative aux K-NN et fenêtres de Parzen. Les lignes qui suivent ont pour objet de

détailler quelques techniques d'apprentissage (en principe non supervisées) pour de telles cartes.

Les carte topologiques ne sont pas systématiquement utilisées en combinaison avec une distance euclidienne. Dés ses premiers articles sur le sujet, T. Kohonen utilisait d'autres métriques telles que le produit scalaire de vecteurs normalisés (cette métrique mesure l'angle entre deux vecteurs) [1]. Faute de pouvoir être exhaustifs, nous n'examinerons que le cas général et le cas particulier de la distance euclidienne.

i Apprentissage de cartes topologiques selon Kohonen

L'apprentissage des cartes topologiques est constitué de plusieurs phases. Pour chaque exemple tiré, on considère la référence la plus adéquate. On ajuste alors les références de ce voisinage en pondérant l'adaptation d'une référence en fonction de la proximité, dans la topologie du maillage, avec la référence la plus adéquate.

Ce principe permet à la carte topologique de conserver une certaine analogie entre la topologie du maillage et la topologie de l'espace des exemples.

On obtient une famille de règles d'adaptation:

$$\tilde{\mathbf{w}}_{i,n+1} = \tilde{\mathbf{w}}_{i,n} - \Gamma_{i,n}\, \lambda_{i,i_n^*}\, \mathbf{f}\,(\tilde{\mathbf{x}}_n, \tilde{\mathbf{w}}_{i_n^*,n}) \qquad (4\text{-}28)$$

où λ_{i,i_n^*} est fonction de la proximité, dans la topologie du maillage, de la référence d'indice i et la référence i_n^* la plus adéquate pour l'exemple $\tilde{\mathbf{x}}_n$. On note Λ la matrice des $\lambda_{i,j}$. La fonction $\mathbf{f}$ dérive d'une métrique $\mathbf{F}$ mesurant l'adéquation de l'exemple $\tilde{\mathbf{x}}_n$ avec une référence.

Dans le cas particulier où $\mathbf{f}$ dérive de la forme quadratique euclidienne, la famille de règles d'adaptation s'écrit:

$$\tilde{\mathbf{w}}_{i,n+1} = \tilde{\mathbf{w}}_{i,n} - \Gamma_{i,n}\, \lambda_{i,i_n^*}\, (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i_n^*,n}) \qquad (4\text{-}29)$$

Malheureusement, ces règles d'adaptation ne dérivent d'aucune fonction de coût, sauf dans le cas très restrictif où la matrice Λ est diagonale (on retrouve alors l'algorithme kMeans).

ii Apprentissage de cartes topologiques par descente de gradient

C'est pourquoi certaines variantes ont été introduites, où la direction de l'adaptation dépend de

---

1 **Kohonen T.:** *Self-organized formation of topologically correct feature maps* - Biological Cybernetics 43, pp 59-69 (1982)

chaque référence:

$$\tilde{\mathbf{w}}_{i,n+1} = \tilde{\mathbf{w}}_{i,n} - \Gamma_{i,n}\, \lambda_{i,i_n^*}\, (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i,n}) \tag{4-30}$$

Ces règles d'adaptation découlent de fonctions de coût de la forme:

$$J^{TM}(\mathbf{x}_n, \mathbf{W}_n) = \frac{1}{2} \sum_1 \lambda_{i,i_n^*}\, (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i,n})^2 \tag{4-31}$$

En général, ces fonctions sont continues par morceau et discontinues aux valeurs de $\mathbf{W}_n$ où la référence $\tilde{\mathbf{w}}_{i_n^*,n}$ la plus adéquate change d'identité. Pour que la continuité soit assurée, il faut que la règle de décision de i^* soit cohérente avec la fonction de coût, c'est à dire:

$$i^* = \underset{j}{\operatorname{argmin}} \sum_1 \lambda_{i,j} (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i,n})^2 \tag{4-32}$$

On se restreint donc aux fonctions de coût ayant la forme:

$$J^{TM}(\mathbf{x}_n, \mathbf{W}_n) = \min_j \sum_1 \lambda_{i,j} (\tilde{\mathbf{x}}_n - \tilde{\mathbf{w}}_{i,n})^2 \tag{4-33}$$

Cette fonction de coût est continue et C^∞ par morceaux. La convergence de cet algorithme de descente de gradient est assurée dès que (ε_n) vérifie les conditions (2-80), si la distribution est continue.

4.3 Associateurs paramétriques

4.3.1 Généralités

4.3.1.1 Définitions

Un associateur est un système où les arguments (entrées) sont associées à des résultats (sorties) de nature éventuellement différentes. On distingue traditionnellement les auto-associateurs et les hétéro-associateurs. Les associateurs constituent en fait le cas général des systèmes de traitement numériques.

Un associateur paramétrique est un associateur dont le comportement est déterminé par un jeu de paramètre. L'apprentissage (réglage) d'un tel système va consister à ajuster ses paramètres afin qu'il se comporte au mieux au regard d'un certain critère (fonction de coût). Les algorithmes de descente de gradient tels ceux mentionnés aux chapitres précédents constituent une technique

relativement générale pour effectuer un tel ajustement.

En général, l'apprentissage d'un associateur est supervisé (cf. § 4.1.1). Cependant il est possible d'entraîner un associateur paramétrique de façon non-supervisée en faisant appel à des mesures de dispersion telles que l'entropie.

Nous exposons dans ce § 4.3 des associateurs universels qui ont la capacité d'approximer n'importe quel type d'association et nous étudions en particulier leur apprentissage par descente de gradient.

4.3.1.2 Familles de fonctions paramétriques

Au sein de l'infinie variété des familles de fonctions paramétriques, on peut distinguer deux catégories.

La première est composée de fonctions paramétriques se complexifiant sans limites lors de l'apprentissage pour s'ajuster parfaitement aux données. Ce sont typiquement des fonctions utilisées pour des tâches d'interpolation. Développements limités, fractales, transformées de Fourier, modélisation par sommation de bosses en sont autant d'exemples.

La deuxième est composée de fonctions paramétriques ayant une capacité limitée. La plupart des réseaux de neurones, les modèles de Markov cachés, les statistiques paramétriques en sont autant d'exemples.

On pourrait penser que les fonctions de la seconde catégorie sont moins intéressantes en raison de leurs limitations structurelles. Bien au contraire, dans le contexte de l'apprentissage, se sont elles qui s'avèrent de loin les plus performantes. En effet, un système sans limites "apprend par cœur" et traite mal des données non apprises. La problématique de la généralisation, aujourd'hui en plein essor, a mis en lumière l'impérieuse nécessité de limiter la capacité d'un système d'apprentissage.

4.3.1.3 Prise en compte de connaissances *a priori*

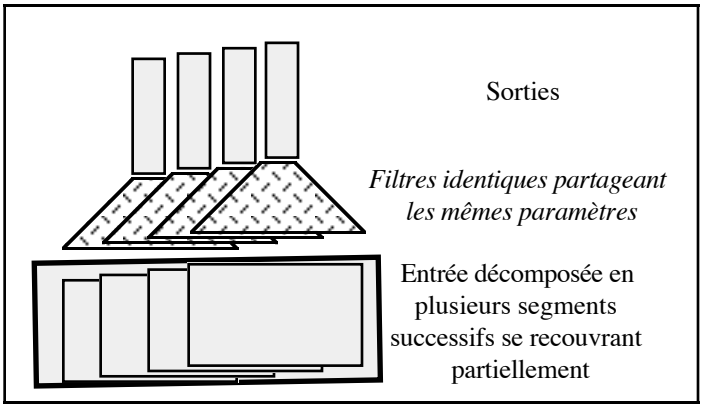
La prise en compte de connaissances *a priori* est généralement plus délicate dans le cas général des associateurs que dans le cas des plus proches voisinages dont la métrique est souvent facile à adapter.

Parmi les constituants permettant de prendre en compte l'information *a priori*, (4.1.1) deux ont un rôle relativement spécifiques aux associateurs: l'architecture et la fonction d'erreur utilisée comme critère d'optimisation lors de l'apprentissage.

On ne peut étudier les architectures qu'une fois déterminée la nature d'un système. Au contraire, la fonction d'erreur est une notion générale applicable à tout système d'apprentissage par optimisation.

i Utilisation de l'architecture et du partage de paramètres

L'architecture permet d'implanter des connaissances *a priori* dans un associateur paramétrique. Une des méthodes les plus couramment employée est celle du partage de paramètres, permettant de concevoir des associateurs dotés de certaines régularités internes.



Partage de paramètres

Cet associateur paramétrique se décompose en plusieurs filtres similaires traitant différents segments de données.
Le mécanisme de partage de paramètres permet d'imposer certaines propriétés telles que des invariances spatiales.

De nombreux associateurs possèdent des régularités structurelles qui peuvent donner sens à des partages de paramètres et traduire des invariances de la distribution sur laquelle le système est optimisé [1]. Ainsi les MLP employés pour la reconnaissance de caractères possèdent des couches inférieures qui sont constituées par le déploiement planaire de blocs élémentaires de traitement, chaque bloc traitant un petit contexte graphique. Il est dès lors possible de partager les paramètres de ces blocs, de façon à rendre identique les traitements qu'ils effectuent. Ce procédé de partage de paramètres est très employé pour traduire des régularités notamment par translation ou par symétrie. Nous détaillerons au § 5.4.2 les TDNN, qui sont des MLP incorporant des invariances temporelles.

ii Utilisation de la fonction d'erreur

La fonction d'erreur permet d'implanter des connaissances *a priori* dans un associateur paramétrique. Le choix de la fonction d'erreur est essentiel, puisque l'apprentissage tout entier a pour objectif de minimiser la fonction de coût qu'elle engendre.

Cette fonction d'erreur a pour objet premier de traduire l'imperfection d'une erreur. Elle permet

---

1 **Le Cun Y.:** *Generalization and Network Design Strategies* - Connectionnism in Perspective, Zurich, Switzerland Elsiever (1989)

également de guider le système en lui fournissant de précieuses indications. Ainsi, elle permet d'inculquer au système certaines propriétés d'invariances [1], certaines contraintes d'intégrité [2], certains indices de supervisation [3].

Enfin, elle peut incorporer des termes "de régularisation" dont le seul but est de réduire la capacité d'apprentissage du système, ce qui conduit à une dégradation des performances du système sur la distribution empirique utilisée lors de l'apprentissage mais permet souvent, d'après la théorie de la généralisation, d'augmenter les performances du même système sur la distribution totale.

4.3.1.4 Quelques fonctions d'erreur

Nous avons détaillé quelques fonctions d'erreur à l'occasion des chapitres précédents, en particulier au chapitre 3 consacré aux critères d'apprentissage des systèmes de classification.

Choisir une fonction d'erreur n'a de sens que vis-à-vis d'une tâche particulière.

Nous présentons ici plusieurs fonctions d'erreurs parmi les plus employées.

i Fonctions d'erreurs usuelles

Erreur quadratique L_2^2

Nous avons déjà évoqué cette fonction d'erreur au § 3.3.2.1, dans l'optique de la classification.

L'erreur quadratique, ou carré de la norme L2, est la fonction d'erreur la plus employée pour optimiser les associateurs paramétriques. La fonction de coût qu'elle engendre s'écrit:

$$J(\mathbf{x}) = J(\tilde{\mathbf{x}},\mathbf{y}) = | \mathbf{f}(\tilde{\mathbf{x}})-\mathbf{y} |^2 \qquad (4\text{-}34)$$

Quand la distribution est sans ambiguïté, cette fonction de coût est optimisée par l'association parfaite entre les arguments (entrées) $\mathbf{x}$ et les résultats (sorties) $\mathbf{y}$. Dans le cas général, quand la distribution est susceptible d'être ambiguë (plusieurs résultats possible pour un même argument), l'associateur minimisant absolument cette fonction d'erreur s'écrit (3-44):

---

1 **Simard P.:** *Tangent prop: a formalism for specifying selected invariances in adaptive networks* - Advances in Neural Information Processing Systems 4 (1992)

2 **Schwenk H., Gallinari P., Driancourt X. :** *a modular system which improves the topological maps* - IEEE IJCNN 92, Baltimore (1992)

3 **Suddhart S., Holden A.:** *symbolic neural systems and the use of hints for developping complex systems* - Int. J. Machine Studies 35 pp 291 (1991)

$$\mathbf{f}^*(\tilde{\mathbf{x}}) = E\{\ \mathbf{y} \mid \mathbf{x}\ \} \qquad (4\text{-}35)$$

Cette propriété remarquable reste cependant théorique. En effet, rien ne prouve que l'associateur paramétrique avec lequel on travaille est capable de vérifier cette égalité, ni que l'algorithme d'optimisation que l'on utilise va converger vers ce minimum absolu plutôt que dans un minimum local de l'espace des paramètres de l'associateur.

Cette fonction d'erreur est utilisée dans de nombreux systèmes: MLP classique, régression linéaire…

Normes Lp, $p \neq 2$

Les formes L_p^p qui correspondent aux normes Lp sont utilisées comme fonction d'erreur dans un esprit proche de l'erreur quadratique L_2.

Cependant, ces fonctions ne conduisent pas à la remarquable propriété (4-35) de l'erreur quadratique. Quand $p > 2$, elles favorisent l'uniformité spatiale de l'erreur. Quand $p < 2$, elles favorisent la concentration spatiale de l'erreur.

C'est précisément pour bénéficier de ces propriétés qu'elles sont utilisées. Il peut être utile, dans une tâche de compression d'image, de concentrer les erreurs sur les contours, toujours confus, des objets présent dans l'image plutôt que sur les parties uniformes. Inversement, dans le domaine financier, il peut être utile de déterminer une tendance avec fiabilité particulière quand une valeur est agitée (quand on prend des décisions), quitte à dégrader un peu le comportement du système dans les périodes de calme (où on ne fait rien).

ii Modifications usuelles des fonctions de coûts

Pondération des sorties

La pondération des sorties (résultat de l'association) permet d'accorder plus d'importance à la précision de certaines sorties qu'aux autres sorties, relativement à leur présence dans la distribution empirique. Cela permet de corriger artificiellement l'effet d'une base d'apprentissage biaisée, ou encore d'implanter l'inégale importance attribuée à la précision de certaines sorties.

Additifs de régularisation

Plusieurs techniques ont été proposées, visant à régulariser l'apprentissage d'associateurs paramétriques. Rappelons que la régularisation consiste à amoindrir les performances d'un système sur sa distribution empirique d'apprentissage en l'appauvrissant, dans l'espoir de le rendre plus robuste et d'améliorer ses performances sur la distribution théorique. On mesure la

réalité de cet espoir sur une distribution de régularisation, indépendante de la première.

Souvent, ces méthodes consistent à agir directement sur la fonction de coût en lui ajoutant un terme visant à limiter le domaine où les poids sont susceptibles d'évoluer, en les "attirant" dans une zone restreinte.

Dans ce cadre, une méthode simple et efficace consiste à adjoindre à la fonction de coût une pondération des normes quadratiques de paramètres choisis parmi l'ensemble des paramètres appris.

$$\mathbf{J}^{\text{Rég}}(\mathbf{x}_n,\mathbf{w}_n) = \mathbf{J}(\mathbf{x}_n,\mathbf{w}_n) + \lambda \bullet (\mathbf{w}_n * \mathbf{w}_n) \qquad (4\text{-}36)$$

où λ est un vecteur de pondérations. Cette méthode a été notamment appliquée avec succès aux réseaux de neurones multicouches MLP [1].

Allusions (Hints)

Introduite par S. Sudhart [2], l'allusion consiste à superviser l'apprentissage en prenant en compte une sortie de supervision en plus de la sortie opérationnelle classique.

Par exemple, quand on cherche à classifier l'image d'un objet, on peut superviser par allusion la localisation de ses constituants, ou encore l'orientation de l'objet par rapport à l'observateur.

Invariances locales

Plaçons-nous dans le cas de la classification. En général, au voisinage d'une forme, le message porté reste identique. On peut donc tenter de favoriser la stabilité spatiale des traitements effectués par l'associateur, en ajoutant à la fonction d'erreur un terme tel que le carré du gradient [3].

Mais cette méthode ne tient pas compte du fait que certaines transformations tendent à perturber le message plus vite que d'autre. Aussi ajouter un terme favorisant la stabilité des traitements par rapport à certaines transformations spécifiques, conduit à de bien meilleures performances

---

1 **Guyon I., Vapnick V. N, Boser B. E., Bottou L., Solla S. A.:** *Structural risk minimization for character recognition* - Advances in Neural Processing Systems 4, Morgan Kaufman (1992)

2 **Suddhart S., Holden A.:** *symbolic neural systems and the use of hints for developping complex systems* - Int. J. Machine Studies 35 pp 291 (1991)

3 **Drucker H., Le Cun Y.:** *Improving generalization performance using double backpropagation* - IEEE Transaction on Neural Networks (1992)

[1]. Ce principe permet d'incorporer des connaissances *a priori* complexes. Ainsi, un système de reconnaissance de caractères gagnera à prendre en compte l'invariance du message écrit face à des transformations telles que des translations, dilatations, affinités, petites rotations… Ces transformations, hautement non-linéaires dans l'espace des pixels, peuvent être approchées linéairement de façon locale (cf. § 4.2.2.1.ii).

4.3.2 Réseaux neuronaux à propagation

4.3.2.1 Généralités

Les réseaux à propagation sont aujourd'hui parmi les méthodes d'association paramétriques les plus populaires. Quelques catégories de réseaux à propagation efficaces ont été développées depuis la fin des années quatre-vingt, les plus connues étant les perceptrons multicouches (Multi-Layer perceptrons ou MLP) et les fonctions radiales de base (Radial Basis Functions ou RBF).

i Définitions

Un réseau de neurones est fondamentalement un graphe d'unités fonctionnelles élémentaires appelées neurones. Il existe différents types d'unités, de connectivités et de règles de réévaluation. Un réseau de neurones peut contenir des unités de différents types; cependant un réseau ne contient généralement, pour des raisons de simplicité, qu'un ou deux types de neurones.

Quand le graphe des unités fonctionnelles est orienté et sans cycle, on parle de réseaux à propagation. Ce terme sous-entend généralement l'usage de la règle de réévaluation synchrone respectant l'ordre partiel des unités dans le graphe. On peut alors définir des classes d'équivalence d'unités fonctionnelles de même ordre partiel. On donne généralement le nom de couches à ces classes d'équivalence.

---

1 **Simard P.:** *Tangent prop: a formalism for specifying selected invariances in adaptive networks* - Advances in Neural Information Processing Systems 4 (1992)

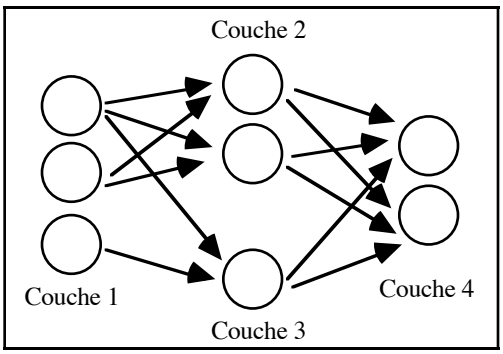


Figure 4.13
Réseau à propagation

Tout réseau à propagation découle d'un graphe orienté et sans cycle. Les unités fonctionnelles peuvent être de différents types et contenir des paramètres optimisables. Elles peuvent être regroupées en classes d'équivalences ou couches.

ii Vocabulaire

Pour chaque réseau, on appelle:

- *neurones d'entrées* l'ensemble des neurones servant à l'importation de données au sein du réseau. L'activation de ces neurones est déterminée non par le calcul mais par des données externes (formes issues d'une distribution) et constituent un module (une couche).

- *neurones de sorties* l'ensemble des neurones servant à l'exportation de données du réseau. Généralement, ces neurones servent à calculer l'erreur lors de l'apprentissage et constituent un module (une couche). Il arrive, notamment pour les tâches de compressions que les neurones de sorties changent entre la phase d'apprentissage et la phase d'utilisation opérationnelle.

- *neurones cachés* les autres neurones.

Souvent, les neurones de sortie sont des nœuds terminaux du réseau.

iii Formalisme modulaire

Il est très souvent possible de décrire un tel réseau sous la forme d'une succession de couches, la première étant constituée des neurones d'entrées, la dernière étant constituée des neurones de sorties. On parle de réseaux multicouches. Cependant ce terme est généralement attribué au seul MLP, qui fut le premier modèle efficace de réseau multicouche à propagation.

Les réseaux neuronaux multicouches du type MLP ou RBF sont constitués d'une succession de couches d'unités neuronales. Ils constituent des systèmes modulaires et peuvent être exprimés selon le formalisme introduit au § 4.1.3.

4.3.2.2 Réseaux multicouches de neurones scalaires (MLP)

i Définition fonctionnelle des couches des MLP

Les réseaux neuronaux multicouches du type MLP sont constitués d'une succession de couches d'unités neuronales scalaires. Généralement, on considère l'information se propage au sein d'un MLP simplement d'une couche vers la suivante. Cette restriction n'est pas impérative, mais elle permet d'alléger les notations et nous l'employons dans les lignes qui suivent.

La formule de propagation vers un module-couche m+1 s'écrit alors:

$$\mathbf{x}_{m+1} = \mathbf{f}^{m+1}_{\mathbf{W}_{m+1}}(\mathbf{x}_m) = \mathbf{t}_{m+1} \angle (\mathbf{W}_{m+1} \bullet \mathbf{x}_m) \qquad (4\text{-}37)$$

Les fonctions de transfert contenues dans les vecteurs fonctionnels $\mathbf{t}_m$ sont typiquement des sigmoïdes.

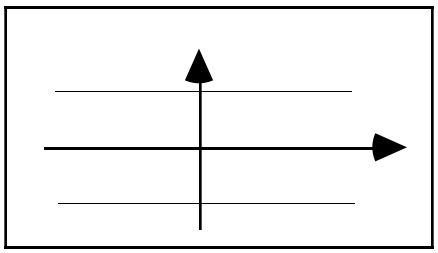


Figure 4.14
Fonction sigmoïde

La fonction sigmoïde approche la fonction à seuil. Elle est C^∞; en particulier elle est dérivable.

ii Formalisme neuronal

La formule 4-37 provient historiquement d'une modélisation très simple du fonctionnement de neurones biologiques. Cette inspiration historique est intéressante car elle met l'accent sur la connectivité de cellules qui constituent un treillis (graphe orienté sans cycle). Pour rendre compte de connaissances *a priori* telles que la topologie de l'espace des formes traitées par le système, il est possible de jouer sur cette connectivité et sur le partage de paramètres (cf. § 5.4.2), qui déterminent la famille de fonctions paramètriques qu'on utilise.

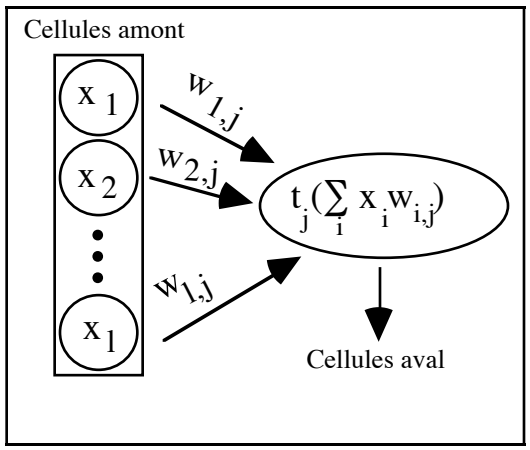


Figure 4.15
Modèle de neurone scalaire

Un neurone est une unité de calcul élémentaire. Elle est connectée à des cellules amont qui lui fournissent ses entrées x_i. Celles-ci sont agrégées (en général par un produit scalaire) en fonction des poids $w_{i,j}$ associés aux connexions. Puis une fonction logistique (généralement une sigmoïde) est appliquée pour fournir une valeur de sortie à la cellule, qui alimentera à son tour les cellules aval.

Pour chaque neurone, on appelle:

- *poids* les valeurs $\mathbf{w}_{i,j}$ associées aux connections
- *excitation* les valeurs $\sum_i \mathbf{w}_{i,j} \bullet \mathbf{x}_i$
- fonctions *logistiques* $\mathbf{t}_j$
- *activation* la valeur de sortie $t_j\left(\sum_i \mathbf{w}_{i,j} \bullet \mathbf{x}_i\right)$

iii Les réseaux multicouches d'unités scalaires sont des associateurs universels

Enoncé de la propriété

Les réseaux multicouches d'unités scalaires sont des associateurs universels de l'espace des fonctions continues de R^n dans R^p.

Plus précisément, sur un compact, deux couches de traitements (on ne compte pas la couche des entrées) dont la première utilise des fonctions de transfert sygmoïdales, suffisent pour approcher n'importe quelle fonction continue au sens de la norme L^∞, (approche uniforme) et n'importe quelle fonction L^p au sens de la norme L^p. Une précision arbitraire peut être atteinte à condition de fournir au réseau un nombre suffisant de neurones sur la couche intermédiaire (cachée) [1], [2]. Les démonstrations de cette propriété reposent sur le théorème de

---

1 **Hornik K., Stinchcombe H. W.:** *Multilayer feedforward networks are universal aproximators* - Neural Networks Vol 2. pp 359-366, (1989)

2 **Funahashi K. I.:** *On the approxmate realization of continuous mapping by neural networks* - Neural Networks Vol 2. pp 183-192, (1989)

Kolmogorov-Arnold-Sprecher [1], [2] ou bien sur le théorème de Stone-Weierstraß.

Critique de la propriété

Cette propriété fondamentale est bien sûr très positive. On peut cependant émettre deux remarques qui en réduisent l'utilité.

- Elle porte sur l'existence de paramètres et non sur leur obtention. En d'autres termes, on sait que, pour un degré de précision donné, il existe une architecture à deux couches de traitement et une configuration de poids telle que la fonction de transfert ainsi constituée approche la fonction continue au degré de précision donné. En revanche, on ne sait pas comment choisir une telle architecture et rien ne garanti que l'apprentissage par descente de gradient, sensible au minima locaux, permet de trouver une configuration de paramètres adéquate.

- Elle porte sur l'approximation d'une fonction continue par des familles de fonctions paramétriques de capacité arbitraire. Le plus souvent on n'approche non pas une fonction connue *a priori* mais une distribution empirique de couples entrées-sorties connus *a posteriori*. Or la théorie de la généralisation nous dit qu'il est nuisible de chercher à apprendre parfaitement une distribution empirique à l'aide d'une famille d'associateurs trop riche.

En d'autres termes, cette propriété est fondamentale, car elle nous dit que les réseaux multicouches d'unités scalaires sont susceptibles d'approcher n'importe quelle association mais elle ne constitue pas une garantie d'apprentissage et d'après la théorie de la généralisation, il est nuisible de l'utiliser pleinement.

iv Apprentissage des réseaux multicouches

Les techniques d'apprentissage généralement employées pour les réseaux multicouches sont des techniques de descente du gradient calculé par rétropropagation (gradient back-propagation, ou GBP).

Les perceptrons multicouches avec apprentissage par descente du gradient de l'erreur quadratique

---

1 **Kolmogorov A. N.:** *On the representation of continuous functions functions of may variables by superposition of continuous functions of one varible and addition* - Doklady Akademii Nauk SSR 144, pp. 679-681 (1957)

2 **Sprecher D. A.:** *On the structure of continuous functions of several variables* - Transactions of the American Mathematical Society, 115, pp 340-355 (1965)

calculé par rétropropagation (MLP-GBP) ont été introduit par Rumelhart [1]. le Cun qui travaillait sur un modèle proche en fournit, en compagnie de Bottou, une version très efficace [2]. Aujourd'hui, le MLP-GBP est la technique neuronale la plus populaire et la plus efficace pour une large gamme de problèmes*/; il en a été fourni de nombreuses variantes.

L'algorithme de rétropropagation du gradient, que nous avons étudié dans un cadre général (cf. § 4.1.3) s'écrit dans le cas de l'erreur quadratique en sortie des réseaux multicouches:

Initialisation de la rétropropagation du gradient (cf. 4.7a):

$$\nabla_{\mathbf{x}_M} J\ (\mathbf{x}_0, \mathbf{W}_1^M) = (\mathbf{x}_M - \mathbf{d}_M) \tag{4-38a}$$

Itération descendante (cf. 4.7b):

$$\nabla_{\mathbf{x}_m} J(\mathbf{x}_0, \mathbf{W}_1^M) = \mathbf{W}_m \bullet \left(\nabla_{\mathbf{x}_{m+1}} J(\mathbf{x}_0, \mathbf{W}_1^M) \times \mathbf{t}'_{m+1} \angle(\mathbf{x}_{m+1}) \right) \tag{4-38b}$$

Calcul du gradient (4.7c):

$$\nabla_{\mathbf{W}_{m+1}} J(\mathbf{x}_0, \mathbf{W}_1^M) = \mathbf{x}_m \bullet \left(\nabla_{\mathbf{x}_{m+1}} J(\mathbf{x}_0, \mathbf{W}_1^M) \times \mathbf{t}'_{m+1} \angle(\mathbf{x}_{m+1}) \right) \tag{4-38c}$$

Ce calcul du gradient local est ensuite utilisé par un algorithme d'apprentissage tel que ceux présentés au § 2.

4.3.2.2 Réseaux de fonctions de bases radiales (RBF)

Considérons des gausiennes paramètriques, dont la forme générale s'écrit:

$$f^{Gaus}(\mathbf{x},(\lambda,\mathbf{w},\mathbf{W})) = \lambda\ e^{-\ (\mathbf{x}-\mathbf{w})^T \mathbf{W}^{-1} (\mathbf{x}-\mathbf{w})} \tag{4-39}$$

On impose différentes contraintes à $\mathbf{W}$ permettent de contrôler la capacité d'apprentissage du système: multiple de l'identité (un seul paramètre), diagonale (n paramètres) ou symétrique définie positive (de l'ordre de $\frac{n^2}{2}$ paramètres).

En fait $\mathbf{W}$ est typiquement une matrice d'autocorrélation des vecteurs la distribution (cas symétrique) ou de chacun de leurs coefficients (cas diagonal) et $\mathbf{w}$ les moyennes de chaque

---

1 **Rumelhart D.E., Hinton G.E., Williams R.J.:** *Learning representations by back-propagating errors* - Nature 323:533-536 (1986)

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

coefficient. Alors f^{Gaus} est une simple fonction de la corrélation de $\mathbf{x}$ aux données qui ont conduit à la définition de $\mathbf{W}$ et $\mathbf{w}$.

Les fonctions radiales de bases [1], au sens restreint sont les fonctions de la forme 4-39. Au sens large, elles sont la combinaison de ces fonctions et de fonctions quasi-linéaires telles que celles utilisées dans les MLP (cf. § 4.3.2.2).

$$f^{RBF}(\mathbf{x},(\lambda,\mathbf{w},\mathbf{W})_{1\le i\le I}) = f^{MLP}\left(\left(\lambda_i \, e^{-(\mathbf{x}-\mathbf{w})^T \mathbf{W}^{-1}(\mathbf{x}-\mathbf{w})} \right)_{1\le j\le J} \right) \qquad (4\text{-}40)$$

Les fonctions radiales de bases sont généralement optimisées par des algorithmes de descente de gradient éventuellement stochastiques. Le gradient est calculé par rétropropagation (cf. § 4.1.3.3) à travers la couche gaussienne et la couche quasi-linéaire.

4.4 Optimisation de systèmes modulaires

4.4.1 Les systèmes modulaires

4.4.1.1 Définition

Dans l'acceptation la plus large du terme "modulaire", tous les systèmes de traitement numériques appliqués au monde industriel sont modulaires; on y trouve toujours un module d'acquisition des données, un module de traitement numérique, un module de présentation des résultats.

L'évolution de la puissance des ordinateurs permet d'envisager dans un proche futur l'optimisation globale des chaînes de traitements. Pour cela, il est nécessaire de mettre au point des outils algorithmiques et d'acquérir la maîtrise du comportement des systèmes modulaires et de leur optimisation.

Nous n'utiliserons le terme "système modulaire" que pour désigner une succession de modules *dont on envisage la mise au point conjointe*.

---

1 **Poggio T., Girosi F.:** *A theory of Networks for approximation and learning* - IEEE vol. 78, no 9, pp 1481-1496 (1990)

4.4.1.2 Catégories

On peut distinguer plusieurs catégories de modularité et en particulier deux extrêmes [1]: les systèmes modulaires par aiguillage (i.e. par arbre) et les systèmes modulaires par enchaînement.

i Systèmes modulaires par enchaînement

On peut également imaginer des systèmes par enchaînement où les différents modules filtrent successivement l'information. Un enchaînement peut être en ligne ou en treillis.

Il nous est apparu très tôt que les perceptrons multicouches avaient d'excellentes aptitudes en reconnaissance des formes. Cependant, ils souffrent de leur architecture figée et de leur incapacité à prendre en compte la dynamique temporelle, même avec des améliorations tels que le partage de poids des TDNN (cf. § 5.4.2). Aussi, nous avons étudié les possiblités de combiner programmation dynamique (cf. § 5.2) et perceptrons multicouches en un même système.

L'essentiel de l'effort de notre travail de recherche a porté sur les systèmes modulaires par enchaînement. Nous avons ainsi mis au point des systèmes originaux [2], [3], [4], [5] tels que les MLP-plus proches voisins (cf. § 4.4.2) et surtout MLP-DP-plus proches voisins (cf. § 5.5.1). C'est pour faciliter leur étude que nous avons tant détaillé:

- les principes de l'optimisation par descente de gradient stochastique (cf. § 2),
- le calcul du gradient par rétropropagation (cf. § 4.1.3.3),
- la formalisation unifiée de nombreux systèmes adaptatifs comme des systèmes d'optimisation.

En effet, les systèmes modulaires par enchaînement peuvent généralement être optimisés par les

---

1 **Bollivier M. de:** *Incorporation de la modularité dans les réseaux connexionnistes par décomposition de tâche* - Thèse de doctorat, Université Paris XI, Orsay , pp 73-74 (1992)

2 **Driancourt X., Bottou L.:** *TDNN Extracted Features* - Actes de Neuronimes, Nimes (1990

3 **Driancourt X., Bottou L., Gallinari P.:** *Multi-Layer Percpetron, Learning Vector Quantization and Dynamic Programming: Comparison and Cooperation* - Proceedings of IJCNN (1991)

4 **Driancourt X., Gallinari P.:** *A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer perceptron* - Proceedings of ICASSP (1992)

5 **Schwenk H., Gallinari P., Driancourt X. :** *a modular system which improves the topological maps* - Proceedings of IJCNN (1992)

techniques de descente de gradient [1]. Ils bénéficient alors d'algorithmes calculatoires tels que la rétropropagation du gradient (cf. § 4.1.3), et d'outils théoriques tels le théorème de convergence du chapitre 2.

Ils requièrent souvent quelques précautions pour la spécification de leur fonction de coût (cf. codage ponctuel, § 4.4.2.3).

ii Systèmes modulaires par aiguillage

On peut imaginer des systèmes modulaires par aiguillage où différents modules orchestrent le flux des données.

C'est le cas des systèmes de classification par arbre [2] où des classifications successives de plus en plus précises activent en cascade certains modules de plus en plus spécialisés pour résoudre une ambiguïtés de plus en plus réduite. Chaque module non-terminal associe chaque donnée à une méta-classe, puis active le(s) module(s) spécialisé(s) dans le traitement des données de cette méta-classe. Pour chaque donnée, la réponse du système modulaire est produite par le(s) module(s) terminal(aux) activé(s).

4.4.2 Exemples de systèmes modulaires enchaînés: la famille des MLP-plus proches voisinages

Les plus proches voisinages simples (e.g. euclidien) souffrent d'une faiblesse. Quand le problème est complexe et/ou de grande dimension, un nombre considérable (combinatoire) d'exemples est requis pour que le système soit performant. Certes, des algorithmes tels que kMeans ou LVQ permettent de rendre ces systèmes de classification plus robustes, mais l'essentiel du problème subsiste.

Deux solutions semblent envisageables pour rendre un système de plus proche voisinage efficace pour les problèmes complexes. La première consiste à utiliser une métrique adaptée au problème en introduisant des connaissances *a priori* (cf. 4.2.2.1.ii). La seconde consiste à utiliser un codage intelligent des formes afin que la distance euclidienne entre formes codées ait

---

1 **Bottou L., Gallinari P.:** *A Framework for the cooperation of Learning Algorithms* - Advances in Neural Information Processing Systems 3, pp 781-788, Morgan-Kaufman. Denver (1990)

2 **Quinlan J.R.:** *Induction of Decision Trees* - Machine Learning, vol 1, n°1, pp 81-106 (1986)

un sens [1].

Cette deuxième solution conduit directement à l'emploi de systèmes modulaires composés d'un module de codage (filtre) suivit d'un module de plus proche voisinage simple (e.g. euclidien). Nous étudions ici deux familles de systèmes utilisant le MLP comme module filtre et des 1-NN (ou des cartes topologiques) comme module de plus proche voisinage.

4.4.2.1 Motivation expérimentale

Des études empiriques ont montré que les MLP entraînés pour des problèmes de classification à l'aide du critère MSE, séparaient progressivement de couche en couche les distributions des formes classifiées. Des projections optimales par ACP (analyse en composante principale, ou principal component analysis) laissent penser qu'un système de plus proche voisin peut être mieux à même de classifier les formes issues des avant-dernières couches des MLP [2].

Cette motivation expérimentale vient s'ajouter aux avantages attendus de l'optimisation globale de systèmes modulaires comme nous l'avons exposé plus haut.

4.4.2.2 Modules entraînés successivement

Les premières expériences que nous avons menées portaient sur des systèmes sous-optimaux: un MLP entraîné normalement se voyait décapité afin de servir de module de pré-traitement à un système de plus proche voisinage classique (entraîné par kMeans, LVQ…).

La fonction de classification par MLP-1NN s'écrit:

$$\mathbf{f}^{\text{MLP-1NN}}(\mathbf{x},\mathbf{W}^{\text{MLP}},\mathbf{W}^{\text{1NN}}) = \underset{i}{\text{Arg Min}}\ d^{\text{1NN}}(\ \mathbf{f}^{\text{MLP}}(\mathbf{x},\mathbf{W}^{\text{MLP}}),\ \mathbf{W}_i^{\text{1NN}}\ ) \qquad (4\text{-}41)$$

où d est généralement la forme quadratique canonique (carré de la distance euclidienne).

Les combinaisons sous-optimales MLP-LVQ se sont montrées aussi performantes que les meilleurs MLP. Néanmoins, l'apprentissage sous-optimal est par nature insatisfaisant…

---

1 **Driancourt X., Bottou L.:** *TDNN Extracted Features* - Actes de Neuronimes, Nimes (1990)

2 **Bennani Y.:** *Approches connectionistes pour la reconnaissance automatique du locuteur: modelisation et identification* - Thèse de doctorat, pp 210, Université de Paris XI (1992)

4.4.2.3 Module entraînés simultanément

Il faut alors déterminer des techniques pour entraîner les deux modules de façon combinée afin d'obtenir un système optimal au sens de la théorie de l'optimisation par descente de gradient.

i Principe général

Comme nous l'avons déjà expliqué au § 4.1.3.3, l'algorithme de rétropropagation du gradient à travers les systèmes modulaires fournit un moyen algorithmiquement efficace pour calculer le gradient stochastique qui est nécessaire aux algorithmes d'optimisation dérivative.

Il faut choisir une fonction de coût. Comme nous allons le voir (cf. ii), il est bon de prendre quelques précautions théoriques. En effet, de nombreuses fonctions de coût conduisent à des aberrations.

ii Exemples d'aberrations: codage ponctuel engendrés par la plupart des optimisations de filtres-plus proches voisins, notamment MLP-KMeans et MLP-LVQ

Principe

Le phénomène de codage ponctuel est une forme de dégénérescence de nombreux systèmes dont le dernier module est de type plus-proche voisinage [1], [2]. Les autres modules constituent un filtre s'appliquant aux entrées du système au profit du dernier module. Dans les lignes qui suivent, nous appellerons *codages* les sorties du filtre (i. e. les entrées du dernier module).

La plupart des fonctions de coût usuelles incitent le système à dégénérer de la façon suivante (codage ponctuel):

- le filtre projette l'espace des entrées sur un unique point de l'espace de codage,
- le plus proche voisinage concentre toutes ses références sur le même point de l'espace de codage.

D'autres fonctions de coût, plus exotiques, incitent le système à dégénérer de façon inverse (codage explosé):

---

1 **Driancourt X., Gallinari P.:** *A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer perceptron* - Proceedings of ICASSP (1992)

2 **Schwenk H., Gallinari P., Driancourt X. :** *a modular system which improves the topological maps* - IEEE IJCNN 92, Baltimore (1992)

- le filtre disperse les entrées dans l'espace de codage loin des références du plus proche voisinage,

- le plus proche voisinage disperse ses références loin des codages.

Nous allons d'abord montrer que ces dégénérescences optimisent la plupart des fonctions de coûts. Puis, nous allons montrer qu'il existe un chemin de dégénérescence partant de toute configuration de paramètres. La démonstration sera faite dans le cas du codage ponctuel, le cas du codage explosé sera discuté après.

Formalisation

Considérons un critère d'optimisation $J^{FLT\text{-}ONN}$, s'appliquant à notre système composé d'un filtre (FLT) et d'un plus proche voisin (NN) de métrique d^{NN} quelconque. La fonction de coût $J^{FLT\text{-}ONN}$ s'écrit:

$$J^{FLT\text{-}ONN}(\mathbf{x},\mathbf{W}^{FLT},\mathbf{W}^{NN}) =$$

$$f^{NN}\Big(\Big[d^{NN}(\mathbf{f}^{FLT}(\mathbf{x},\mathbf{W}^{FLT}), \mathbf{W}^{NN}_i) \Big]_{1\le i\le I} \Big) \qquad (4\text{-}42)$$

où f^{NN} est une fonction croissante d'une façon quasi-stricte, c'est-à-dire:

$$\exists\, \varepsilon_0 \in]0,1[,\ \forall\, \varepsilon \in [0,\varepsilon_0[,\ \forall\, \delta_1,\delta_2,\ldots,\delta_I \text{ tq: } f^{NN}(\delta_1,\delta_2,\ldots,\delta_I) > 0 :$$

$$f^{NN}(\delta_1\bullet(1-\varepsilon), \delta_2\bullet(1-\varepsilon),\ldots, \delta_I\bullet(1-\varepsilon)) \ < \ f^{NN}(\delta_1, \delta_2,\ldots, \delta_I) \qquad (4\text{-}43a)$$

où d^{NN} est une métrique continue vérifiant:

$$\exists\, \varepsilon_0 \in]0,1[,\ \forall\, \varepsilon \in [0,\varepsilon_0[,\ \forall\, \eta,\ \exists\, \eta' \in]0,1[,\ \forall\, \mathbf{y},\mathbf{y}',\mathbf{v} :$$

$$d^{NN}(\mathbf{y},\mathbf{y}) = 0 \quad \text{et} \quad d^{NN}((1-\eta)\mathbf{y}+\eta\mathbf{v},(1-\eta')\mathbf{y}'+\eta'\mathbf{v}) = (1-\varepsilon)\, d^{NN}(\mathbf{y},\mathbf{y}') \qquad (4\text{-}43b)$$

et où $\mathbf{f}^{FLT}$ vérifie:

$$\forall\, \mathbf{W}^{FLT}\ \exists\, \mathbf{v}\ \forall\, \xi,\eta_0 \in]0,1[\ \exists\, \eta \in [0,\eta_0[\ \exists\, \mathbf{W}'^{FLT}\ \forall\, \mathbf{x} :$$

$$|\, \mathbf{W}^{FLT} - \mathbf{W}'^{FLT}\, | < \xi \quad \text{et} \quad \mathbf{f}^{FLT}(\mathbf{x},\mathbf{W}'^{FLT}) = (1-\eta)\, \mathbf{f}^{FLT}(\mathbf{x},\mathbf{W}^{FLT}) + \eta\, \mathbf{v} \qquad (4\text{-}44)$$

Concrètement l'hypothèse (4-43a) et l'hypothèse (4-43b) signifient que le critère d'optimisation est sensible aux contractions uniformes des codages et des références. Ces hypothèses, assez souples, sont vérifiées par la plupart des métriques et des systèmes d'optimisation de plus proche voisinage (KMeans, LVQ…).

L'hypothèse (4-44) signifie que pour toute configuration de paramètres, il existe un point

autour duquel le filtre est capable, de façon homogène, de contracter un peu son espace de sortie. Cette propriété est vérifiée par un grand nombre de filtres, et notamment ceux composés d'un enchaînement de modules dont le dernier est une fonction affine de ses entrées.

Existence de minima absolus aberrants

On cherche à optimiser $\mathbf{W}^{FLT}$ et $\mathbf{W}^{NN}$ simultanément. Or, il existe généralement des configurations $(\mathbf{W}^{FLT}, \mathbf{W}^{NN})$ annulant ce coût. En effet, d'après les hypothèses (4-42) et (4-43):

$$\begin{cases} \forall\ \mathbf{x} : \mathbf{f}^{FLT}(\mathbf{x},\mathbf{W}^{FLT}) = \mathbf{v} \\ \forall\ i : \mathbf{W}^{NN}_i = \mathbf{v} \end{cases} \Rightarrow J^{FLT\text{-}ONN}(\mathbf{x},\mathbf{W}^{FLT},\mathbf{W}^{NN}) \equiv 0 \qquad (4\text{-}45)$$

Alors la fonction de coût est parfaitement optimisée. Pourtant le système obtenu est complètement dénué d'intérêt: toutes les classes (et toutes les références) obtiennent le même score… Les hypothèses de la relation (4-45) découlent de l'hypothèse (4-44).

Existence de chemins menant aux minima absolu aberrants

Il reste à établir dans quelle mesure la procédure d'optimisation conduit effectivement aux minima locaux aberrants. Pour cela, nous allons établir qu'au voisinage de toute configuration de paramètres, on peut réduire infinitésimalement le coût sauf si celui-ci est déjà nul.

Posons:

$$\mathbf{V} = (\mathbf{v},\ldots,\mathbf{v}) \text{ puis: } \mathbf{W}'^{NN} = (1-\eta')\,\mathbf{W}^{NN} + \eta'\,\mathbf{V} \qquad (4\text{-}46)$$

Les équations qui suivent ne font pas apparaître de quantificateurs mais font référence aux hypothèses (4-42), (4-43), (4-44). En fait, les quantificateurs de ces hypothèses servent simplement a déclarer les propriétés d'uniformité nécessaires aux lignes qui suivent. Nous ne les rapportons pas afin d'alléger les notations.

La notation (4-42) s'écrit:

$$J^{FLT\text{-}ONN}(\mathbf{x},\mathbf{W}'^{FLT},\mathbf{W}'^{NN})$$

$$= f^{NN}\Big(\big[d^{NN}(\mathbf{f}^{FLT}(\mathbf{x},\mathbf{W}'^{FLT}), \mathbf{W}'^{NN}_i) \big]_{1\le i\le I} \Big) \qquad (4\text{-}47a)$$

puis l'hypothèse (4-44) :

$$= f^{NN}\Big(\big[d^{NN}((1-\eta)\mathbf{f}^{FLT}(\mathbf{x},\mathbf{W}^{FLT})+\eta\mathbf{v},\ (1-\eta')\mathbf{W}^{NN}_i+\eta'\mathbf{v}) \big]_{1\le i\le I} \Big) \qquad (4\text{-}47b)$$

puis l'hypothèse (4-43b) :

$$= f^{NN}\Big(\big[(1-\varepsilon)\, d^{NN}(\mathbf{f}^{FLT}(\mathbf{x},\mathbf{W}^{FLT}), \mathbf{W}_i^{NN}) \big]_{1\le i\le I} \Big) \quad (4\text{-}47c)$$

puis l'hypothèse (4-43a) :

$$< J^{FLT\text{-}ONN}(\mathbf{x},\mathbf{W}^{FLT},\mathbf{W}^{NN}) \quad (4\text{-}47d)$$

Au total:

$$\boxed{\begin{array}{l}\forall\, \mathbf{W}^{FLT}, \mathbf{W}^{NN} \text{ si } J^{FLT\text{-}ONN}(\mathbf{x},\mathbf{W}^{FLT},\mathbf{W}^{NN}) \text{ alors } \forall\, \xi>0\ \exists\, \mathbf{W}'^{FLT}, \mathbf{W}'^{NN} : \\ \quad |\, \mathbf{W}^{FLT} - \mathbf{W}'^{FLT}\, | < \xi \text{ et } J^{FLT\text{-}ONN}(\mathbf{x},\mathbf{W}'^{FLT},\mathbf{W}'^{NN}) < J^{FLT\text{-}ONN}(\mathbf{x},\mathbf{W}^{FLT},\mathbf{W}^{NN})\end{array}} \quad (4\text{-}48)$$

En d'autre terme, aucune configuration de paramètres n'est minimum local de $J^{FLT\text{-}ONN}$ sauf celles qui l'annulent. A partir de toute configuration de paramètre, il est possible de réduire un peu la fonction de coût en contractant un peu l'espace de sortie du filtre tout en contractant un peu les références du module de plus proche voisinage.

Les propriétés de convergences presque sûres, applicables à un grand nombre de systèmes de la forme $J^{FLT\text{-}ONN}$ (4-42) nous permettent de conclure que ces systèmes vont converger vers une fonction de coût nulle.

Parfois, cela peut signifier un système se comportant parfaitement sur la distribution empirique utilisée lors de l'apprentissage. Le plus souvent, cela signifie simplement que l'on se retrouve dans la situation du codage ponctuel (4-45), où la procédure d'apprentissage conduit à un système dégénéré et sans intérêt.

Fonctions de coût sujettes à ces phénomènes de dégénérescence

Le problème se pose de la même façon pour toute fonction de coût usuelle favorisant la réduction de l'échelle du codage et des références. C'est le cas des fonctions de coût MLP-LVQ1, MLP-LVQ2, MLP-LVQ2bis, MLP-LVQ2ter, MLP-TMAP [1]…

La démonstration précédente n'est pas valable pour tous les MLP. Elle est valable dans le cas notable où sa dernière couche est affine; on retrouve alors les propriétés d'uniformités requises par (4-44). Nous ne connaissons pas de démonstration simple dans le cas où la dernière couche du MLP n'est pas linéaire. Cependant, le codage ponctuel se produit tout de même avec les autres MLP: les expériences confirment toutes ce phénomène de dégénérescence.

---

1 **Driancourt X., Gallinari P.:** *Empirical risk optimisation: neural networks and dynamic programming* - Proceedings of NNSP (1992)

Les fonctions de coût MLP-LVQ2, MLP-LVQ2bis, MLP-LVQ2ter sont un peu plus lentes à connaître ce phénomène que les fonctions MLP-KMeans et MLP-TMap. Quand la classification sur la base d'apprentissage est parfaite, les systèmes MLP-LVQ2, MLP-LVQ2bis, MLP-LVQ2ter cessent leur apprentissage. Il arrive donc que de tels systèmes, particulièrement bien initialisés ne dégénèrent pas car ils classifient parfaitement leur base d'apprentissage avant d'avoir laissé le temps à la dégénérescence d'intervenir [1]…

Phénomène inverse: codages explosés

Un phénomène inverse mais tout aussi aberrant se produit avec des fonctions de coûts plus exotiques quand:

$$\lim_{|x-y|\to\infty} d(x,y) = 0.$$

Des références infiniment éloignées des codages permettent alors de rendre la fonction de coût aussi petite que possible (codage explosé).

iii Une première solution: la fonction de coût MLP-LVQ2+

La fonction de coût MLP-LVQ2+, utilisée pour la clasification, s'écrit:

$$J^{\text{MLP-LVQ2+}}(\mathbf{x},\mathbf{W}^{\text{MLP}},\mathbf{W}^{\text{LVQ2+}}) = \min\left(1, \frac{1}{\delta}\left(\frac{|\mathbf{f}^{\text{MLP}}(\tilde{\mathbf{x}},\mathbf{W}^{\text{MLP}}) - \tilde{\mathbf{w}}^{+}|^2}{|\mathbf{f}^{\text{MLP}}(\tilde{\mathbf{x}},\mathbf{W}^{\text{MLP}}) - \tilde{\mathbf{w}}^{*}|^2} - 1\right)\right) \qquad (4\text{-}47)$$

Où $\mathbf{w}^*$ (resp. $\mathbf{w}^+$) représente la référence issue de $\mathbf{W}^{\text{LVQ2+}}$ la plus proche (resp. la plus proche ayant la bonne classe) de $\mathbf{f}^{\text{MLP}}(\tilde{\mathbf{x}},\mathbf{W}^{\text{MLP}})$.

Elle est indépendante de l'échelle du codage et des références. Elle n'est donc pas sujette aux problèmes du codage ponctuel.

De même que la fonction LVQ2+ (cf. § 4.2.3.1), elle approche le risque moyen (cf. § 3.2).

iv Une deuxième solution: ajouts de termes anti-dégénérescence dans la fonction de coût

Le problème du codage ponctuel, tel que posé en ii, est la contraction progressive de l'ensemble des références et des sorties du MLP. Cette contraction peut être empêchée par l'ajout de coûts

---

1 **Driancourt X., Gallinari P.:** *A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer perceptron* - Proceedings of ICASSP (1992)

portant sur la variance des sorties du MLP et l'éloignement des références les unes des autres. Nous appellerons de tels ajouts *prime d'intégrité*.

Il est possible de donner une estimation adaptative de la variance de l'image par le MLP de la distribution empirique d'apprentissage. Ceci permet de conserver son caractère adaptatif et stochastique à l'algorithme.

Nous avons mené de telles expériences avec succès [1] sur un problème synthétique connu [2]. Le système dont il s'agissait d'empêcher la dégénérescence était un MLP-TMap. L'ajout d'une *prime d'intégrité* portant sur la variance des sorties du MLP a permis à la carte topologique de ne plus se contracter. Ce système modulaire a obtenu de meilleures performances que les cartes topologiques simples. Ceci est probablement dû à la capacité du MLP à projeter utilement les données sur des plans adéquats, permettant à la topologie de la carte de Kohonnen de retrouver sa signification dans un espace de dimension plus réduit.

Ces expériences sortent du cadre de la reconnaissance de la parole. Aussi, nous avons choisi de donner en annexe 2 un exposé plus détaillé en reproduisant une de nos contributions à la conférence IJCCN 1992.

4.4.3 Exemples de systèmes modulaires par aiguillage

L'utilisation d'aiguillages au sein de systèmes modulaires permet d'incorporer certaines informations *a priori*. On peut par exemple décomposer un problème de reconnaissance du locuteur en deux étapes, la première reconnaissant la typologie du locuteur [3].

Il existe également quelques systèmes génériques de classification par arbres, notamment les systèmes ID3 [4] et CART [5]. Chaque nœud (et chaque feuille) de l'arbre fait une dichotomie dans l'espace des formes à l'aide d'une fonction discriminante linéaire. Les fonctions

---

1 **Schwenk H., Gallinari P., Driancourt X. :** *a modular system which improves the topological maps* - IEEE IJCNN 92, Baltimore (1992)

2 **Breiman L., Friedman J. H., Olshen R. A., Stone C. J.:** *Classification and Regression Trees* - Wadsworth International Group (1984)

3 **Bennani Y., Gallinari P.:** *On the Use of TDNN-Extracted Features Information In Talker Identification* - ICASSP S6.5, pp 385-388, Toronto (1991)

4 **Quinlan J. R.:** *Induction of Decision Trees* - Machine Learning, vol 1, n°1, pp 81-106 (1986)

5 **Breiman L., Friedman J. H., Olshen R. A., Stone C. J.:** *Classification and Regression Trees* - Wadsworth International Group (1984)

discriminantes des nœuds sont entrainées sur des critères d'entropie.

L'idée fondamentale de ces systèmes est de transformer un problème complexe en une multitude de problèmes simples.

4.5 Conclusion

Nous venons d'exposer quelques techniques d'utilisation de l'information *a posteriori*. Nous avons reformulé plusieurs règles d'apprentissage et d'adaptation pour leur faire prendre la forme d'une descente de gradient stochastique. Nous avons introduit une fonction de coût originale LVQ2+, qui approxime le risque moyen mais reste différentiable (ainsi que les fonctions de coût qui en sont issues, telles que MLP-LVQ2+).

Nous avons particulièrement insisté sur trois aspects: la reconnaissance des formes, l'optimisation par les techniques de gradient éventuellement stochastique et l'emploi de systèmes modulaires.

Nous avons rappelé l'usage de la rétropropagation du gradient dans le cas général des systèmes modulaires. Nous avons fourni une démonstration originale du phénomène de codage ponctuel qui affecte un grand nombre de systèmes modulaires. Nous avons proposé deux solutions originales: l'une d'entre elle consiste à utiliser des fonctions d'erreurs plus rigoureuses telles que LVQ2+, l'autre consiste à imposer des contraintes d'intégrité, empêchant la dégénérescence du système en cours d'optimisation.

Nous allons étudier dans le chapitre suivant les systèmes dédiés au traitement de l'information de nature temporelle.

5 Quelques systemes utilisant l'information temporelle *a posteriori*

A l'occasion du chapitre 4, nous avons détaillé de nombreux systèmes utilisant l'information *a posteriori*, en écartant toutefois les systèmes dédiés au traitement de l'information temporelle.

Le présent chapitre a pour objet d'exposer un grand nombre des méthodes de traitement de l'information temporelle *a posteriori*, notamment celles utilisant pour leur ajustement des techniques de descente de gradient.

Nous évoquerons au § 5.1 la nature temporelle de l'information. Au § 5.2, nous traiterons des méthodes de plus proche voisinage et notamment la métrique induite par la programmation dynamique. Conformément à notre démarche des chapitres précédents, nous étudions les possibilités d'optimisation par descente de gradient stochastique. C'est ainsi que nous proposons au § 5.2.1.2 une série d'algorithmes d'optimisation originaux: DP-KMeans, DP-LVQ2… Ces algorithmes permettent d'optimiser des références synthétiques de façon cohérente avec la métrique induite par la programmation dynamique.

Au § 5.3, nous décrirons succintement les modèles de Markov cachés discrets. Puis nous aborderons les associateurs paramétriques et notamment les TDNN et les réseaux récurrents.

Au § 5.5, nous proposerons différents systèmes modulaires par enchainements calqués sur ceux proposés au chapitre précédent.

En particulier, nous proposerons un système original optimisable globalement: le MLP-DP-LVQ2+. Ces systèmes sont originaux ; comme le confirmeront les mesures expérimentales du § 6.3 et du § 6.4, ils permettent effectivement de combiner l'efficacité des perceptrons multicouches et de la programmation dynamique.

Nous proposerons également une famille de fonctions de coût dont des fonctions de coût discriminantes PDP-LVQ2bis pour l'optimisation des systèmes modulaires prédictifs combinant MLP (pour la prédiction) et programmation dynamique (pour l'alignement temporel et la classification).

5.1 Utilisation la nature temporelle de l'information

5.1.1 La nature temporelle de l'information prise en tant que connaissance *a priori*

L'utilisation de l'information *a posteriori* est généralement un pis-aller ayant pour objet de pallier un manque d'information *a priori*. Cependant, il n'est pas toujours aisé de tenir compte de l'information *a priori* au sein de systèmes conçus pour traiter l'information *a posteriori*. Ce problème se pose notamment pour la prise en compte de la nature temporelle des données traitées.

C'est ainsi qu'ont été développés différents systèmes dédiés au traitement de l'information temporelle. Chaque système a été conçu initialement pour les besoins d'une application spécifique traitant par exemple le signal sonar, les données économiques ou la parole. Un grand nombre d'entre eux ne se fondent sur aucune autre information *a priori* que la nature temporelle des données traitées. C'est donc fort logiquement que ces systèmes ont été réutilisés avec succès pour d'autres applications que celles ayant occasionné leur conception.

La plupart des systèmes utilisant l'information *a posteriori* ont donné lieu à des variantes permettant de traiter certaines propriétés connues *a priori*, telles que les invariances par translation, homothétie, distorsions... Ces informations *a priori* peuvent faire l'objet de prétraitements ou bien de contraintes imposées au coeur des systèmes.

5.1.2 Notion de classe pour séquences de vecteurs

Dans le cas de problèmes de classification, la première tâche consiste à définir… les notions de classes et de classifications.

Ce travail est souvent délicat car il conditionne la décomposition analytique des procédures formant le système de traitement. Il est souvent possible de décomposer une classe à l'aide de la structure temporelle de la forme à laquelle elle est associée.

Prenons l'exemple de la reconnaissance du message parlé en nous limitant au cas de la conversion de l'information acoustique d'un spectrogramme (une suite temporelle composée de vecteurs fréquentiels contenant des coefficients énergétiques) en son message lexico-syntaxique.

Un spectrogramme contenant la prononciation d'un mot isolé peut avoir pour classe l'entrée lexicale de ce mot. On est alors dans un contexte proche de celui de la reconnaissance des formes "ordinaires".

Un spectrogramme contenant la prononciation d'une phrase peut avoir pour classe la suite des entrées lexicales de ce mot. Une version rigoriste consiste à associer chaque entrée lexicale à une partie précise du spectrogramme. Une version laxiste consiste à se contenter de la succession des entrées lexicales, sans position stricte au sein du spectrogramme.

Dans cet exemple, la classe d'une phrase est la suite d'entrées lexicales composant la phrase. Cependant, afin de superviser plus complètement l'apprentissage, il est possible de rendre plus astreignante la notion de classe, en étiquetant chaque vecteur du spectrogramme.

Ainsi, l'aspect temporel étend son domaine, au delà de la notion de forme, à la notion de classe.

5.2 Plus proches voisinages

La notion de proximité est naturelle pour des données de dimensions fixes. Or, ce n'est pas toujours le cas des données de nature temporelle telles que les ondes sismiques, acoustiques ou de données financières… De plus, des distorsions temporelles sont souvent sans effet sur la classe de données de nature temporelle. Ainsi, dans le domaine de la reconnaissance de la parole, des transformations telles que le ralentissement ponctuel du rythme d'élocution laissent invariant le message parlé.

Divers systèmes de plus proche voisinage ont été proposés, destinés à prendre en considération ces spécificités. Pour ce faire, ils ont la possibilité d'influer sur la métrique (notamment les métriques induites par la programmation dynamique) ou bien sur la stratégie de vote.

5.2.1 Programmation dynamique

La programmation dynamique est, fondamentalement, un algorithme de calcul de plus court chemin dans des treillis (cf. § 5.2.1.1.i). Or, les automates finis à délai engendrent des treillis (cf. § 5.2.1.1.ii). La programmation dynamique permet également de calculer des métriques entre séquences de vecteurs tout en donnant un contrôle précis sur les distortions autorisées par la dynamique temporelle du problème qu'on traite (cf. § 5.2.1.1.iii). On peut étendre le principe

à la reconnaissance de formes complexes telles que les séquences de formes élémentaires. On peut donc concevoir des systèmes de reconnaissance des formes temporelles par plus proche voisin, utilisant la métrique induite par la programmation dynamique.

La métrique induite par la programmation dynamique est différentiable presque partout par rapport à ses entrées. Nous proposons donc des systèmes originaux DP-KMeans, DP-LVQ2, DP-LVQ2+, qui permettent d'optimiser des références pour la métrique induite par la programmation dynamique.

5.2.1.1 Métriques induites par la programmation dynamique

La programmation dynamique est un algorithme qui permet de comparer efficacement deux suites de vecteurs en contrôlant l'importance accordée aux distorsions temporelles qui contribuent à les différencier.

Nous fournissons ici une description théorique structurée de la programmation dynamique.

i Calcul par programmation dynamique du chemin le moins coûteux reliant deux nœuds d'un treillis

Considérons un treillis composé de nœuds et de connexions monodirectionnelles (sans boucle). Un chemin est une succession de nœuds reliés par des connexions correctement orientées. Un nœud sans ascendance est dit nœud initial. Un nœud sans descendance est dit nœud final.

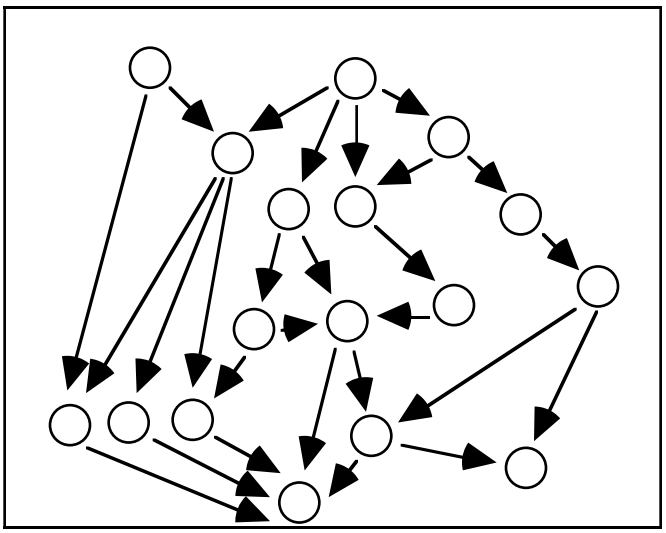


Figure 4.1
Exemple de treillis

Un treillis est un graphe orienté induisant un ordre partiel. Il est composé de nœuds et de connexions orientées.

Un chemin est une succession de nœuds et de connexions cohérente avec l'orientation des connexions du graphe.

On associe un coût élémentaire à valeurs dans $\mathbb{R}^+$ à chaque connexion et à chaque nœud. On en déduit pour chaque chemin un coût cumulatif qui est par définition la somme des coûts élémentaires des nœuds et des connexions composant le chemin. On nomme *coût cumulatif d'un nœud* le moindre des coûts cumulatifs des chemins le reliant à un nœud initial.

On cherche à calculer le chemin le moins coûteux pour relier deux nœuds (supposés respectivement initial et final) du treillis.

On connaît le coût cumulé des nœuds initiaux. On suppose les nœuds ordonnés en conformité avec l'ordre partiel induit par le treillis. En procédant parmi les nœuds *selon cet ordre*, pour chaque nœud:

- on calcule le coût cumulé d'un nœud en fonction des coûts cumulés des nœuds qui lui sont immédiatement antécédents et du coût élémentaire de leur connexion,
- on mémorise l'identité du nœud immédiatement antécédent le plus avantageux.

La complexité de cet algorithme est proportionnelle au nombre total de connexions du treillis, c'est-à-dire à la complexité du problème lui-même.

Par ailleurs, il est possible d'étendre l'algorithme afin de déterminer les K meilleurs chemins, mais le coût d'une telle extension est de multiplier le temps de calcul par K. De plus, les K meilleurs chemins sont souvent presque identiques; cette information coûteuse est généralement peu exploitable.

ii Automate fini et alignement temporel par programmation dynamique

Considérons un automate fini, constitué d'états et de relations orientées liant chacune deux états. On associe à chaque relation un délai élémentaire N positif. On admet l'existence de relations ayant un délai élémentaire N nul sous réserve qu'il n'existe aucun cycle de délai cumulé nul.

Le déploiement temporel d'un tel automate prend la forme d'un treillis tel que défini précédemment.

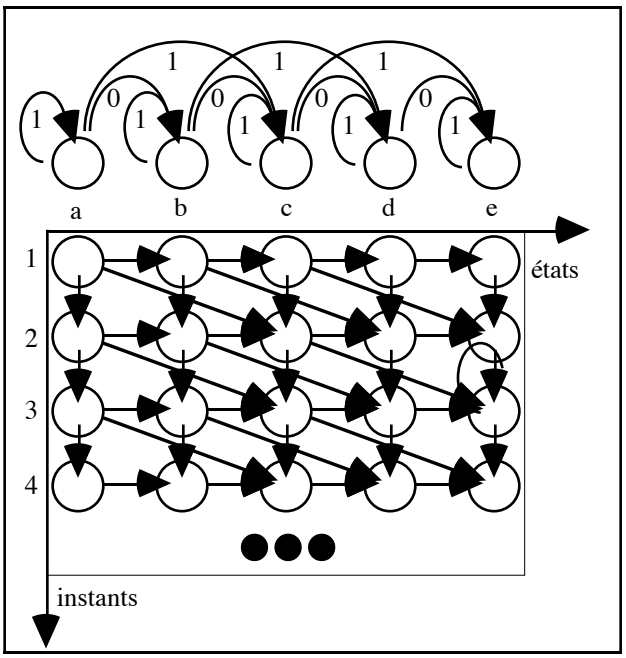


Figure 5.2
Le déploiement temporel d'un automate à délai engendre un treillis

Il existe une correspondance directe entre les automates à délais (exemple en haut de la figure) et certains treillis (exemple en bas de la figure) présentant des régularités.

Cette correspondance est obtenue en déployant dans le temps les états possibles d'un automate à délai.

Les couples état-instant définissent les nœuds du treillis et les couples relation-instant définissent les connexions du treillis.

De même que pour n'importe quel autre treillis, on peut associer un coût μ à chaque nœud et chaque connexion. La programmation dynamique permet alors de calculer le chemin le moins coûteux qui représente physiquement l'alignement temporel optimal entre les états et les instants.

Un chemin s'écrit: $p=(I,\sigma,\hat{\sigma})$, où I est la longueur du chemin et $\sigma(i)$ (resp. $\hat{\sigma}(i)$) les indices des instants (resp. des états) pour la $i^{ème}$ étape du chemin. On appelle $\mathfrak{C}$ est l'ensemble des chemins autorisés $p=(I,\sigma,\hat{\sigma})$ issus du treillis. Chaque couple état-instant $(\sigma(i)\ ,\ \hat{\sigma}(i))$ du déploiement temporel d'un automate à délai représente un nœud du treillis. Chaque connexion $(\sigma(i\text{-}1)\ ,\ \hat{\sigma}(i\text{-}1)\ ,\ \sigma(i)\ ,\ \hat{\sigma}(i))$ est associée à un coût élémentaire $\mu_{\sigma(i-1),\hat{\sigma}(i-1),\sigma(i),\hat{\sigma}(i)}$.

Le coût cumulé minimal sur l'ensemble $\mathfrak{C}$ des chemins autorisés s'écrit:

$$D^{DP}(\mu,\mathfrak{C}) = \left(\underset{\mathfrak{C}}{\text{Min}} \sum_{i=1}^{I} \mu_{\sigma(i-1),\hat{\sigma}(i-1),\sigma(i),\hat{\sigma}(i)} \right) \qquad (5\text{-}1)$$

L'algorithme de programmation dynamique permet de calculer ce coût en un temps proportionnel au nombre de connexions du treillis, c'est-à-dire proportionnel au nombre de relations du jeu d'automate, multiplié par la durée du déploiement temporel (nombre d'instants).

iii Alignement temporel de deux suites de vecteurs

Cet algorithme peut être utilisé afin de calculer une métrique entre deux suites de vecteurs **x** et

w en contrôlant l'importance accordée aux distorsions temporelles qui contribuent à les différencier. La programmation dynamique est alors employée pour aligner temporellement chacune des deux suites de vecteurs.

Utilisation de la programmation dynamique pour calculer une métrique entre deux séquences de vecteurs

On associe à chaque instant un vecteur de **x**, et à chaque état un vecteur de **w**. On note $\mathbf{X} = \mathbf{x}_1,\ldots, \mathbf{x}_{lx}$ et $\mathbf{W} = \mathbf{w}_1,\ldots, \mathbf{w}_{lw}$ nos deux suites de vecteurs. Un chemin p appariant **X** à **W** s'écrit: $(I , \sigma , \hat{\sigma})$, où l est la longueur du chemin et $\sigma(i)$ (resp. $\hat{\sigma}(i)$) les indices qu'on applique à **x** (resp. **w**) pour la $i^{ème}$ étape du chemin.

Pour chaque connexion, on peut définir un coût élémentaire prenant la forme:

$$\mu_{\sigma(i-1),\hat{\sigma}(i-1),\sigma(i),\hat{\sigma}(i)} = f\left(\lambda_{\sigma(i-1),\hat{\sigma}(i-1),\sigma(i),\hat{\sigma}(i)} , d(\mathbf{x}_{\sigma(i)},\mathbf{w}_{\hat{\sigma}(i)}) \right) \quad (5\text{-}2)$$

Les paramètres λ sont propres à chaque connexion. La fonction f est destinée à faire la part des choses entre les paramètres des connexions et les distances vecteur-à-vecteur. En général f effectue tout simplement l'addition ou la multiplication de ces arguments. Alors la métrique calculée par un DP simple s'écrit:

$$D^{DP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{P}) = \left(\underset{\mathfrak{P}}{\text{Min}} \sum_{i=1}^{I} f\left(\lambda_{\sigma(i-1),\hat{\sigma}(i-1),\sigma(i),\hat{\sigma}(i)} , d(\mathbf{x}_{\sigma(i)},\mathbf{w}_{\hat{\sigma}(i)}) \right) \right) \quad (5\text{-}3)$$

où $\mathfrak{P}$ est l'ensemble des chemins autorisés $p=(I,\sigma,\hat{\sigma})$ appariant **X** à **W**.

L'équation (5-3) peut être traduite très simplement: la métrique entre deux suites de vecteurs est fonction des métriques vecteur à vecteur le long du meilleur chemin engendré lors de l'appariement des deux suites, et fonction des coûts élémentaires des transitions. Ceci détermine l'alignement optimal de ces deux suites de vecteur.

En termes de plus proche voisinage, ce système a pour métrique D^{DP}, pour entrée **X** et pour paramètres **W** et λ.

Propriétés de la métrique induite par la programmation dynamique

En matière de stratégie de vote, ce système est restrictif puisque l'algorithme de programmation dynamique ne détermine que le meilleur chemin et non les meilleurs chemins. Comme nous l'avons déjà dit, il est possible d'étendre l'algorithme afin de déterminer les K meilleurs chemins, mais le coût d'une telle extension est de multiplier le temps de calcul par K (hors calcul des distances vecteur à vecteur qui, lui, ne change pas). De plus, les meilleurs chemins

sont souvent presque identiques. C'est pourquoi, en général, la programmation dynamique est utilisée avec la méthode du plus proche voisin mais pas avec celle des K plus proches voisins ou des fenêtres de Parzen.

Il convient également de signaler que la métrique D^{DP} n'est généralement pas une distance. Elle est réflexive si d est réflexive. En revanche, elle n'est pas nécessairement symétrique et ne respecte généralement pas l'inégalité triangulaire.

La métrique D^{DP} est continue si d est continue. De plus, si d est de classe C^n (n fois continuement dérivable) alors la métrique D^{DP} est C^n par morceaux.

Métriques alternatives engendrées par la restriction de l'ensemble $\mathfrak{C}$ des chemins autorisés

L'ensemble des chemins autorisés a une influence essentielle sur la métrique D^{DP}.

Considérons un ensemble $\mathfrak{C}$ de chemins autorisés. Considérons une partition de $\mathfrak{C}$ en sous ensembles $\mathfrak{C}_k$. On peut ainsi établir des métriques alternatives utilisant les ensembles de chemins restreints $\mathfrak{C}_k$:

$$D^{DP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}_k) = \left(\underset{\mathfrak{C}_k}{\text{Min}} \sum_{i=1}^{I} f\left(\lambda_{\sigma(i-1),\hat{\sigma}(i-1),\sigma(i),\hat{\sigma}(i)}, d(\mathbf{x}_{\sigma(i)},\mathbf{w}_{\hat{\sigma}(i)})\right)\right) \qquad (5.4)$$

Exemple 1: on peut définir pour chaque état k le sous ensemble $\mathfrak{C}_k$ composé des chemins de $\mathfrak{C}$ ayant k pour état final. D^{DP} représente la métrique des chemins autorisés au sens de $\mathfrak{C}$ et finissant dans l'état k.

Exemple 2: on peut définir pour chaque classification k le sous-ensemble $\mathfrak{C}_k$ composé des chemins engendrant la classification k. D^{DP} représente la métrique des chemins autorisés au sens de $\mathfrak{C}$ et engendrant la classification k.

Dans ces deux exemples, les sous-ensembles $\mathfrak{C}_k$ forment une partition de $\mathfrak{C}$.

Déploiement temporel d'une séquence de vecteurs et alignement temporel de deux séquences de vecteurs

La programmation dynamique permet, nous l'avons vu, de calculer une métrique entre deux séquences $\mathbf{W}$ et $\mathbf{X}$.

La programmation dynamique induit également un alignement temporel entre les deux suites de vecteur $\mathbf{W}$ et $\mathbf{X}$, permettant d'obtenir des formes (alignées, donc distordues temporellement) ayant mêmes dimensions.

Ainsi si le calcul de l'alignement optimal de $\mathbf{X} = (\mathbf{x}_1,\ldots, \mathbf{x}_{Ix})$ et de $\mathbf{W} = (\mathbf{w}_1,\ldots, \mathbf{w}_{Iw})$ emploie le chemin $q=(I,\sigma,\hat{\sigma})$, les formes issues de $\mathbf{X}$ et de $\mathbf{W}$ et alignées temporellement par q sont: $\sigma(\mathbf{X})=(\mathbf{x}_{\sigma(1)},\ldots, \mathbf{x}_{\sigma(I)})$ et $\hat{\sigma}(\mathbf{W})=(\mathbf{w}_{\hat{\sigma}(1)},\ldots, \mathbf{w}_{\hat{\sigma}(I)})$.

Nous dirons que $\sigma(\mathbf{X})$ est le déploiement temporel de $\mathbf{X}$ selon σ.

Déploiement temporel selon un automate fini

Considérons une forme $\mathbf{W}$, constituée d'une séquence de vecteurs. Un automate fini tel que celui décrit en § ii, permet d'engendrer en au plus n instants un faisceau de formes à partir de $\mathbf{W}$. En employant le formalisme des treillis, ce faisceau s'écrit:

$$\Lambda_n(\mathbf{W},\mathfrak{C}) = \{ \hat{\sigma}(\mathbf{W}) : \exists\ p=(I,id,\hat{\sigma}) ;\ p \in \mathfrak{C},\ I \le n\} \qquad (5.5)$$

iv Utilisation de l'alignement temporel par les systèmes de plus proche voisinage

Nous venons de mettre en évidence que l'algorithme de programmation dynamique induit une famille de métriques particulièrement adaptées pour comparer et aligner temporellement des séquences de vecteurs. On peut alors se ramener au cas simple des formes de dimensions fixes (cf. § 4), ce qui permet d'employer des techniques similaires aux techniques d'apprentissage pour les plus proches voisins ordinaires (cf. § 4.2.3.1).

Cependant, il existe plusieurs façons d'accommoder la programmation dynamique pour obtenir des systèmes de plus proches voisinages aptes à prendre en compte des notions spécifiques aux données de nature temporelles.

On cherche à mettre en évidence un formalisme pour systèmes de plus proches voisinages employant la topologie induite par la programmation dynamique. Comme nous venons de le rappeler, dans les cas simples, on peut se ramener au cas de la reconnaissance de formes de dimensions fixes avec un nombre de classes réduit. Mais dans les cas complexes, tels que la reconnaissance du message parlé où les mots s'enchaînent naturellement, la classification est beaucoup plus ardue et doit tenir compte de la dynamique temporelle.

Approche simple: plusieurs références

Un système de plus proche voisinage est constitué de plusieurs références. Nous venons de voir que la programmation dynamique permet de calculer une métrique entre deux séquences de vecteurs. Considérons un système de plus proche voisinage, constitué par une stratégie de vote, par la métrique D^{DP} (5-2) et par un ensemble de références $\{\mathbf{W}_1, \mathbf{W}_2,\ldots \mathbf{W}_n\}$. La

programmation dynamique permet alors, pour chaque exemple $\mathbf{X}$, de déterminer quelles sont les références les plus proches au sens de la métrique D^{DP} qu'elle induit.

On reste alors dans un cadre très proche du cadre habituel des systèmes de plus proches voisinages. Cette approche permet par exemple de classifier les images acoustiques issues de mots isolés.

Approche évoluée: plus proche voisinage "virtuel"

Plaçons-nous à présent dans le cas où le treillis des chemins est engendré par un automate fini où toutes les connexions ont un délai de 1. Cela se traduit par une propriété simple que vérifient les chemins possibles: $\sigma(i) \equiv i$. Un chemin $p = (I,\sigma,\hat{\sigma}) = (Ix,Id,\hat{\sigma})$ ne se caractérise plus que par $\hat{\sigma}$.

Alors les paramètres des connexions prennent la forme $\lambda_{i-1,\hat{\sigma}(i-1),i,\hat{\sigma}(i)}$. Très souvent, ces paramètres sont indépendants de l'instant et prennent la forme $\lambda_{\hat{\sigma}(i-1),\hat{\sigma}(i)}$.

$$D^{DP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}) = \left(\underset{\hat{\sigma}\in\mathfrak{C}}{\text{Min}} \sum_{i=1}^{l} f\ (\ \lambda_{i-1,\hat{\sigma}(i-1),i,\hat{\sigma}(i)}\ ,\ d(\mathbf{x}_i,\mathbf{w}_{\hat{\sigma}(i)})\ ) \right) \tag{5.6}$$

Nous avons vu que le déploiement temporel peut transformer une séquence de vecteurs $\mathbf{W}$ en un faisceau de séquences-références virtuelles $\Lambda_n(\mathbf{W},\mathfrak{C})$. Cette idée suggère que des opérations d'alignement fort complexes effectuées par la programmation dynamique sont en fait des opérations de sélection du plus proche voisin, au sein d'un faisceau de références virtuelles. Mais l'alignement effectué par la programmation dynamique prend en compte les paramètres des connexions, ce que ne fait pas une métrique ordinaire opérant sur des formes ordinaires.

C'est pourquoi nous allons changer légèrement de topologie et incorporer les paramètres de connexions λ dans les vecteurs des références virtuelles. Posons:

$$\mathbf{x}'_i = \begin{pmatrix} \mathbf{x}_i \\ 0 \end{pmatrix} ;\ \mathbf{w}'_i(\lambda,\hat{\sigma}) = \begin{pmatrix} \mathbf{w}_{\hat{\sigma}(i)} \\ \lambda_{i-1,\hat{\sigma}(i-1),i,\hat{\sigma}(i)} \end{pmatrix} \tag{5-7a}$$

Les vecteurs $\mathbf{x}'_i$ et $\mathbf{w}'_i(\lambda,\hat{\sigma})$ sont de simples extensions des vecteurs $\mathbf{x}_i$ et $\mathbf{w}_{\hat{\sigma}(i)}$, incorporant les paramètres de connexion. On définit de même les séquences qui leur sont associées:

$$\mathbf{X}' = (\mathbf{x}'_1,\ldots,\mathbf{x}'_{lx})\ ;\ \mathbf{W}'(\lambda,\hat{\sigma},lx) = (\mathbf{w}'_1(\lambda,\hat{\sigma}),\ldots,\mathbf{w}'_{lx}(\lambda,\hat{\sigma})) \tag{5-7b}$$

Nous cherchons à reformuler avec ces notations la formule (5-56). Posons encore:

$$m(\mathbf{x}'_i,\mathbf{w}'_i(\lambda,\hat{\sigma})) = f\ (\ \lambda_{i-1,\hat{\sigma}(i-1),i,\hat{\sigma}(i)}\ ,\ d(\mathbf{x}_i,\mathbf{w}_{\hat{\sigma}(i)})\ ) \tag{5-7c}$$

$$M^{DP}(\mathbf{X}',\mathbf{W}'(\lambda,\hat{\sigma},lx)) = \sum_{i=1}^{l} m(\mathbf{x}'_i,\mathbf{w}'_i(\lambda,\hat{\sigma})) \qquad (5\text{-}7d)$$

On peut alors écrire:

$$D^{DP}(\mathbf{X},\mathbf{W},\lambda,\Sigma) = \min_{\hat{\sigma}\in\Sigma} M^{DP}(\mathbf{X}',\mathbf{W}'(\lambda,\hat{\sigma},lx)) \qquad (5.8)$$

On retrouve l'équation fondamentale d'un système de plus proche voisin simple: le 1NN. La programmation dynamique apparaît alors comme un moyen d'engendrer des références virtuelles $\mathbf{W}'(\lambda,\hat{\sigma},lx)$ à partir d'une référence-source **W**. Nous parlerons de plus proche voisinage "virtuel".

Cette écriture trouve toute son utilité quand on traite des séquences complexes telles que des spectrogrammes de parole prononcée naturellement. Pour reconnaître une phrase, il est matériellement impossible de comparer son spectrogramme avec les spectrogrammes de toutes les phrases possibles… En revanche, il est possible de reconnaître la suite de phonèmes (ou d'allophones, ou de triphones…) contenus dans le signal de parole, et, partant de là, on peut espérer reconnaître la phrase dans sa globalité. Pour reconnaître la suite de phonème, on peut bâtir un automate fini constituant un modèle de production de parole. La reconnaissance se fait alors par plus proche voisinage "virtuel".

Il faut ensuite définir une méthode pour étiqueter un chemin par une classe.

Prenons l'exemple de l'étiquetage d'un spectrogramme de parole par une suite de phonèmes. Il est possible de modéliser chaque phonème par un petit automate dont chaque état est associé à un vecteur, puis de relier tous ces petits automates, qui constituent alors un grand automate. Un chemin est une succession d'états $\hat{\sigma}(i)$ qui sont chacun attachés à un petit automate et donc à un phonème.

vi conclusion sur la métrique induite par programmation dynamique

L'algorithme de programmation dynamique est un des algorithmes les plus populaires et les plus efficaces en matière de reconnaissance de la parole et de correction lexicale. Il permet de prendre en compte avec une remarquable efficacité les problèmes de variabilité temporelle (hétérogénéité des rythmes d'émission, distorsion temporelle…).

Sa version classique ne prend pas efficacement en compte la variabilité intemporelle (dans le domaine fréquentiel, par exemple): la méthode souffre de l'insuffisance des distances euclidiennes ou même L^P pour traiter efficacement des problèmes complexes des vecteurs

multidimensionnels.

C'est pourquoi la programmation dynamique a été supplantée par les modèles de Markov cachés. Ces modèles, très proches de la programmation dynamique, permettent de prendre en compte la variabilité intemporelle avec un peu plus de réussite. De plus, ils fournissent un formalisme probabiliste contraignant (principe de causalité des vraisemblances conditionnelles) mais sain (plus grande facilité de travail analytique, algorithmes d'estimation-maximisation).

Plus récemment, des techniques hybrides combinant programmation dynamique et réseaux de neurones ont remis à l'honneur cet algorithme. De telles techniques sont également en cours de développement, qui combinent modèles de Markov cachés et réseaux de neurones.

5.2.1.2 Méthodes d'apprentissage pour programmation dynamique

i généralités

L'alignement temporel par programmation dynamique permet de calculer une métrique. La classification par programmation dynamique consiste traditionnellement à employer une multitude de références, et à appliquer une technique de plus proche voisinage. Ceci est très coûteux et il faut résumer les nombreuses références empiriques en quelques références évenruellement synthétiques. Quelques techniques ont été proposées pour cela. Mais il nous a semblé qu'elles étaient assez grossières et peu formalisées.

Nous avons donc développé une approche originale du problème. Nous optimisons des références synthétiques à l'aide de procédures de descente de gradient stochastique opérant sur des fonctions de coût inspirées des méthodes d'apprentissage pour plus proches voisins: kMeans, LVQ…

Nous avons défini ces techniques dans [1], [2] et nous en avons fourni les premiers résultats

---

1 **Driancourt X., Bottou L., Gallinari P.:** *Multi-Layer Percpetron, Learning Vector Quantization and Dynamic Programming: Comparison and Cooperation* - Proceedings of IJCNN (1991)

2 **Driancourt X., Bottou L., Gallinari P.:** *Comparison and Cooperation of Several Classifiers* - Proceedings of ICANN (1991)

dans [1], [2] (cf. § 6.3).

Il arrive qu'on considère, pour un exemple donné $\mathbf{X}_n$, l'ensemble $\mathfrak{C}^+$ des chemins de classe compatible avec celle de l'exemple. Comme nous l'avons déjà évoqué au § 5.1.2, la notion de classe peut donner lieu à plusieurs interprétations. L'interprétation rigoriste impose un étiquetage de chaque vecteur de $\mathbf{X}_n$. L'interprétation laxiste se contente d'imposer le respect de la succession des étiquettes.

ii DP-KMeans

La méthode d'apprentissage DP-KMeans supervisé consiste à minimiser la fonction de coût local:

$$J^{\text{DP-kMeans-sup}}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}) = \min_{\hat{\sigma}\in\mathfrak{C}} + M^{DP}(\mathbf{X}',\mathbf{W}'(\lambda,\hat{\sigma},lx)) \qquad (5\text{-}10a)$$

$$J^{\text{DP-kMeans-sup}}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}) = D^{DP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}^+) \qquad (5\text{-}10b)$$

où $\mathfrak{C}^+$ est l'ensemble des chemins de $\mathfrak{C}$ de classe compatible avec celle de $\mathbf{X}$. Rappelons que M est la métrique de la programmation dynamique (5-8) ou (5-9).

Concrètement, $J^{\text{DP-kMeans-sup}}$ représente le coût de l'appariement de $\mathbf{X}$ avec la plus proche référence virtuelle de bonne classe.

La méthode DP-KMeans n'utilise pas réellement les coûts de transitions λ. Quand la fonction f introduite dans la formule 5-2 effectue simplement l'addition de ses arguments, la méthode DP-KMeans se contente de faire converger λ vers $\mathbf{0}$ afin de minimiser le coût de l'appariement.

iii DP-LVQ2bis

La méthode d'apprentissage DP-LVQ2bis supervisé consiste à minimiser la fonction de coût local:

$$J^{\text{DP-LVQ2bis}}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C})$$

$$= \text{Min}\left(1 , \min_{\hat{\sigma}\in\mathfrak{C}} + M^{DP}(\mathbf{X}',\mathbf{W}'(\lambda,\hat{\sigma},lx)) - \min_{\hat{\sigma}\in\mathfrak{C}} M^{DP}(\mathbf{X}',\mathbf{W}'(\lambda,\hat{\sigma},lx)) \right)$$

---

1 **Driancourt X.:** *Extraction de Primitives Caractéristiques du Signal par Réseaux Neuronaux pour la Reconnaisance de la Parole* - Rapport DRET CJC-89/1591 (1991)

2 **Driancourt X., Gallinari P.:** *A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer perceptron* - Proceedings of ICASSP (1992)

$$= \text{Min}\left(1,\ D^{DP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}^{+}) \ -\ D^{DP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}) \right) \qquad (5\text{-}11)$$

où $\mathfrak{C}^{+}$ est l'ensemble des chemins de $\mathfrak{C}$ de bonne classe.

Concrètement, $J^{DP\text{-}LVQ2bis}$ représente la différence plafonnée des coûts de l'appariement de $\mathbf{X}$ avec la plus proche référence virtuelle de bonne classe et avec la plus proche référence virtuelle (de classe quelconque).

Contrairement à DP-KMeans, la méthode DP-LVQ2bis utilise réellement les coûts des transitions λ.

iv DP-LVQ2+

La méthode d'apprentissage DP-LVQ2+ supervisée consiste à minimiser stochastiquement la fonction de coût local:

$$J^{DP\text{-}LVQ2+}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}) = \min\left(1, \frac{1}{\delta}\left(\frac{\underset{\hat{\sigma}\in\mathfrak{C}}{\text{Min}}\ M^{DP}(\mathbf{X}',\mathbf{W}'(\lambda,\hat{\sigma},lx))}{\underset{\hat{\sigma}\in\mathfrak{C}}{\text{Min}}\ M^{DP}(\mathbf{X}',\mathbf{W}'(\lambda,\hat{\sigma},lx))} \ -\ 1 \right) \right)$$

$$J^{DP\text{-}LVQ2+}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}) = \min\left(1, \frac{1}{\delta}\left(\frac{D^{DP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C})}{D^{DP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}^{+})} \ -\ 1 \right) \right) \qquad (5\text{-}12)$$

Cette fonction de coût approche le risque empirique qu'on cherche à minimiser.

Elle a deux défauts. Tout d'abord, elle est irrégulière et est destinée à effectuer une optimisation après une bonne initialisation (avec DP-KMeans par exemple) sous peine de faire tomber le processus d'optimisation dans des minima locaux de mauvaise qualité. D'autre part, elle manque de nuance dans le jugement qu'elle porte sur la classification. Avec elle, une séquence est bien ou mal reconnue. Elle n'est pas capable de prendre en compte qu'une séquence peut être partiellement reconnue (taux de substitution / insertion / suppression).

5.2.1.3 Conclusion sur la programmation dynamique

L'algorithme de programmation dynamique est un des algorithmes les plus populaires et les plus efficaces en matière de reconnaissance de la parole et de correction lexicale. Il est utilisé sous sa forme spécialisée (5-8) mais aussi sous sa forme générale (5-1) pour les calculs nécessaires aux modèles de Markov cachés.

L'algorithme de programmation dynamique permet de prendre en compte la dynamique des phénomènes temporels. Dans le domaine de la reconnaissance de la parole, il permet de prendre

en compte la variété et l'irrégularité des rythmes d'élocution. En revanche, il éprouve quelques difficultés pour traiter convenablement la variété des spectres d'élocution ou le bruit. Les modèles cachés de Markov ont permis d'atténuer un peu ces faiblesses.

Nous avons proposé une famille générique de méthodes d'optimisation des paramètres de la programmation dynamique (références, paramètres des connexions) inspirée des plus proches voisinages: DP-KMeans, DP-LVQ2, DP-LVQ2+…

5.2.2 Stratégies de vote: l'exemple du shift-LVQ

Dans le cas de formes temporelles simples, telles des mots courts, il est possible d'utiliser des méthodes rudimentaires prenant en compte l'aspect temporel.

L'une d'entre elle est le shift-KNN. Elle consiste à découper les formes en tronçons (éventuellement partiellement superposés). Chaque référence donne lieu à plusieurs tronçons, associés chacun à la classe de la référence dont ils sont extraits. On suppose que les formes et les références sont de longueurs identiques (éventuellement, on les complète par du silence).

Lors de la phase de reconnaissance, on découpe la forme en tronçons. Chaque tronçon est comparé avec les tronçons de même position des références. Puis on fait voter les k plus proches tronçons de chaque position, pour obtenir une vote global qui fait office de décision.

Chaque position peut donner lieu à une optimisation des tronçons qui lui sont associés. Une de ces méthodes d'apprentissage est le shift-LVQ [1].

5.3 Modèles de Markov cachés

5.3.1 Chaînes de Markov, modèles de Markov cachés

Considérons un automate fini, constitué d'états $S = (s_1,\ldots s_{Ns})$ et de relations liant chacune deux états. On définit une chaîne de Markov en associant à chaque état s_i une probabilité de transition $a_{i,j}$ et une probabilité d'initialisation Π_i. Si, de plus, on associe à chaque état s_i une loi $b_i(\mathbf{x})$ d'émmission de vecteurs (ou observations) $\mathbf{x}$, on obtient un modèle de Markov caché.

On appelle λ la donnée d'un modèle de Markov caché, c'est-à-dire d'un réseau d'automate et de

1 **Mc Dermott E., Katagari S.:** *Shift-invariant multi-category phonem recognition unsing Kohonen's LVQ2* - Proceedings of ICASSP, pp 81-84, Galsgow U.K. (1989)

ses coefficients $\mathbf{A}$, $\mathbf{b}$, Π .

Il existe de nombreuses variantes de modèles de Markov cachés, dont nous ne mentionnons ici que les deux plus connues. Les modèles de Markovs cachés discrets font appel à une loi d'émission discrète (nombre fini de vecteurs engendrables pour chaque état). Les modèles de Markovs cachés continus font appel à une loi d'émission continue (généralement une gaussienne par état).

Les modèles de Makovs sont naturellement des modèles producteurs. Considérons un marqueur, circulant dans l'automate d'état en état en empruntant les relations. Si on déplace le marqueur en fonction des probabilités de transitions, le marqueur va suivre un chemin d'états. Si, de plus, on emploie à l'issue de chaque transition la loi d'émission de vecteurs asociée à l'état courant du marqueur, on produit une série d'observations $\mathbf{X} = (\mathbf{x}_1 \ldots \mathbf{x}_T)$. Ainsi un modèle de Markov caché est associé à un processus stochastique et engendre une distribution de séries d'observations. Si le modèle est bien paramètré, chaque série d'observation ainsi "produite" est de même nature que les observations extérieures (e.g. spectrogrammes d'élocution de parole ou d'écho sonar…).

On désigne par t un instant, donnant lieu à une observation $\mathbf{x}_t$ extraite de $\mathbf{X}$. On désigne par σ un chemin d'états dans l'automate. Un tel chemin associe tout instant $t \in [1;T]$ l'état $s_{\sigma(t)}$.

La présentation classique des modèles de Markovs cachés repose sur trois questions.

- Quelle est la probabilité d'émmission d'une série d'observations $\mathbf{X}$ par un modèle de Markov caché ? (problématique de la mesure de l'adéquation d'un modèle à une série d'observations).

- Quel est le chemin $\mathbf{S}$ engendrant une série d'observations $\mathbf{X}$ avec la plus forte probabilité ? (problématique de l'étiquetage automatique).

- Etant donné un automate, quelle est le jeu de probabilité de transitions, de probabilités d'initialisation et de probabilité d'émmission définissant un modèle de Markov caché qui maximise, la probabilité de production d'une série d'observations $\mathbf{X}$? (problématique de l'apprentissage par optimisation du maximum de vraissemblance).

Répondre à la première question permet de mesurer l'adéquation d'un modèle de Markov caché à une observation $\mathbf{X}$.

Répondre à la seconde question permet d'utiliser un modèle de Markov caché pour des tâches de classification de séquences (reconaissance de la parole…).

Répondre à la troisième question permet d'optimiser un modèle de Markov pour que son comportement soit ausi proche que possible de la réalité observée, représenté par $\mathbf{X}$. Il faut souligner que cette question est un peu biaisée, puisqu'il existe en réalité de nombreux critères d'optimisation différents du maximum de vraissemblance qu'on se propose d'utiliser ici (cf. §3.3).

5.3.2 Mesure de l'adéquation

Il s'agit ici de calculer $P(\mathbf{X} \mid \lambda)$.

$$P(\mathbf{X} \mid \sigma, \lambda) = \prod_{t=1}^{T} P(\mathbf{x}_t \mid s_{\sigma(t)}, \lambda) = \prod_{t=1}^{T} b_{\sigma(t)}(\mathbf{x}_t)$$

$$P(\mathbf{X} \mid \lambda) = \sum_{\sigma} P(\mathbf{X} \mid \sigma, \lambda) P(\sigma \mid \lambda)$$

$$= \sum_{\sigma} \Pi_{\sigma(1)} b_{\sigma(1)} \prod_{t=2}^{T} b_{\sigma(t)}(\mathbf{x}_t) \, a_{\sigma(t-1),\sigma(t)}$$

Cette expression peu paraître satisfaisante. Cependant elle fait intervenir l'ensemble des chemins de longueur T que l'automate peut engendrer: sa complexité est $2 * T * Ns^T$… Il n'est donc pas possible de l'utiliser en pratique sinon dans des cas très simples.

Fort heureusement, on peut employer des techniques de programmation dynamique pour calculer $P(\mathbf{X} \mid \lambda)$. Deux procédures, nommées "forward" et "backward" sont traditionnellement employées.

Pour la procédure "forward", on définit:

$$\alpha_t(i) = P(\mathbf{x}_1, \ldots, \mathbf{x}_t, \sigma(t) = i \mid \lambda)$$

On a alors la relation de récurrence:

$$\alpha_1(i) = \Pi_i \, b_i(\mathbf{x}_1) \;\; ; \;\; \alpha_{t+1}(j) = \left[\sum_{i=1}^{Ns} \alpha_t(i) \, a_{i,j} \right] b_j(\mathbf{x}_{t+1})$$

Avec la propriété terminale:

$$P(\mathbf{X} \mid \lambda) = \left[\sum_{i=1}^{Ns} \alpha_t(i) \right]$$

La procédure "backward" est similaire, elle repose sur une récurrence avec:

$$\beta_t(i) = P(\mathbf{x}_{t+1},\ldots, \mathbf{x}_T, \sigma(t) = i \mid \lambda)$$

Chacune des deux procédures "forward" et "backward" permettent de calculer $P(\mathbf{X} \mid \lambda)$ avec une complexité calculatoire de $N^2 * T$.

5.3.3 Etiquetage

Etant donné un modèle et une séquence d'observations, on désire étiqueter chaque observation par un état.

On distingue classiquement deux critères permettant à un modèle de Markov caché d'effectuer un tel étiquetage.

- Le premier consiste à maximiser $P(\sigma=i, \mid \mathbf{X}, \lambda)$. Ceci revient à déterminer, pour chaque instant t, l'étiquette la plus probable $\sigma(t)$, sans tenir compte des étiquettes précédentes et suivantes.
- Le second consiste à maximiser $P(\sigma, \mid \mathbf{X}, \lambda)$. Ceci revient à déterminer le chemin (suite d'étiquettes) le plus probable.

i Calcul indépendant de l'étiquette la plus probable

On définit et on cherche à calculer:

$$\gamma_t(i) = P(\sigma(t)=i \mid \mathbf{X}, \lambda) = \frac{P(\sigma(t)=i, \mathbf{X} \mid \lambda)}{P(\mathbf{X} \mid \lambda)}$$

$$= \frac{P(\sigma(t)=i, \mathbf{X} \mid \lambda)}{\sum_{j=1}^{Ns} P(\sigma(t)=j, \mathbf{X} \mid \lambda)}$$

En remarquant que :

$$P(\sigma(t)=i ,\mathbf{X} \mid \lambda) = P(\sigma(t)=i ,\mathbf{x}_1,\ldots,\mathbf{x}_t \mid \lambda) * P(\sigma(t)=i ,\mathbf{x}_{t+1},\ldots,\mathbf{x}_T \mid \lambda)$$

$$= \alpha_t(i) * \beta_t(i)$$

on se ramène aux procédures forward et backward du § 5.3.2 :

$$\gamma_t(i) = P(\sigma(t)=i \mid \mathbf{X}, \lambda) = \frac{\alpha_t(i) * \beta_t(i)}{\sum_{j=1}^{Ns} \alpha_t(j) * \beta_t(j)}$$

L'étiquette la plus probable est, bien sur, l'indice i maximisant $\gamma(i)$.

ii Calcul du chemin le plus probable

On définit:

$$\delta_t(i) = \max_{\sigma(1),\ldots,\sigma(t-1)} P(\sigma(1),\ldots,\sigma(t-1), \sigma(t-1)=i, \mathbf{x}_1,\ldots,\mathbf{x}_t \mid \lambda)$$

On utilise pour cela la relation de récurrence:

$$\delta_1(i) = \Pi_i\, b_i(\mathbf{x}_1) \;;\; \delta_{t+1}(i) = \left[\max_j \delta_t(j)\, a_{i,j} \right] b_i(\mathbf{x}_{t+1})$$

Il faut de plus conserver l'identité du chemin. Pour cela on définit:

$$\psi_1(i) = 0 \;;\; \psi_t(i) = \operatorname{argmax}_j \left(\delta_{t-1}(j)\, a_{j,i} \right)$$

Ceci permet de calculer les matrices $\Delta=\delta_t(i)$ puis $\Psi=\psi_t(i)$. Le chemin optimal σ^* peut être retrouvé rétrospectivement par la procédure:

$$\sigma^*(T) = \operatorname{argmax}_j \left(\delta_T(j) \right) \;;\; \sigma^*(t) = \psi_{t+1}\left(\sigma^*_{t+1} \right)$$

5.3.4 Optimisation du modèle

Etant donné un automate, on désire optimiser les coeficients $\mathbf{A}$, $\mathbf{b}$, Π . On peut utiliser pour cela de nombreux critères tels que ceux présentés au §3.3.

On emploie classiquement l'algorithme d'estimation-maximisation de Baum (qui est un cas limite de la descente de gradient sous contrainte) appliqué à la fonction d'erreur MLE (cf. § 3.3.1.2). En faisant varier $\mathbf{A}$, $\mathbf{b}$ et Π, on cherche à optimiser la quantité:

$$C^{MLE}(\lambda) = p(O \mid \lambda)$$

On définit la quantité:

$$\xi_t(i,j) = \frac{P(\sigma(t) = i, \sigma(t+1) = j \mid O, \lambda)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_t(i)\, a_{ij}\, bj(O_{t+1})\, \beta_{t+1}(i)}{P(\, O \mid \lambda \,)}$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(i)}{\sum_{i=1}^{Ns} \sum_{j=1}^{Ns} \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}$$

$$\gamma_t(j) = \sum_{i=1}^{Ns} \xi_t(i,j)$$

Ces quantité, calculable à l'aide des procédures "forward" et "backward" permet d'exprimer le réajustement des paramètres Π et A selon l'algorithme de Baum:

$$\Pi_i^{n+1} = \gamma_t(i)$$

$$a_{i,j}^{n+1} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\gamma_t(i)}$$

Dans le cas où la loi d'émission b est discrète, on peut la représenter à l'aide d'une matrice $B = (b_{i,k})_{i \le Ns, k \le No}$. Alors le réajustement de B selon l'algorithme de Baum s'écrit:

$$b_i^{n+1}(k) = \frac{\sum_{t=1, \mathbf{O}_t = \mathbf{x}_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(i)}$$

5.3.5 Conclusion

Les modèles de Markov cachés permettent d'aprocher des phénomènes de nature temporels fort complexes. Ils proposent un cadre probabiliste clair. Ils bénéficient de procédures de calcul efficaces, découlant de la programmation dynamique. Ils disposent de nombreuses procédures d'apprentissage et d'optimisation, en particulier l'algorithme d'estimation-maximisation de Baum-Welch. Ces qualités font des modèles de Markov cachés un outil très efficace pour le traitement de données temporelles (sonar, parole…).

5.4 Associateurs paramétriques

5.4.1 Utilisation du partage de paramètres

Nous avons déjà évoqué § 4.3.1.3, § 5.1.1 le partage de paramètres, qui permet de réduire la capacité d'apprentissage du système. Quand ce partage permet de tenir compte d'informations *a priori*, il permet au système d'apprentissage d'obtenir de bien meilleurs scores sur des bases de données de test (distributions empiriques autres que celles employées lors de l'apprentissage). Le partage de paramètres est particulièrement employé pour traduire des invariances par transformations géométriques simples dans la topologie discrète des entrées: translations, rotations de 90°…

Dans le cas du traitement de formes temporelles, le partage de paramètres permet d'implanter efficacement l'invariance dans le temps de certains traitements.

Considérons le cas de la reconnaissance de la parole. Il est intuitif de chercher à transformer un spectrogramme en une information de plus haut niveau grâce à des filtres invariants dans le temps. Pour implanter ces filtres, il est généralement facile d'imposer l'égalité de paramètres entre de multiples unités fonctionnelles accomplissant chacune le traitement d'un petit contexte temporel.

5.4.2 Application au MLP: le TDNN

Le TDNN fut introduit par Hanazava, Hinton, Lang, Shikano et Waibel [1], pour la reconnaissance de la parole afin d'implanter, au sein des réseaux multicouches type MLP, l'invariance dans le temps de traitements bas-niveau. La modélisation initiale (automate à pile et à délais) explique le nom donné à ces réseaux.

Une modélisation équivalente (et plus générale) fait appel au partage de paramètres au sein de chaque couche neuronale à structure temporelle. Les couches à structure temporelle accomplissent un filtrage, qui prend la forme de "convolutions non-linéaires".

---

1 **Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K.:** Phoneme recognition neural networks vs hidden Markov models - Proceedings of ICASSP vol 1, pp 107-110 (1988)

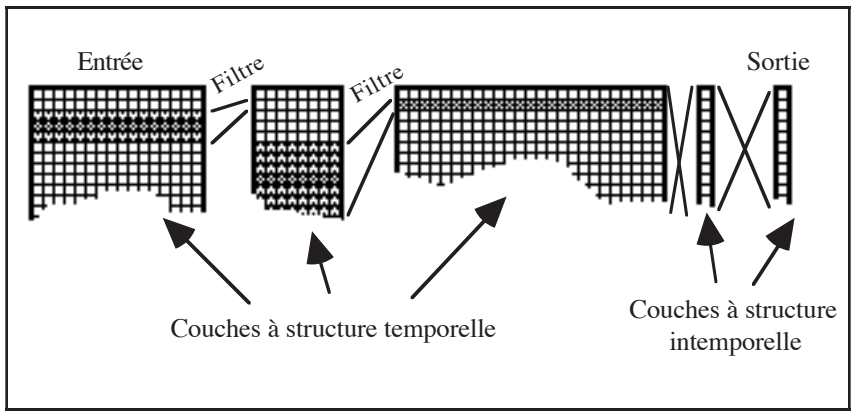


Figure 5.3
TDNN

Le TDNN est un MLP à poids partagés.
Ses couches inférieures sont fonctionnellement identiques à des convolutions non-linéaires.

L'utilisation d'un TDNN pour une tâche de reconnaissance des formes se fait en ajoutant des couches ordinaires aux couches à structure temporelle. Ces couches ordinaires produisent le vecteur de sortie utilisé pour la classification.

Le TDNN est devenu une technique très populaire dans le domaine de la reconnaissance de la parole. Il s'est montré particulièrement efficace pour la reconnaissance de petites formes telles que des unités phonémiques [1], [2] ou pour la reconnaissance de mots isolés [3].

Néanmoins, le TDNN n'est pas apte à traiter de longues séquences de formes enchaînées ni à traiter le problème de grands vocabulaires. Il ne permet pas de prendre en compte l'aspect dynamique de la parole, c'est à dire la variété des rythmes d'élocution, contrairement à la programmation dynamique et aux modèles markoviens.

5.4.3 Réseaux récurrents

Considérons un MLP. Ajoutons à ce réseau des connexions qui, contrairement aux autres, sont associées à un délai et peuvent relier une couche amont à une couche aval.

---

1 **Waibel A., Hanazawa T., Hinton G., Lang K. :** *Phoneme recognition: neural networks vs. Hidden Markov Models* - Proceedings of ICASSP 88, S-Vol 1, pp 107-110, (1988)

2 **Devilers L. :** *Reconnaissance monolocuteur des phonèmes français au moyen de réseaux à masques temporels* - Actes des XVIIIèmes Journées d'Etudes sur la Parole - Montréal, (1990)

3 **Bottou L. :** *Reconnaissance de la parole par réseaux multi-couches* - Rapport de stage, DEA d'Informatique du LRI, Université de Paris XI, Orsay (1988)

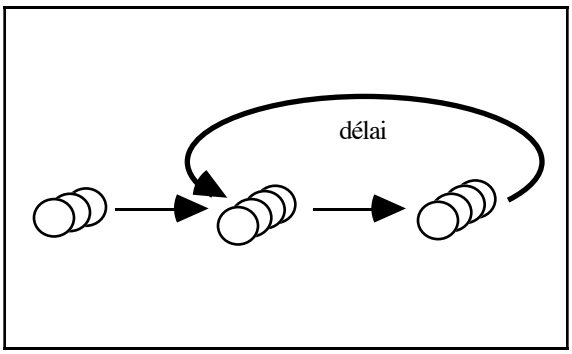


Figure 5.4
Réseau récurrent

*Les réseaux récurrents sont des extensions des réseaux multicouches. Ils contiennent des connexions "rétrograde*s" *introduisant des boucles dans le réseau. Ces connexions sont associées à un délai.*

On leur donne le nom de réseaux récurrents ou itératifs [1]. Il est possible de représenter un réseau récurrent en le déployant dans le temps.

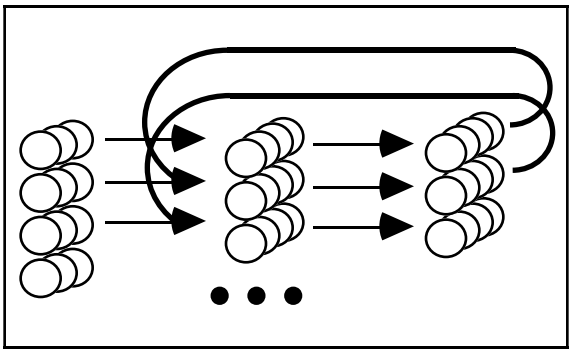


Figure 5.5
Déploiement temporel d'un réseau récurrent

On peut déployer temporellement un réseau récurrent. On trouve alors une structure proche du TDNN, auquel on ajoute des connexions rétrogrades.

Le partage de paramètres permet alors d'implanter un tel réseau pour les besoins de la propagation.

Cependant, le partage de paramètres ne permet pas d'implanter efficacement un tel réseau pour les algorithmes de descente de gradient. En effet, la récurrence implique qu'une partie du flux de l'information reçoit l'influence de diverses instances de connexions récurrentes qui partagent les mêmes paramètres… Les paramètres des connexions récurrentes ont donc une influence d'ordre supérieur. Se contenter d'appliquer l'algorithme de rétropropagation du gradient engendre donc un biais qui fausse l'apprentissage (cf. 6.7).

5.5 Systèmes modulaires

Nous avons déjà décrit plusieurs systèmes utilisant l'information temporelle *a posteriori*. Nous pouvons les regrouper en deux catégories: les systèmes traitant efficacement la variabilité

1 **Watrous R. L., Lokendra S.:** *Learning phonetic features using connectionnist networks : an experiment in speech recognition* - pp 381-388, IEEE vol 4, san Diego California (1987)

temporelle et les systèmes traitant efficacement la variabilité intemporelle.

De nombreux problèmes conjuguent ces deux variabilités, d'où de nombreuses recherches visant à concevoir des systèmes doublement robustes. Une voie semble prometteuse: l'utilisation de systèmes modulaires.

5.5.1 Un précurseur

A la fin des années 80 H. Bourlard et C. Wellekens proposent de concevoir des systèmes modulaires HMM-MLP [1], notamment pour la reconnaissance de la parole. La coopération des deux modules, mise au point depuis lors [2] est très contrainte: au MLP la tâche de calculer la probabilité qu'un vecteur représente un phonème (en tirant partie de la propriété (3-45) du critère MSE), au HMM la tâche de fournir une information de plus haut niveau. Ce système a rapidement attiré l'attention.

Une question se pose cependant: peut-on donner en entrée au MLP un contexte temporel de plusieurs vecteurs, ce qui enrichi le modèle, mais risque d'engendrer une certaine incohérence entre les causalités des probabilités calculées par le MLP (probabilité sachant un contexte) et de celles attendues par le HMM (qui utilisent les probabilités sachant un vecteur) [3]…

La réponse est mitigée: on pert la cohérence nécessaire à la formulation probabiliste. Néammoins, on peut concevoir un HMM non comme un modèle probabiliste, mais comme un automate optimisable. Ces systèmes modulaires retrouvent une cohérence si on emploie des procédures d'optimisation, robustes à la disparition de l'interprétation probabiliste, telles que les descente de gradient stochastique.

5.5.2 Quelques références

Depuis, plusieurs autres équipes ont proposé des systèmes modulaires combinant des MLP et des HMM ou des DP : M. Fanzini et A. Waibel à Carnegie Melon University puis à Barcelone

---

1 **Bourlard H., Wellekens C. :** *Links between Markov Models and Multilayer Perceptrons* - Advance in Neural Information Processing Systems, Vol 1, pp. 502-510, Morgan Kaufman, Denver (1989)

2 **Bourlard H., Morgan N., Wooters Ch. :** *Connectionist aproaches to the Use of Markov Models for Speech Recognition* - Advances in Neural Information Processing System 3, Morgan Kaufman (1991)

3 **Bourlard H., Morgan N., Woters Ch., Renals S. :** *CDNN: A context-dependent neural network for continuous speech recognition* - ICASSP, vol II-349, San Francisco (1992)

[1], K. Iso et T. Watanabe [2], P. Haffner et A. Waibel à Carnelie Melon University puis au CNET [3], Y. Bengio à McGill University puis aux laboratoires AT&T Bell [4], L. Devillers au LIMSI du CNRS [5], X. Driancourt et P. Gallinari à l'Université d'Orsay [6], [7], [8], A. Mellouk et P. Gallinari à l'Université d'Orsay [9], [10].

Outre leur aspect modulaire, ces systèmes ont en commun de représenter, pour la plupart d'entre eux, l'état de l'art en matière de reconnaissance de la parole.

Nous ne pouvons donner de description détaillée de tous ces systèmes. Nous détaillerons les principales équations des deux systèmes développés dans notre équipe, en mentionnant les points communs avec les autres systèmes modulaires que nous venons de mentionner.

5.5.3 Un système modulaire MLP-DP-1NN

Nous avons proposé une méthode originale de coopération entre MLP et DP. La programmation dynamique permet de calculer une métrique. Il est possible d'enrichir le système DP en codant les entrées à l'aide d'un filtre de type MLP.

---

1 **Franzini M., Lee K. F., Waibel A.:** *Connectionnist Viterbi Training: A new hybrid method for continuous speech recognition* - ICASSP pp 245 (1990)

2 **Iso K., Watanabe T.:** *Large vocabulary speech recognition using neural prediction model* - pp 61-64 ICASSP (1991)

3 **Haffner P., Fanzini M., Waibel A.: Integrating** *Time alignment and neural networks for high performance continuous speech recognition* - Proceeding of ICASSP, S2.17, pp 105-108 (1991)

4 **Bengio Y., De Mori R., Flammia G. and Kompe R.:** *Gloabal Optimization of a Neural Network-Hidden Markov Model Hybrid* - Proceedings of IJCNN (1991)

5 **Devillers L., Dugast C.:** *Comparison of Continuous Mixture Densities and TDNN in a Viterbi-framework: Experiment on Speaker Dependant Darpa RM1* - Eurospeech, Genova, Italy, 24-26 Sept. 1991.

6 **Driancourt X., Bottou L.:** *TDNN Extracted Features* - Actes de Neuronimes, Nimes (1990)

7 **Driancourt X., Bottou L., Gallinari P.:** *Comparison and Cooperation of Several Classifiers* - Proceedings of ICANN (1991)

8 **Driancourt X., Gallinari P.:** *A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer perceptron* - Proceedings of ICASSP (1992)

9 **Mellouk A., Gallinari P.:** *A Discriminative Neural Prediction System for Speech Recognition* - ICASSP (1993)

10 **Mellouk A., Gallinari P.:** *Discriminative Training for Improved Neural Prediction Systems* - ICASSP (1994)

$$D^{DP\text{-}MLP\text{-}1N}(\mathbf{X},\mathbf{W}^{MLP},\mathbf{W}^{DP},\lambda,\mathfrak{C}) = D^{DP}(f^{MLP}(\mathbf{X},\mathbf{W}^{MLP}),\mathbf{W}^{DP},\lambda,\mathfrak{C}) \quad (5\text{-}13)$$

où D^{DP} est donné par l'équation (5-8). Ceci peut être appliqué à la reconnaissance de la parole en mots isolés aussi bien qu'à la reconnaissance de la parole continue.

Il reste à optimiser un tel système. Tant que l'optimisation porte sur les paramètres d'un seul des deux modules, on peut employer des fonctions de coûts de la forme MLP-DP-KMeans et MLP-DP-LVQ2bis.

En revanche, quand l'optimisation porte sur les paramètres des deux modules, de même que pour les combinaisons MLP-1NN évoquées au § 4.4.2.3, il faut prendre garde aux problèmes de codages ponctuels engendrés par l'emploi de fonctions de coûts proportionnelles à l'échelle des sorties du MLP et des références du module de plus proche voisin.

On est donc amené à parachever l'apprentissage avec des fonctions de coûts de la forme MLP-DP-LVQ2+ qui présentent en outre l'avantage d'approcher le risque moyen:

$$\begin{aligned} J^{MLP\text{-}DP\text{-}LVQ2+}(\mathbf{X},\mathbf{W}^{MLP},\mathbf{W}^{DP},\lambda,\mathfrak{C}) &= J^{DP\text{-}LVQ2+}(f^{MLP}(\mathbf{X},\mathbf{W}^{MLP}),\mathbf{W}^{DP},\lambda,\mathfrak{C}) \\ &= \min\left(1, \frac{1}{\delta}\left(\frac{D^{DP}(f^{MLP}(\mathbf{X},\mathbf{W}^{MLP}),\mathbf{W}^{DP},\lambda,\mathfrak{C}^{+})}{D^{DP}(f^{MLP}(\mathbf{X},\mathbf{W}^{MLP}),\mathbf{W}^{DP},\lambda,\mathfrak{C})} - 1\right)\right) \end{aligned} \quad (5\text{-}14)$$

où $\mathfrak{C}^{+}$ est l'ensemble des chemins de $\mathfrak{C}$ de bonne classe.

Le chapitre 6 est consacré aux expériences que nous avons menées, notamment avec les systèmes modulaires MLP-plus proches voisins. Ces systèmes ont donné les meilleurs résultats de toutes les expériences que nous avons pu mener.

5.5.4 Un système modulaire MLP-prédictif-DP-1NN

Il existe plusieurs façons de faire coopérer perceptrons multicouches et programmation dynamique.

Nous avons évoqué un système où le Perceptron estime les probabilités a posteriori. Nous avons exposé notre système où le Perceptron filtre les formes. Nous allons à présent décrire succintement un système modulaire MLP-DP prédictif.

Dans un tel système, plusieurs MLP interviennent. Chaque MLP est associé à un état d'un automate à délai. L'entrée du MLP est composée d'un petit contexte temporel extrait de la forme à reconnaître. La sortie désirée du MLP est composée d'un ou plusieurs vecteurs contigüs au contexte temporel fourni en entrée. On mesure l'erreur de prédiction comise par le MLP. Cette

valeur détermine un coût propre à l'appariement du MLP et du contexte temporel fourni en entrée. On obtient un système d'une forme proche de (5-3):

$$D^{PDP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}) = \left(\underset{\mathfrak{C}}{\text{Min}} \sum_{i=1}^{I} f\ (\ \lambda_{\hat{\sigma}(i-1),\hat{\sigma}(i)}\ ,\ d(c(\mathbf{X},i),\mathbf{w}_{\hat{\sigma}(i)})\ ) \right) \qquad (5\text{-}15)$$

où $c(\mathbf{X},i)$ définit le $i^{ème}$ contexte temporel de $\mathbf{X}$,

et d n'est plus un métrique ordinaire mais la mesure de la qualité de l'appariement de $\mathbf{x}$ avec le MLP associé à $\hat{\sigma}(i)$ et ayant pour paramètres $\mathbf{w}_{\hat{\sigma}(i)}$:

$$d(\mathbf{x}_i,\mathbf{w}_{\hat{\sigma}(i)}) = m(\ f^{MLP}\ (\mathbf{X},\mathbf{w}_{\hat{\sigma}(i)}) - \mathbf{x}_i\ ) \qquad (5\text{-}16)$$

où m est une métrique (souvent l'erreur quadratique).

L'appelation "prédictives" est trompeuse. En effet, ces systèmes:

- n'utilisent pas toujours des contextes passés (par rapport au vecteur présent qu'on veut prévoir); il gagnent à utiliser un contexte passé-futur;
- ont souvent pour objet de classifier et non d'apprendre à prédire. Dès lors, il est souhaitable d'employer des systèmes discriminants. Ceci conduit à l'expression surprenante de "prédiction discriminante".

Plusieurs méthodes ont été proposées pour effectuer de la prédiction discriminante.

On retrouve dans la litérature la méthode discriminante qui consiste à entrainer le système prédictif à fournir une erreur:

- nulle pour le modèle prédicteur désiré (i.e. le modèle sélectionné pour l'instant i par la recherche du meilleur chemin de bonne classe)
- forte pour les autres modèles

Nous avons proposé une méthode originale conforme à notre démarche habituelle qui est fondée sur des fonctions d'erreur telles que LVQ2bis ou LVQ2+ qui se rapprochent plus du risque moyen tout en restant différentiables.

$$J^{PDP\text{-}LVQ2bis}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C})$$

$$= \min(\ \delta,\ D^{PDP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C}^{+}) - D^{PDP}(\mathbf{X},\mathbf{W},\lambda,\mathfrak{C})\ ) \qquad (5\text{-}17)$$

Cette approche a été implantée, adaptée et testée avec succès par A. Mellouk au sein de son

système de décodage accoustico-phonétique de la parole continue [1]. Sur la base de données TIMIT les taux d'erreurs (substitution, suppression, insertion) ont été réduits d'un tiers par rapport à une prédiction discriminante simple évoquée plus haut et réduits de plus de la moitié par rapport aux approches non-discriminantes.

5.6 Conclusion

Nous avons exposé en détail la programmation dynamique et son application à l'appariement de séquences de vecteurs. Nous avons proposé une famille originale d'algorithmes d'optimisation de références de systèmes du plus proche voisin pour la métrique de la programmation dynamique: DP-KMeans, DP-LVQ2bis, DP-LVQ2+…

Nous avons rappelé brièvement l'existence du shift-LVQ puis présenté les TDNN, qui sont des perceptrons multicouches à poids partagés permettant de tenir compte grandement de l'invariance du message par translation de la forme temporelle.

Nous avons exposé succintement les modèles de Markov cachés.

Nous avons évoqué plusieurs systèmes modulaires de reconnaissance de la parole combinant perceptrons multicouches et programmation dynamique ou modèles de Markov cachés.

Nous avons proposé un système modulaire original de reconnaissanc de la parole : un système ou la sortie du MLP constitue l'entrée du module programmation dynamique. Nous avons proposé une procédure d'optimisation originale: MLP-DP-LVQ2+. Ce système est générique; il n'y a pas d'obstacle théorique à son emploi pour la parole continue et pour les grands vocabulaires. Nous l'avons testé avec succès sur un problème multilocuteur de reconnaissance de la parole en mots isolés. La mise au point de ce système représente une proportion importante de notre travail.

Pour les systèmes modulaires prédictifs combinant MLP-prédictif et programmation dynamique, nous avons également proposé une famille de fonctions de coût discriminantes, dont le PDP-LVQ2bis. Cette technique a été appliquée avec succès par A. Mellouk sur un problème de reconnaissance de la parole continue (TIMIT).

Le chapitre 6 propose une description de notre démarche expérimentale, qui a accompagné notre

---

1 **Mellouk A., Gallinari P.:** *Discriminative Training for Improved Neural Prediction Systems* - ICASSP (1994)

démarche théorique, en particulier dans la mise au point des techniques d'apprentissage telles que MLP-DP-LVQ2+ appliquables aux systèmes modulaires MLP-plus proche voisin.

6 Un système robuste de reconnaissance de la parole en mots isolés

6.1 Introduction

Nous avons comparé dans les chapitres précédents la reconnaissance de la parole en mots isolés et la reconnaissance de la parole continue. La première est plus aisée que la deuxième, mais les problèmes qu'elles posent sont pour la plupart similaires. C'est pourquoi nous avons développé un système de reconnaissance de la parole en mots isolés, pouvant s'appliquer presque directement à la reconnaissance de la parole continue.

Comme nous l'avons déjà évoqué au § 1.5, ce système est le fruit d'une progression, tant théorique que pratique. Ce chapitre a pour objet d'exposer cette progression principalement à travers notre démarche expérimentale. Il s'achève par la description complète de notre système.

Le système modulaire MLP-DP-1NN divise par trois l'erreur des TDNN, au prix d'une multiplication considérable du coût informatique: il nécessite 50 références par classe. Le système modulaire MLP-DP entrainé séquentiellement réduit substantiellement le coût informatique: il ne nécessite que 10 références par classe. Le système modulaire MLP-DP entraîné coopérativement permet d'améliorer encore le taux de reconnaissance tout en compactant radicalement le système: il ne nécessite qu'une référence par classe.

Ce chapitre est entièrement original.

6.1.2 Projet Pygmalion

Le projet Pygmalion était un projet européen ESPRIT, portant sur les systèmes neuromimétiques et leur application au traitement d'image et reconnaissance de la parole.

Notre équipe, dont quelques membres étaient connus pour avoir développé un des meilleurs logiciel neuronal du marché, fut chargée de concevoir l'environnement de développement neuronal du projet.

Le projet comportait également la réalisation d'un système de reconnaissance de la parole multilocuteur, en mots isolés à vocabulaire réduit (30 mots), en milieu bruité, pour chacune des langues des partenaires. Notre équipe fut également chargée du développement du système pour langue française dont j'allais avoir la charge durant la dernière année du projet.

Le projet Pygmalion fut considéré comme un succès et conduisit à Galatea, un nouveau projet ESPRIT plus richement doté.

6.1.3 Base de donnée de parole Pygmalion

La réalisation d'un système de reconnaissance de la parole a pour première étape la constitution d'une base de donnée.

Cinq locuteurs de chaque sexe (pour un total de 10) furent conviés à la lecture des 30 mots présentés 10 fois (réparties en plusieurs cessions) dans des ordres aléatoires, devant un micro relié à une vulgaire chaîne hi-fi, dans un bureau autour duquel le monde universitaire vaquait à ses occupations habituelles: longues conversations téléphoniques, claquement de porte, discussions, apostrophes…

Les données de chaque cession furent échantillonnées à 16 KHz par une carte électronique spécialisée. Puis elles furent segmentées par un système développé pour l'occasion. De conception classique, il se fondait la faible densité de passage par zéro et la faible intensité pour détecter les silences.

Un total de 3000 exemples furent ainsi collectés.

Plusieurs jeux de prétraitements furent testés (LPC, BDF,…). Les premiers systèmes TDNN que nous avons développés obtinrent des performances similaires sur les données issues de ces différents prétraitements. Toutes nos expériences qui suivirent employèrent les données issues de banc de filtres.

Le prétraitement par banc de filtres découle d'une transformée de Fourier (ici discrète). Seules les

énergies sont prises en considération: les phases sont oubliées. Ensuite, les énergies font l'objet d'une pondération en échelle logarithmique destinée à rendre compte de notre grande sensibilité aux hautes fréquences. Puis les fréquences sont rassemblées par bandes (en fait par fenêtres de pondérations triangulaires se recouvrant partiellement) en échelle logarithmique pour rendre compte de notre moindre discrimination des hautes fréquences entre elles. Les valeurs des paramètres de ce prétraitement nous ont été fournies par le LIMSI [1] avec lequel nous avions quelques contacts informels. Chaque exemple devint ainsi une séquence de 5 à 100 vecteurs séparés par un centième de seconde et composés de 16 coefficients réels. Par soucis d'homogénéité, chaque exemple fut présenté comme une séquence de 90 vecteurs, précédé et suivi par quelques vecteurs de silence.

Pour chaque mot, les cinq premières élocutions de chaque locuteur furent rassemblées dans la base d'apprentissage, les cinq autres dans la base de test: au total 1500 exemples dans chaque base, répartis en 30 classes.

Les expériences qui suivirent permirent d'isoler, au sein de la base d'apprentissage, environ 1% d'exemples mal segmentés et/ou trop violemment bruités pour être facilement reconnaissables. Ces exemples furent détectés sur la base d'apprentissage puis extraits. En revanche, la base de test fut laissée intacte, donc polluée.

6.1.4 Logiciels

La quasi-totalité des expériences dont la description suit furent menées à l'aide du logiciel SN [2]. Celui-ci inclut un langage Lisp enrichi de primitives graphiques et matricielles. Il inclut également un certain nombre de primitives et de librairies neuronales. Il permet l'adjonction au Lisp de fonctions écrites par l'utilisateur en langage C.

Nous avons ajouté au logiciel les procédures génériques de programmation dynamique pour la reconnaissance de la parole en mots isolés puis pour la reconnaissance de la parole continue. Pour les besoins de recherches menées par d'autres personnes de notre équipe, nous avons également ajouté un système de modèles de Markov cachés; cet ajout fut fait alors que nous avions arrêté l'exploitation de la base de données Pygmalion.

---

1 **Gauvain J. L.:** *Reconnaissance de mots enchaînés et détection de mots dans la parole continue* - Thèse de 3ème cycle, LIMSI, Orsay (1982)

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

Nous avons également ajouté de multiples fonctions pour gérer les bases de données de paroles: Pygmalion, puis TIMIT.

Enfin, nous avons écrit toutes les procédures que requéraient nos système modulaires, leur apprentissage et leurs initialisations successives.

6.2 Réseaux à délais temporels

Nous avons évoqué le principe des réseaux à délais temporels au § 5.4.2.

6.2.1 Eléments techniques

i Algorithme d'apprentissage

Nos TDNN ont tous été entraînés avec une variante de l'algorithme de rétropropagation du gradient. Cette variante, implantée en standard dans le logiciel SN, est une version adaptative de l'algorithme de Levenberg Marquardt qui est une simplification des méthodes quasi-newtoniennes (cf. § 2.3.2.). Ces méthodes qui règlent automatiquement la matrice Γ des pas d'apprentissage, ont plusieurs intérêts pratiques:

- affranchir le développeur du contrôle fastidieux de l'apprentissage,
- accélérer la convergence en termes de nombre d'itérations; cependant, ce gain est en grande partie réduit par le coût calculatoire accru de chaque itération,
- améliorer les performances finales du système.

ii Présentation des exemples

La présentation des exemples est importante, car elle permet, par adjonction de bruit et décalage temporel:

- de rendre continue la distribution empirique qui est, au naturel discontinue (problématique de l'aprentissage)
- d'étendre considérablement le nombre d'exemples et de réduire la différence entre la performance sur la base d'apprentissage et la base de test (problématique de la généralisation et de l'adéquation)

Lors de l'apprentissage, à chaque présentation, les séquences de vecteurs ont été précédées d'un

petit nombre aléatoire de vecteurs de silences, puis bruitées de façon gaussienne avec un écart-type égal au dixième de l'écart-type des données naturelles.

iii Biais non partagés

Les TDNN, en principe, effectuent des convolutions quasi-linéaires dans leurs couches inférieures.

Nos TDNN ont une particularité : l'invariance temporelle des filtres n'est imposée que pour les poids des connexions neuronales, et non pour les biais des neurones. Ceux-ci restent libres.

Cette particularité a montré son efficacité en matière de reconnaissance de mots isolés [1].

Elle peut s'expliquer physiquement par nécessité de prendre en compte la variation d'intonation et d'énergie en début et en fin d'élocution. Elle peut s'expliquer mathématiquement par un degré de liberté plus adéquat: les biais représentent alors environ la moitié des degrés de liberté la première couche de filtres.

iv Contrôle de l'apprentissage et essais d'architecture

Il existe quelques règles permettant de contrôler l'apprentissage et de guider les choix d'architectures et de conduire les apprentissages.

- De couche en couche, le nombre de neurones se réduit, afin de favoriser une synthèse progressive de l'information. Une technique qui contribue à cette réduction est le sous-échantillonnage, qui consiste à déplacer les filtres des couches inférieures de plus d'une trame. Cette technique est employée quand on n'a pas besoin d'une grande finesse temporelle dans le filtrage de l'information.

- Quand la courbe des performances sur la base d'apprentissage se met à osciller, ou à stagner, il convient de réduire le pas d'apprentissage (nous le divisons par dix). Comme nous utilisons un algorithme du second degré, cette opération n'est requise que trois fois par apprentissage.

- Du bruit gaussien et des translations aléatoires sont appliqués aux exemples présentés au réseau lors de l'apprentissage. Le bruit est réduit progressivement jusqu'à disparaître.

- L'apprentissage s'achève quand la courbe de performance sur la base de test chute car c'est

---

1 **Bottou. L.:** *Reconnaissance de la parole par réseaux multi-couches* - Neuro-Nîmes pp 197-218 EC2 (1988)

le signe que le sur-apprentissage devient plus important que l'apprentissage. Il faut mesurer la performance définitive sur une autre base de test. Dans nos expériences, en fin d'apprentissage, les courbes de performance sur la base de test n'ont jamais chuté mais se sont mises à osciller avec une amplitude de 0,5% qui se réduisait au fur et à mesure que l'algorithme convergeait. Ces oscillations proviennent en partie du bruit gaussien appliqué aux exemples présentés au réseau lors de l'apprentissage. Nous avons décidé de mener tous les apprentissages jusqu'à convergence apparente de l'algorithme.

- Les performances du système sur la base d'apprentissage indiquent dans quelle mesure l'architecture a une capacité suffisante pour traiter le problème de reconnaissance. Dans un premier temps, ceci constitue l'information sur laquelle on se focalise pour obtenir les premières architectures.

- L'écart entre les performances du système sur la base d'apprentissage et sur la base de test est une mesure du sur-apprentissage. Il est d'autant plus important que l'architecture a une grande capacité. Une mesure rudimentaire et inexacte de cette capacité est le nombre de degrés de liberté. On peut tenter de réduire cet écart en diminuant la capacité l'architecture, ce qui détériore les performances du système sur la base d'apprentissage. Dans un deuxième temps, ceci constitue l'information sur laquelle on se focalise pour orienter les choix d'architecture.

Nos expériences et nos choix d'architecture ont conduit à un écart moyen approximatif de 2% entre les performances de classification obtenues sur la base d'apprentissage et sur la base de test. Nous nous sommes concentrés sur les architectures implantant un sous-échantillonage, c'est-à-dire que les filtres quasi-linéaires des couches inférieures se déplaçaient de plusieurs trames à la fois, conduisant de couche en couche à une réduction du nombre de trames.

6.2.2 TDNN ordinaires

La figure 4.1 montre notre meilleure architecture, baptisée TDNN-IWR2.

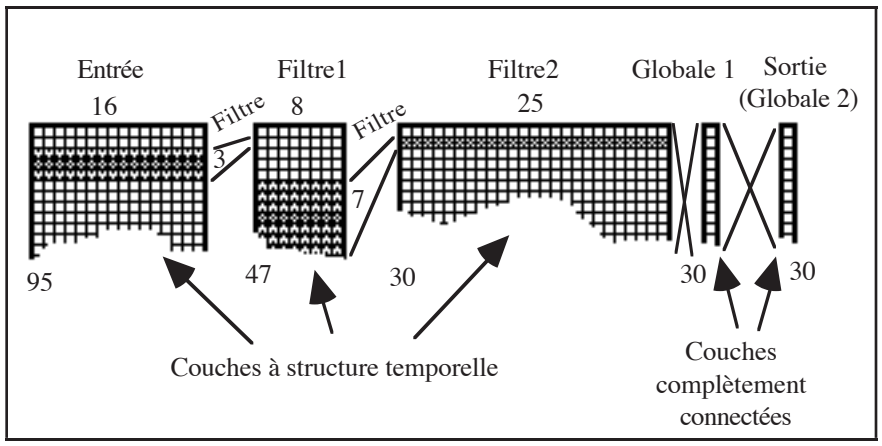


Figure 6.1
TDNN IWR2

TDNN IWR2 est le plus efficace des TDNN classiques que nous avons mis au point pour Pygmalion.
Les couches de filtres sont sous-échantillonées.

De nombreuses autres architectures ont été testées. Nous faisons mention de deux architectures performantes, toutes deux dérivées de TDNN-IWR2. Dans TDNN-IWR1, la couche “couche globale 1” a été supprimée. Dans TDNN-IWR1-16, la couche “couche globale 1” a été supprimée et la “couche filtre 1” utilise 16 unités par trame, au lieu de 8 dans TDNN-IWR2. Les résultats (arrondis au demi-pourcentage le plus proche) sont les suivants:

- TDNN-IWR2 **96.5%**
- TDNN-IWR1 **94.5%**
- TDNN-IWR1-16 **96.5%**

Plusieurs éléments méritent d'être signalés.

i Faible influence des prétraitements

Les performances des TDNN semblent être peu influencées par les prétraitements. Plusieurs prétraitements classiques proposés par littérature ont été testés: bancs de filtres, coefficients prédictifs linéaires, cepstres…

ii Caractéristiques quantitatives

Le tableau suivant résume les caractéristiques quantitatives du réseau IWR2.

Couche	Neurones	Connexions	Poids	Biais
Input	1520			
		18048	384	376
Filtre 1	376			
		12600	1400	225
Filtre 2	225			
		6750	6750	30
Globale 1	30			
		900	900	30
Sortie (Globale 2)	30			

Total	2181	38298	9434	661

Pour un tel réseau:

- la grande majorité des connexions (donc des calculs) est concentrée dans les couches à convolutions quasi-linéaires,
- la grande majorité des paramètres libres (poids et biais) est concentrée dans les couches de classification (complètement connectées et sans partage de paramètres).

Pour accélérer les calculs, on est tenté de réduire le nombre de connexions des couches inférieures du réseau. Pour réduire l'erreur de généralisation, on est tenté de s'attaquer au nombre de connexions des couches supérieures.

Il apparaît cependant délicat de supprimer *a priori* des connexions des couches de classification, dont la vocation est de relier aux sorties les valeurs issues du filtrage. De telles suppressions peuvent se faire *a posteriori* [1] mais requièrent une capacité de calcul considérable pour un gain de performance peu important pour les problèmes de classifications avec beaucoup de données comme le nôtre.

Le paragraphe suivant (6.2.3) décrit une technique employée qui réduit le nombre de connexions des couches inférieures et introduit des connaissances *a priori* supplémentaires dans l'architecture des filtres.

6.2.3 FTDNN

Dans les TDNN ordinaires, les filtres quasi-linéaires ne traitent qu'un petit contexte temporel à la fois. En revanche, sur ce contexte temporel, ils traitent toutes les fréquences.

Le formalisme neuronal permet facilement de supprimer de nombreuses connexions, afin de spécialiser les neurones de la première couche cachée sur un domaine tempo-fréquentiel. Il est alors possible d'introduire une connaissance *a priori* dans l'architecture du TDNN. Par ailleurs, on peut espérer améliorer de façon non négligeable la vitesse de calcul du réseau, car ces suppressions de connexions portent sur les couches bas-niveau, qui sont les plus gourmandes en calcul.

Une première tentative avec cette méthode a donné **97%** de reconnaissance. Le TDNN

---

1 **Le Cun Y., Denker J., Solla S.:** *Optimal Brain Damage* - Advances in Neurla Information Processing Systems 2 Morgan Kaufman (1990)

correspondant est TDNN-IWR1-16, qui obtenait un score légèrement inférieur. Les connexions fréquentielles de la première couche ont été conçues après visualisation des spectrogrammes compacts que représentent les données issues des bancs de filtres. Ainsi, les zones fréquentielles dont le signal est le plus riche en événements, ont bénéficié d'un maillage de neurones plus serré. La quantité de calcul nécessaire pour chaque itération était ainsi réduite d'un tiers par rapport au réseau TDNN-IWR1-16. Le nombre d'itérations nécessaires pour obtenir une apparente convergence du réseau a également été divisé par près de 2, d'où un temps d'apprentissage divisé par 3.

6.2.4 Synthèse des résultats obtenus avec les TDNN

Les résultats sont résumés dans le tableau suivant.

Classifieur	TDNN IWR1	TDNN IWR2	TDNN IWR1 16	FTDNN IWR1-16
Performance (%)	94,5	96,5	96,5	97
Connexions et biais	47 000	48 000	78 000	48 000

Chaque connexion "coûte" plusieurs accès, une multiplication et une addition. Les fonctions logistiques, en l'occurrence des sigmoïdes, ont un impact négligeable au regard du nombre important de connexions par neurone.

6.3 Extraction de codage par réseau à délais temporels

6.3.1 Principe

Tout TDNN possède une partie bas-niveau invariante par translation dans le temps et une partie haut-niveau agrégeant les informations traitées au cours du temps. Après l'apprentissage, la partie bas-niveau peut être extraite et utilisée pour coder les données. On obtient un codage extrait du TDNN, sur lequel les classifieurs habituels peuvent être utilisés.

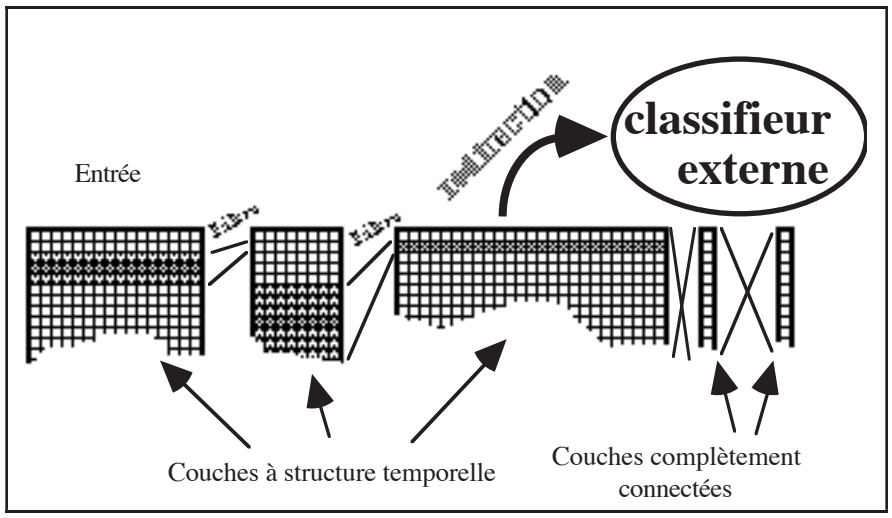


Figure 6.2
Extraction de codage

La propagation à travers le réseau est détournée à la sortie de la dernière couche de filtres vers un autre classifieurs.

Chaque TDNN entraîné est susceptible de fournir un tel codage. Lors de l'apprentissage d'un TDNN, la partie inférieure (codage) du TDNN est optimisée conjointement à la partie supérieure (classification) du TDNN.

Aussi le codage n'est pas optimisé pour l'utilisation faite par d'autres classifieurs. Néanmoins les expériences ont montré des performances généralement équivalentes et supérieures dans le cas de la programmation dynamique.

6.3.2 Mesures expérimentales

Nous avons rassemblés les résultats obtenus sur les codage extraits dans les quelques tableaux qui suivent.

i Précaution méthodologique: utilisation des MLP

Après quelques essais, une précaution méthodologique s'est imposée à nous: vérifier les performances de MLP sur les codages extraits, afin de mesurer l'impact du "réapprentissage" et la complexité de l'exploitation codage.

La combinaison Codage TDNN IWR1 + Classifieur MLP1 (resp. TDNN IWR2 + Classifieur MLP2) est fonctionnellement égale à TDNN IWR1 (resp. TDNN IWR2). La seule différence réside dans la procédure d'apprentissage: dans le premier cas MLP1 (resp. MLP2) est "réappris". Pourtant, d'importantes différences de performances apparaissent, comme le montre le tableau suivant.

	TDNN IWR1	TDNN IWR2	TDNN IWR1 16	F-TDNN IWR1
MLP1	94,5	86	**96,5**	**97**
MLP2	**96,5**	95		

RNN	**96,5**				

Les codages issus de TDNN ayant une seule couche de classification (complètement connectée) répondent à une topologie simple. Les différents classifieurs utilisés sur ces codages obtiennent de bonnes performances. Ainsi, la combinaison TDNN IWR1 + MLP2 permet d'obtenir les performances du TDNN IWR2, c'est-à-dire deux fois moins d'erreurs que TDNN IWR1 ! Ceci montre également que le codage par TDNN IWR1 est aussi riche en information que le codage TDNN IWR2.

En revanche, l'absence d'apprentissage simultané dégrade les performances des classifieurs employés sur les codages TDNN IWR2. Ainsi, la combinaison TDNN IWR2 + MLP2 conduit à deux tiers d'erreurs en plus par rapport au TDNN IWR2, pourtant structurellement identique. Le codage résultant est donc moins simple à exploiter et les classifieurs obtiennent de moins bonnes performances sur ce codage. Ceci vient probablement de ce que le codage conçu par le TDNN IWR2 est destiné à une partie haut-niveau (architecture identique à MLP2) effectuant un traitement complexe, mis au point dans TDNN IWR2 de façon conjointe au codage (ce qui permet sans doute de réduire les problèmes de minima locaux).

Les expériences qui suivent furent pour la plupart menées sur les codages extraits du TDNN IWR1 et sur les codages extraits du FTDNN IWR1. Les résultats furent similaires, avec une légère tendance (de l'ordre de 0,1% pour les systèmes ayant des performances de 98,5%) en faveur du FTDNN. Nous mentionnons dans les lignes qui suivent les résultats obtenus sur les codages FTDNN IWR1.

ii Utilisation de plus proches voisins euclidiens et par programmation dynamique

Replaçons-nous dans la chronologie de notre recherche. Nous cherchions un moyen de tirer partie des qualités des TDNN et des qualités de la DP. Aussi, nous avons introduit la notion d'extraction de codage, afin d'utiliser la programmation dynamique sur les codages extraits. Cette technique a donné d'excellents résultats dès que nous avons utilisé un réseau de la famille IWR1 comme codeur.

Le tableau suivant contient les résultats obtenus par les systèmes hybrides FTDNN IWR1 + 1NN.

Nb références / classe Algorithme	1 ref/cl	2 ref/cl	10 ref/cl	50 ref/cl	*+ du classif. pour 1 ref/classe codage: 48 000
KMeans	66,3	73,4	95		8 100
LVQ2-bis	93,8	94,1	97,3		8 100
DP				**98,5**	72 900
DP-KMeans	87,3	91,9	97,3		72 900

DP-LVQ2bis	97	97,4	**98,5**		72 900

Deux éléments apparaissent importants pour réduire le taux de mauvaise classification: l'apprentissage discriminant et la prise en compte de la dynamique temporelle grâce à l'emploi de la programmation dynamique.

Ces deux éléments sont essentiels quand on désire obtenir un système compact (ayant peu de références). Leur influence décroît progressivement avec le nombre de références utilisées.

iii Synthèse

Les systèmes modulaires TDNN + DP optimisés séquentiellement permettent de diviser par un facteur 2 le taux d'erreur des TDNN simples, au prix d'une augmentation conséquente du temps de calcul en raison du nombre important de références (dix pour chaque classe).

Ceci tend à confirmer que le TDNN effectue un traitement local, invariant dans le temps, relativement intelligent, dans sa partie bas-niveau. D'après les expériences que nous avons menées, ce codage effectué au sein du TDNN IWR1 simplifie le problème pour *tous* les classifieurs utilisés appartenant à la famille des plus proches voisins (DP, kMeans, LVQ…). Ce phénomène peut notamment être expliqué par la réduction de dimension effectuée par la partie bas-niveau des TDNN que nous avons utilisés.

Enfin, signalons un autre avantage de l'extraction de codages, dans le domaine de la vitesse d'apprentissage ou de reconnaissance. La procédure DP est huit fois plus rapide sur les codages extraits que sur les bancs de filtre, en raison de la réduction du nombre de vecteurs codant chaque mot. De même, la procédure LVQ apprend beaucoup plus vite sur ces codages que sur les bancs de filtres. En tenant compte d'améliorations telles que le seuillage, l'algorithme de programmation dynamique, sans rien perdre de sa précision nécessite en moyenne environ deux fois moins de calculs que ce que les tableaux précédents indiquent. La quantité de calcul du module DP devient alors nettement inférieure à celle du module TDNN, avec des facilités comparables en matière de parallélisation et d'implémentation en microprocesseur ou carte électronique.

6.5 Systèmes modulaires MLP-DP optimisés coopérativement

Ainsi que le paragraphe précédent l'expose, nous avons obtenu un ensemble de systèmes modulaires TDNN + classifieur. Les modules de ces systèmes ont été optimisés séquentiellement. Nous exposons dans ce paragraphe les résultats de l'optimisation simultanée

des différents modules de nos systèmes. Soulignons que ces optimisations simultanées partent d'une situation favorable, issue des optimisations séquentielles. Ces expériences ont été menées avec une seule référence par classe.

Algortihme	Score	Remarques	*+
MLP-KMeans	nul	codage ponctuel	8 100
MLP-LVQ2bis	93.0% ?	oscillations avec tendance positive puis négative	8 100
MLP-DP-KMeans	nul	codage ponctuel	72 900
MLP-DP-LVQ2bis	98.6%	classification parfaite sur base d'apprentissage	72 900
MLP-DP-LVQ2+	98,9%	classification parfaite sur base d'apprentissage	72 900

Ces optimisations se heurtent à un double problème: les fonctions de coûts proportionnelles à la variance des références et des vecteurs issus du codages (KMeans, partiellement LVQ2-bis) sont sujettes au phénomène de dégénérescence type codage ponctuel (cf. § 4.4.2.3.ii). Les fonctions de coûts non proportionnelles à cette dimension sont relativement irrégulières et sont particulièrement sujettes aux phénomènes de minima locaux. C'est pourquoi nous en sommes réduits à utiliser des techniques d'initialisation sophistiquées (optimiser chaque module séparément) avant d'employer les techniques d'optimisation simultanée des deux modules.

Nous avons ainsi obtenu un système légèrement plus efficace que le système coopératif sous-optimal; il utilise un seul représentant par classe au lieu de 50. Ce résultat est très satisfaisant, dans la mesure où le système de reconnaissance résultant est beaucoup plus petit et vingt fois plus rapide (pour une machine séquentielle).

D'autres expérience ont été menées avec plusieurs références par classes. Elles n'ont pas permis de dépasser le score de 98,9%, qui est le meilleur jamais obtenu sur cette base de donnée.

6.7 Quelques comparaisons

Nous venons d'exposer notre démarche expérimentale, qui, accompagnant notre démarche fondamentale, visait à mettre au point des systèmes modulaires hybrides MLP - DP. A cette occasion, nous avons donc présenté les performances des MLP façon TDNN, puis celles de différents systèmes hybrides.

La base de donnée française de Pygmalion a donné lieu à des mesures relatives à plusieurs autres méthodes.

Adaline

Adaline est un système linéaire. Sa performance constitue donc un point de départ. Adaline a

donné une performance de **74%**.

Shift-KMeans

Nous avons utilisé une version stochastique de kMeans: les paramètres sont mis à jour avec chaque exemple présenté. La base d'apprentissage est présentée plusieurs fois.

Afin de superviser kMeans, nous avons séparé les exemples d'apprentissage de chaque classe.

De plus, sur les bancs de filtres, nous avons utilisé une version tolérant la translation, basée sur le même principe que le LVQ tolérant la translation (cf. § 5.2.2).

Malgré sa simplicité, kMeans est une méthode de classification rapide, générale et efficace. Notre version a obtenu **82%** de succès.

Shift-LVQ2bis

Nous avons utilisé un LVQ2-bis tolérant les translations (cf. § 5.2.2). Nous initialisons les références de LVQ à l'aide de kMeans.

Notre version a obtenu **83%** de succès avec 11 références par classe, et **79%** avec 16 références par classe.

Réseaux récurrents

Le RNN que nous proposons ici est un peu particulier:

- La partie proprement récurrente du RNN est implantée avec des poids partagés, en dépliant la récurrence le long du temps. En conséquence, l'extension temporelle du RNN est finie.

- D'autre part, le réseau effectue une classification globale, et utilise une sortie globale. L'idée de base est de donner au réseau la capacité de se comporter comme une sorte d'automate à état.

La figure suivante montre l'architecture (temporellement déployée) résultante.

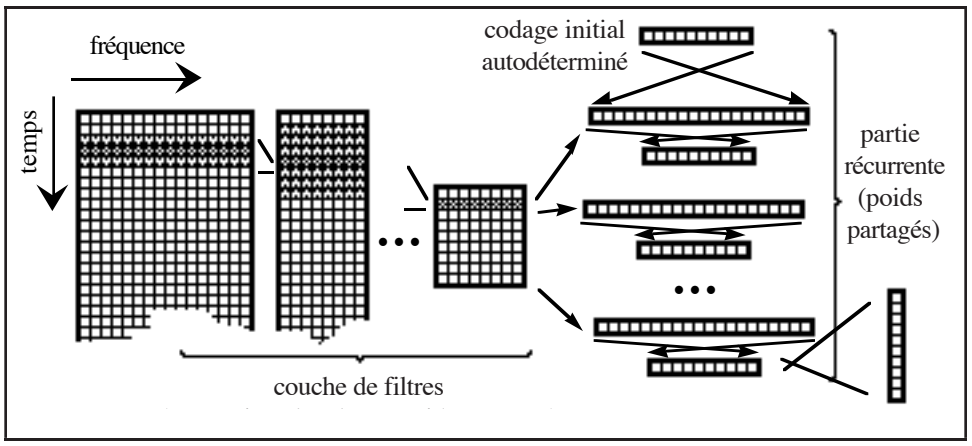


Figure 6.3
Réseau récurrent

Ce réseau récurrent a une extention temporelle finie et une architecture fixe; la partie itérative est implantée à l'aide de poids partagés.

Le réseau que nous avons mis au point donne **96%** de succès. Nous avons rencontré quelques difficultés de convergence. Le pas d'apprentissage a dû être considérablement abaissé pour éviter que le réseau ne diverge brutalement (passage d'un taux de reconnaissance de 75% à 20% en quelques passages ratés de la base d'apprentissage…) .

Programmation dynamique (DP)

Notre algorithme de programmation dynamique est très simple et utilise comme références les exemples de la base d'apprentissage. Ce DP obtient 86.7% de reconnaissance.

6.7 Conclusion

Ce chapitre présente notre démarche expérimentale, qui valide notre approche théorique des chapitres précédents, en particulier pour les systèmes modulaires MLP-DP.

Le système modulaire MLP-DP-1NN divise par trois l'erreur des TDNN, au prix d'une multiplication considérable du coût informatique: il nécessite 50 références par classe. Le système modulaire MLP-DP entrainé séquentiellement réduit substantiellement le coût informatique: il ne nécessite que 10 références par classe. Le système modulaire MLP-DP entraîné coopérativement par une fonction de coût MLP-DP-LVQ2+ permet d'améliorer encore le taux de reconnaissance tout en compactant radicalement le système: il ne nécessite qu'une référence par classe.

Il convient d'insister sur l'initialisation des composants successifs d'un système modulaire. Une fonction de coût telle que MLP-DP-LVQ2+ est irrégulière; elle est parfaitement adaptée pour l'optimisation finale d'un système bien initialisé. Employée seule, elle conduit à des minima locaux de mauvaise qualité.

Par ailleurs, il convient de rappeler que des principes proches sont à l'origine de systèmes

prédictifs discriminants de reconnaissance de la parole (cf. § 5.5)

7 Conclusion

7.1 Apprentissage numérique

L'apprentissage numérique a beaucoup progressé ces dernières années. La nouvelle puissance des ordinateurs a démocratisé le traitement de problèmes en grande dimension. Il existe de solides fondation théoriques telles que l'optimisation, le filtrage adaptatif, la décision statistique…

Les méthodes neuronales ont rapidement obtenu de bons résultats dans des domaines aussi variés que la reconnaissance de caractères, la prévision de séries temporelles ou le contrôle. Les réseaux neuronaux apparaissent aujourd'hui comme un ensemble de modèles facilement optimisables, permettant de traiter des tâches de classification aussi bien que des tâches d'approximation.

Dès lors, au lieu d'en faire une discipline à part entière, il convient de les replacer dans une perspective plus vaste. Les meilleures équipes du domaine l'ont compris. Des techniques commencent à apparaître, dont on ne sait plus très bien s'il convient de les qualifier de neuronales.

Les nouveaux systèmes sont le fruit de trois problématiques. L'adéquation vise à incorporer des connaissances *a priori* au sein d'un système. L'apprentissage consiste à définir puis employer une procédure permettant d'optimiser le comportement du système sur un ensemble de données empiriques. La généralisation a pour objet de prendre en compte la pauvreté relative des données dont on dispose. Ces trois problématiques sont généralement antagonistes; il s'agit alors d'effectuer des compromis. Ici, réside l'effort de mise au point de systèmes complexes.

7.2 Systèmes modulaires

La complexité des tâches conduit à vouloir décomposer leur traitement. On obtient des systèmes modulaires. Souvent, on se contente de mettre au point les modules isolément les uns des autres. Pourtant, seul le comportement du système dans sa globalité importe véritablement.

Or, de nombreuses techniques d'optimisation peuvent être appliquées à des systèmes modulaires. Ceci peut rendre plus cohérente et plus efficace leur mise au point. Il s'agit alors de prendre de plus grandes précautions, car de telles mises au point peuvent s'avérer erratiques ou même incohérentes. Il faut porter une attention particulière au critère d'optimisation. Il est souvent nécessaire de faire une mise au point préalable de chaque module pris séparément, avant d'effectuer l'optimisation globale.

7.3 Reconnaissance de la parole

La reconnaissance de la parole fait l'objet de nombreuses convoitises. Cette tâche est suffisamment retorse pour qu'aujourd'hui aucun produit connu ne soit satisfaisant hors un champ d'application très restreint.

Les méthodes commerciales les plus performantes sont à base de modèles de Markov cachés ou de programmation dynamique. Cependant, des méthodes hybrides combinant réseaux de neurones et programmation dynamique émergent. Largement perfectibles, elles ont déjà rattrapé et peut-être dépassé leurs rivales en matière de décodage accoustique. Elles devront être perfectionnées, allégées et surtout recevoir une expertise de leur utilisation opérationnelle.

Annexe 1 La parole

La parole est le moyen de communication qui nous est le plus naturel. Un message linguistique est transformé en onde sonore puis décodée par l'auditeur. La simplicité de cette description ne doit pas cacher la grande complexité des mécanismes mis en jeu.

Ce chapitre se propose d'en donner au lecteur une description rudimentaire. Les informations rapportées peuvent paraître relever de la simple culture générale: pour l'heure, il n'y a pas de vue globale et miraculeuse permettant de maîtriser la production de la parole ou sa compréhension. Néanmoins, les bribes d'information que l'on peut tirer de l'étude des phénomènes naturels sont très utiles; de nombreux traitements ayant prouvé leur valeur en sont issus.

Les informations contenues dans ce chapitre proviennent en grande partie de la thèse de Christophe d'Alessandro [1] et du livre de synthèse déjà vieux [2] de J.S Liénard.

A1.1 Production naturelle de la parole

L'étude de la formation naturelle de la parole permet d'identifier et de caractériser les grandes classes phonatoires.

---

1 **Alessandro (d') C. :** *La Représentation du Signal de Parole par une Somme de Fonctions Elémentaires* - Thèse de Doctorat, Université Paris VI (1989)

2 **Liénard, J.S. :** Les Processus de la Communication Parlée - Masson (1977).

A1.1.1 Système phonatoire

Le conduit vocal se compose essentiellement de la cavité du pharynx et de la cavité buccale. L'énergie de la parole provient de la pression de l'air expulsé de la cage thoracique, excitant selon les cas les cordes vocales et les parois de la cavité buccale. L'onde sonore est ensuite modifiée par l'articulation qui modifie la forme du conduit vocal, conduisant à des résonances multiples.

A1.1.2 Sources d'excitation

Trois sources d'excitation du conduit vocal transforment l'énergie de l'air chassé des poumons en onde sonore: la source sonore, la friction et l'explosion.

La source sonore provient des cordes vocales, situées à l'extrêmité du larynx. Le signal produit est quasi-périodique. Il s'agit de la voix proprement dite. La fréquence de ce signal constitue la *fréquence fondamentale* de la parole, ou *pitch*.

La constriction étroite en un point du conduit vocal constitue une source d'excitation des cavités situées en aval de cette zone. Le résultat est un bruit dont le spectre est large et uniforme, et dont les limites dépendent de la forme de la cavité. Exemples: f,v,s,z,…

L'explosion est produite par la libération soudaine de l'air retenu par occlusion vocale malgré la contraction de la cage thoracique. Exemples: b,d,g,p,t,k,…

Certaines langues utilisent les impulsions du souffle provenant de la cage thoracique pour ajouter à la richesse phonémique du langage.

A1.1.3 Articulation

L'articulation détermine la forme du conduit vocal. Les principaux constituants de cette forme sont les sections et les longueurs des cavités successives. Le conduit vocal débute par les cordes vocales et se termine par les lèvres; il est bordé par la langue, le palais, et les joues.

La cavité nasale constitue une dérivation supplémentaire commandée par l'ouverture du vélum.

La forme du conduit vocal a pour effet de filtrer les sons émis par les sources d'excitation. Le filtre qu'il constitue est caractérisé par des maxima de réponse fréquentielle. On les appelle les *formants*.

La prononciation d'un phonème élémentaire est influencée par la prononciation du phonème précédent et la préparation de la prononciation du phonème à venir. C'est ce qu'on appelle la

coarticulation.

A1.1.4 phonèmes

Les phonèmes sont définis comme les constituants oraux minimaux d'une langue. La notion de phonème n'est pas rigide. Ainsi en Français, la distinction entre certains phonèmes s'estompe (par exemple le son "un" de brun et brin).

A1.2 Reconnaissance naturelle de la parole

L'étude de l'audition [1] fait appel à deux sources d'informations principales. La première source est interne: il s'agit de la physiologie. La seconde source est externe: il s'agit de la psychologie de la perception [2] : psycho-accoustique et perception de la parole. La validation de l'étude interne est effectuée par des études externes. Nous reprenons ici des éléments épars; la plupart d'entre eux proviennent de la thèse de Christophe d'Alessandro [3].

A1.2.1 Physiologie du système auditif

L'étude physiologique du système auditif [4] est pour l'instant limitée au système auditif périphérique, par opposition au système auditif central. Ces deux systèmes sont fonctionnellement et géographiquement séparés.

Le système auditif périphérique est constitué par nos deux oreilles, incluant une partie externe, une partie moyenne et une partie interne.

A1.2.1.1 Oreille externe et oreille moyenne

L'oreille externe se compose du pavillon (ensemble cartilagineux de forme complexe) et du conduit auditif externe.

---

1 **Gribenski A.:** L'audition - Presse Universitaires de France, Paris (1951)

2 **Fraisse P. :** La Psychologie Expérimentale - Presse Universitaires de France, Paris (1966)

3 **Alessandro (d') C. :** La Représentation du Signal de Parole par une Somme de Fonctions Elémentaires - Thèse de Doctorat, Université Paris VI (Avril 1989)

4 **Keitel W.D, Neff W.D. :** *Auditory System* - in Handbook of sensory physiology, Volume V, Springer-Verlag, Berlin (1975)

L'oreille externe semble jouer un rôle dans le filtrage directionnel du son, et dans l'amplification de larges bandes de fréquences. Le premier point n'a aucune importance pour la reconnaissance de la parole, quant à l'amplification fréquentielle, son maximum est atteint vers 2500 Hz.

L'oreille moyenne se compose de la membrane du tympan et de la caisse du tympan, et de la chaîne des osselets (*marteau*, *enclume* et *étrier*) qui relie la membrane tympanique avec la fenêtre ovale.

L'oreille moyenne a essentiellement pour tâches la protection contre les sons trop puissants et une adaptation d'impédance acoustique. Les sons de forte intensité et de basse fréquence sont les plus concernés.

A1.2.1.2 Oreille interne

L'oreille interne, ou *labyrinthe*, est la partie la plus sophistiquée de l'oreille. La fenêtre ovale donne sur la *cochlée* ou limaçon: il s'agit d'une spirale; elle se termine par la fenêtre ronde. La cochlée est séparée en deux rampes successives enspiralées l'une avec l'autre. La première, reliée à la fenêtre ovale, est la rampe vestibulaire; la deuxième, remplie d'une solution salée, la *périlymphe*, et reliée à la fenêtre ronde est la rampe tympanique. Les deux rampes sont séparées par la *membrane de Reissner*, du coté de la rampe vestibulaire, puis par la *membrane basilaire*, du coté de la rampe tympanique. Le *canal cochléaire* compris entre les deux membranes contient une substance appelée *endolymphe*. La membrane basilaire porte l'*organe de Corti*, détenteur de deux ensembles de *cellules ciliées*, séparées par une arche: les cellules ciliées externes (environ 20 000) et les cellules ciliées internes (environ 3 500). Les cils de ces cellules se meuvent dans le canal cochléaire, et sont recouverts par la membrane *tectoriale* . Des fibres nerveuses (environ 30 000) sont en contact avec les cellules ciliées, principalement (95%) les cellules internes (les moins nombreuses) . Le corps de ces fibres se trouve dans le ganglion spinal, d'où part le nerf auditif.

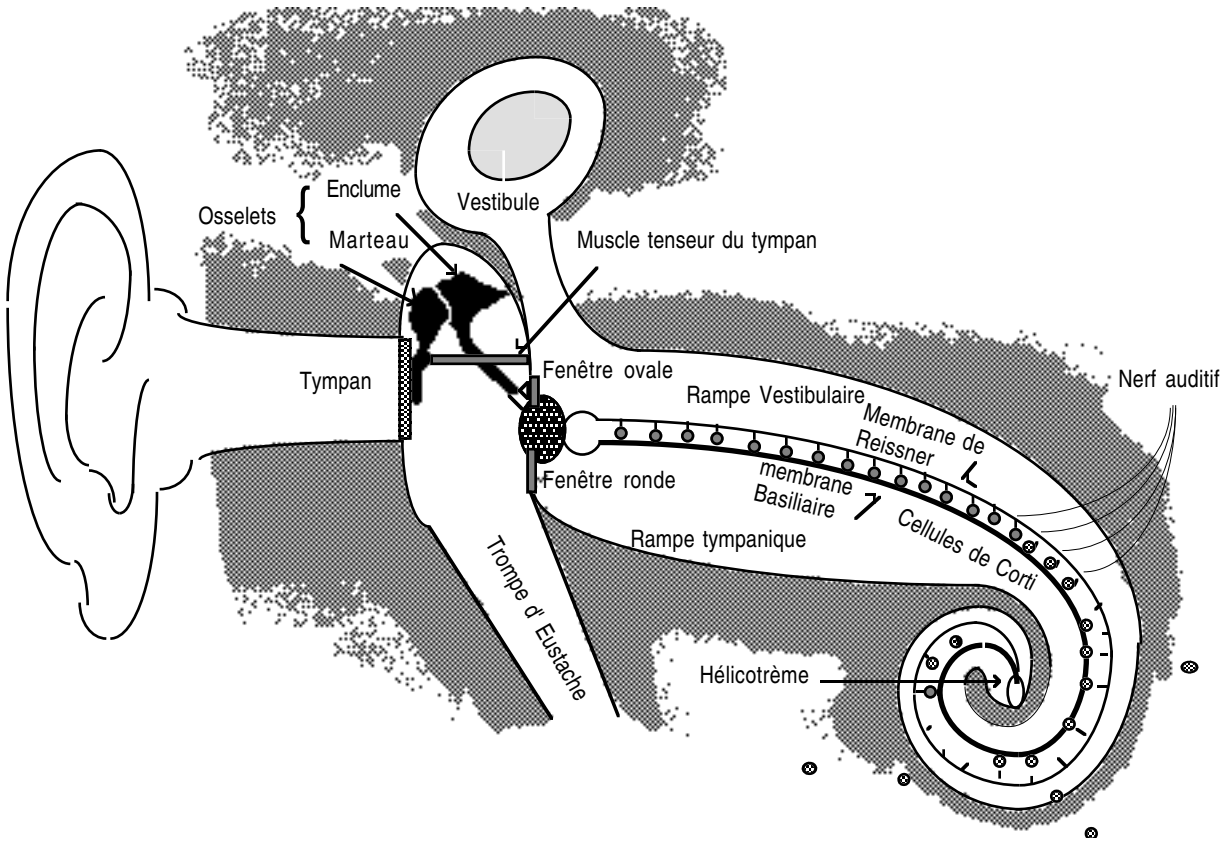


Figure A1.1: Système auditif périphérique

L'échelle utilisée pour représenter les différentes parties du système auditif n'est pas uniforme. L'échelle croît entre l'oreille externe, l'oreille moyenne et l'oreille interne.

L'oreille interne sert d'interface afin de transformer le signal mécanique en signal électrique. Trois étapes marquent cet interface:

- le mouvement de la membrane recouvrant la fenêtre ovale se transmet au fluide de la rampe vestibulaire puis de la rampe tympanique et de la membrane basilaire;
- les mouvements successifs des membranes basilaire et tectoriale agitent les cils des cellules ciliées;
- les cellules ciliées répondent en excitant les fibres nerveuses.

L'essentiel des propriétés de filtrage de l'oreille est concentré dans l'oreille interne. A l'issue de l'observation, on peut mentionner plusieurs propriétés du système auditif périphérique:

la tonotopie: une certaine fréquence a tendance à exciter préférentiellement une certaine zone le long de la membrane basilaire, de l'aigu vers le grave;

la sélectivité fréquentielle: les fibres nerveuses sont plus sélectives dans le domaine fréquentiel

pour de faibles intensités;

le contraste fréquentiel: le filtrage effectué au niveau du nerf auditif n'est pas linéaire, entraînant éventuellement un contraste fréquentiel.

la suppression à deux tons: la réponse d'une fibre à sa fréquence caractéristique diminue quand on lui superpose une fréquence proche;

l'apparition de sons subjectifs: la non-linéarité de la réponse cochléaire fait apparaître des sons subjectifs;

l'adaptation à court terme: la transduction des cellules ciliées aux fibres nerveuses crée un contraste temporel.

le codage non-linéaire de l'intensité: le codage de l'intensité dans une fibre se traduit par une augmentation du taux de décharge dés que l'excitation est supérieure à une valeur minimale. Au delà d'un certain seuil, il y a saturation. La sensation d'intensité est liée à un codage quasi-logarithmique (de même que l'échelle des volumes sonores: dB);

la distribution retardée: du fait de l'étalement des sensibilités le long de la cochlée de l'aigu au grave, la perception sons graves est retardée;

l'hétérogénéité de la population des fibres nerveuses: quelques fibres semblent ne réagir que pour les fortes intensités;

A1.2.2 Psycho-acoustique

La psycho-acoustique est la conscience de la perception des phénomènes acoustiques. L'étude de la psycho-acoustique est extrêmement instructive sur les transformations effectuées par notre système auditif.

A1.2.2.1 Sensation de hauteur

La hauteur perçue dépend de la fréquence de répétition de la forme d'onde d'un son. L'information principalement concernée est la fréquence fondamentale. La hauteur perçue d'un son quasi-stationnaire est fonction quasi-logarithmique de la fréquence de ce son. L'échelle des Mels rend compte de ce phénomène:

$$f_{mels} = 2595 \log_{10} \left(1 + \frac{f_{Hz}}{700} \right) . \qquad \text{(A1-1)}$$

La perception des hauteurs est fine et complexe. Ainsi:

- Un son possédant plusieurs fréquences, multiples d'un fondamental absent physiquement, aura une hauteur égale au fondamental virtuel.

- Une translation de fréquences harmoniques du son décrit au paragraphe précédent, engendre la perception d'une hauteur partiellement translatée.

- Un son répété aura sa hauteur déterminée par l'écart entre les deux répétitions, si cet écart est suffisamment court.

- Certains sons évolutifs engendrent l'étrange sensation d'une hauteur simultanément croissante et décroissante.

A1.2.2.2 Sensation de force

La sensation de force d'un son est, elle aussi non linéaire. Pour les fréquences moyennes, l'échelle de la sensation de force est quasi-logarithmique. Par ailleurs, des sons d'intensité égale mais de fréquence différente, donneront lieu à des sensations de forces différentes: dans des circonstances précises, une sensation de force donnée peut être conçue comme un niveau sur une carte dont les deux axes représentent respectivement l'intensité et la fréquence.

A1.2.2.3 Filtrage psycho-acoustique

L'appareil auditif effectue une analyse spectrale. On peut se demander si des filtres fréquentiels sont utilisés lors du traitement cérébral de l'information auditive.

On peut citer, en réponse à cette interrogation, les bandes critiques. La combinaison de deux sons purs donne lieu à une sensation de force qui correspond à l'addition des deux sensations de force ou bien à l'addition de leurs deux intensités physiques dans le cas ou les deux fréquences font partie de la même *bande critique*. L'étalement des bandes critiques est donnée approximativement par l'échelle Bark [1]:

$$f_{Hz} = 600 \, \mathrm{sh}\left(\frac{f_{Bark}}{6} \right) . \qquad \text{(A1-2)}$$

L'échelle Bark est très utilisée pour les systèmes de traitement automatique de la parole.

On peut aussi citer l'effet de masque. Le seuil de perception d'un son en présence d'un autre son varie en fonction de l'intensité et de la fréquence de cet autre son. L'effet de masque est

---

1 **Seiker A., Hanson B.A. :** *Improved 1-Bark bankwidth auditory filter* - in JASA, Vol. 75 No 6 (Jully 1984)

maximum pour des fréquences proches, l'effet de masque croît moins vite que le niveau du son masquant, les fréquences aiguës sont plus facilement masquées par des fréquences graves que l'inverse.

A1.2.2.4 Sons subjectifs

Il est possible de percevoir des sons subjectifs en l'absence d'énergie dans la région spectrale concernée. Il semble que les sons subjectifs se forment dans l'oreille interne.

A1.2.3 Perception de la parole

La perception de la parole n'est pas simplement dûe à la psycho-acoustique. La représentation cérébrale de l'information acoustique subit de nombreux traitements, fruits d'un long apprentissage, avant que l'information linguistique n'émerge.

Les indices perceptifs qui permettent de discriminer des entités phonétiques sont redevables à leur mode de production. Il existe même une théorie motrice, selon laquelle la perception de la parole serait permise par l'identification inconsciente du geste articulatoire de sa production.

On peut détailler la liste des sons par catégories phonétiques, et la perception qu'on en a.

sons vocaliques: la perception des sons vocaliques est principalement dûe aux deux premiers formants. Néanmoins, des éléments viennent atténuer ce fait: voix chuchotée, voyelles nasales. De plus, une élocution rapide empêche la stationnarité des sons vocaliques, sans nuire à une interprétation convenable.

sons fricatifs: la perception des sons fricatifs fait appel à des caractéristiques statiques (centre de gravité du bruit) et dynamiques (évolution du contexte vocalique) .

sons plosifs: les plosives sont caractérisées par l'explosion suivant l'occlusion, et par l'évolution des formants du contexte vocalique, révélatrice du lieu d'articulation.

De plus il faut noter que la reconnaissance naturelle de la parole se fait malgré le bruit et les conversations parasites environnants. Cette résistance étonnante de notre système de reconnaissance fait probablement appel à l'utilisation de critères de cohérence faisant intervenir de nombreux éléments (vision de l'articulation de son interlocuteur, intuition de l'articulation à venir, …) et adaptatifs (type de bruit, fréquence fondamentale du locuteur …).

A1.3 Traitements bas niveau

La formation de la parole suggère deux constituantes: analyse prédictive linéaire et analyse fréquentielle.

La reconnaissance naturelle de la parole dérive pour l'essentiel de l'analyse fréquentielle. L'étude du système auditif périphérique, de la psycho-acoustique et de la perception de la parole nous fournissent des indices précieux, jetant les bases de modèles fonctionnels du système auditif global[1]. Cette source d'inspiration a permis d'obtenir les traitements bas niveau les plus utilisés dans le monde de la reconnaissance de la parole.

Par ailleurs, certains paramètres très simples mais utiles peuvent être obtenus à l'aide d'une analyse temporelle du signal.

D'autres domaines que la parole font appel au traitement du signal: analyse de signaux radar et Hertziens, transmission par réseau, …

Néanmoins, rien n'indique que ces traitements bas niveau sont suffisamment bons pour ne pas dégrader excessivement l'information utile apportée par la parole. Le problème des traitements bas niveau n'est donc pas résolu.

A1.3.1 Obtention d'un signal digitalisé

Le signal sonore se manifeste par des variations locales de la pression: on est en présence d'une onde, qu'on peut supposer localement plane.

La transformation de ce signal, continu dans la dimension de la pression et la dimension du temps en un signal digitalisé, échantillonné et discrétisé, est nécessaire pour traiter informatiquement le signal sonore. Certaines précautions doivent être prises lors de cette opération.

D'une part, la discrétisation du signal doit permettre de rendre compte de sa dynamique: de petites variations peuvent être porteuses d'informations vitales.

D'autre part, la fréquence d'échantillonnage doit être suffisante pour éviter les repliements spectraux: l'analyse de Fourier d'un signal digitalisé fait apparaître un repliement des fréquences

1 **Chistovich I.A., Granstrem M.P., Kozhevnikov V.A., Lesogor L.W., Shupljakov V.S., Taljasin P.A., Tjulkov W.A. :** *A functional model of signal processing in the périférical aauditory system* - Acustica, Vol. 31, (1974)

supérieures à la moitié de la fréquence d'échantillonnage sur les fréquences plus basses. Pour éliminer le repliement des hautes fréquences, on utilise préalablement un filtre passe-bas. Néanmoins, il est nécessaire que la fréquence d'échantillonnage soit suffisamment élevée pour conserver les hautes fréquences porteuses d'informations utiles.

On obtient, à l'issue de la digitalisation du signal une suite de valeurs de {1, … , P }.

La digitalisation s'effectue typiquement sur deux octets (seize bits) , avec une fréquence de 10 à 20 KHz, selon le luxe dont on veut se doter et la langue que l'on étudie; par exemple, l'existence de fricatives à très hautes fréquences en Anglais nécessite un échantillonnage à au moins 16 KHz.

A1.3.2 Analyse temporelle

Trois paramètres sont fréquemment utilisés: l'énergie totale, la densité de passage par zéro, et l'autocorrélation.

A1.3.2.1 Energie totale

Le signal sonore est assimilable à une combinaison d'ondes planes dont la densité d'énergie moyennée au cours d'une tranche temporelle de longueur N s'écrit:

$$E_n = \frac{1}{N} \sum_{k=n}^{n+N-1} S(k)^2 \ . \qquad \text{(A1-3)}$$

Une énergie particulièrement forte caractérise les voisements, en particulier les voyelles.

A1.3.2.2 Densité de passage par zéro

Notre appareil auditif est très sensible aux émissions hautes fréquences. La densité de passage par zéro de la surpression constituant le signal sonore permet de déceler, quand l'énergie du signal est faible, l'existence d'un émission haute fréquence, peu énergétique mais porteuse d'informations importantes, caractérisant les fricatives.

A1.3.2.3 Autocorrélation

L'autocorrélation du signal s'écrit, pour une tranche de signal de longueur N:

$$A_n(p) = \frac{1}{N} \sum_{k=n}^{n+N-1} S(k)\ S(k+p) \ . \qquad \text{(A1-4)}$$

En présence de voisement, ce coefficient est particulièrement élevé pour les valeurs de p multiples de la fréquence fondamentale du voisement. L'autocorrélation permet donc de déterminer le *pitch*.

A1.3.2.4 Etiquetage et segmentation de bases de données

Le stockage de la parole en vue d'un examen et d'une mise en correspondance du signal de parole et ses correspondances symboliques que nous percevons (phonèmes, mots, phrases), nécessite un étiquetage.

Si les différents symboles sont séparés par des silences, on peut faire appel à des éléments simples pour segmenter le signal. Un silence est caractérisé par une faible énergie et par une densité de passage par zéro peu importante. Néanmoins, des éléments extérieurs à la parole peuvent se trouver enregistrés par le système. Le plus souvent, des ajustements manuels sont ainsi nécessaires.

Si les différents symboles ne sont pas séparés par des silence, on est obligé, soit d'effectuer un étiquetage manuel, soit d'utiliser un système de reconnaissance déjà opérationnel, et de corriger manuellement les erreurs du système. Le procédé est fastidieux.

Les bases de données de taille suffisante sont donc rares; il est souvent difficile de les obtenir auprès de ceux qui en disposent. Néanmoins, ces dernières années, de grosses bases de données ont été largement diffusées.

A1.3.3 Analyse prédictive linéaire

L'idée de base de l'analyse par prédiction linéaire est qu'un échantillon de signal à un instant donné peut être approché par une combinaison linéaire d'échantillons passés. On minimise:

$$D_{N0,N1,n} = \sum_{N=N0}^{N0+N1} \left| \left(\sum_{k=1}^{n} a_k \, x_{N-n-1-k} \right) - x_N \right|^2 \quad , \qquad \text{(A1-5)}$$

pour une tranche {N0-n, … ,N1-1} d'un signal (x_i) donné.

Si on considère que le signal est le produit de convolution d'un signal d'excitation e_n dans un filtre H, on peut représenter le signal x_n par l'équation récurrente:

$$x_n = h_0 \, e_n + \sum_{k=1}^{n} a_k \, x_{n-k} \quad , \qquad \text{(A1-6)}$$

L'analyse prédictive linéaire modélise cette expression.

A1.3.4 Analyse fréquentielle

L'analyse fréquentielle ou analyse spectrale consiste à extraire les paramètres dans le domaine fréquentiel.

A1.3.4.1 Spectrogramme

L'analyse de Fourier conduit à l'obtention d'une énergie et d'une phase pour chaque fréquence analysée. La phase n'est pas utile pour la perception de la parole, elle ne sert qu'à déterminer l'origine spatiale du signal sonore quand on le perçoit avec deux récepteurs proches, telles nos fidèles oreilles. Le résultat est un *spectre d'énergies*, ou *spectrogramme*.

On cherche à ne conserver de cette information volumineuse que ce qu'on pense utile afin de compresser l'information et mettre en valeur ses caractéristiques. Différents traitements ont été mis au points et viennent condenser le spectrogramme.

A1.3.4.2 Bancs de filtres

Parmi ces traitements, citons les bancs de filtres, étendus sur une échelle, en général quasi-logarithmique, telle que Mel ou Bark. Chaque vecteur du spectrogramme, ou trame, est résumé à l'aide d'un moyennage pondéré des énergies de fréquences proches. Le fenêtrage fréquentiel est étalé sur l'échelle; chaque fenêtre regroupe plusieurs coefficients du spectrogrammes en un coefficient résultat. Ainsi, dans le cas d'une échelle quasi-logarithmique, les hautes fréquences sont moins différenciées que les basses fréquences. Puis on opère un codage logarithmique des coefficients ainsi obtenus.

Ce traitement bas-niveau respecte certaines des principales caractéristiques connues de notre système auditif.

A1.3.4.3 Cepstres

Une modélisation linéaire de la production de la parole permet de considérer le signal comme le produit de convolution du signal source et de la fonction de transfert du conduit vocal. La transformée de Fourier transforme le produit de convolution en produit simple. Le passage au logarithme transforme le produit simple en somme. La différence de variabilité entre les deux fonctions permet de séparer les composantes du signal à l'issue d'une transformation de Fourier inverse.

A1.3.4 Analyse en ondelettes

L'analyse en ondelettes est souvent présentée comme une future solution miracle.

Au delà de ce vœu compréhensible, mais peut-être vain, il faut reconnaître que l'analyse par ondelettes recèle par sa nature un potentiel réel de détection événementielle à des échelles différentes [1]. L'analyse par ondelettes n'est pas encore très répandue dans le domaine du traitement de la parole. Certains affirment que des techniques telles les bancs de filtres constituent une forme simplifiée d'ondelettes.

A1.4 Conclusion sur le signal de parole

L'étude de la production et de la perception de la parole, bien qu'incomplète, nous fournit un enseignement essentiel. Les applications directes qu'on peut en avoir sont les traitements bas-niveau, permettant de représenter de manière dense la parole, en conservant autant que possible les informations utiles qu'elle contient. Quand on travaille dans le domaine du traitement de la parole, la connaissance de ce sujet relève de la culture générale. Il est donc hautement souhaitable d'en disposer: elle fournit une culture générale permettant de concevoir des modèles dédiés à la parole (DP,…) et des adaptations de modèles généraux (TDNN,…).

---

1 **Mallat S.G. :** *A theory for Multiresolution Signal Decomposition: the Wavelet Representation* - Technical report MS-CIS-87-22, University of Pensylvania,Philadelphie, (1987)

Annexe 2 Un autre exemple de système modulaire

Nous reproduisons dans cette annexe une publication (IJCNN 1992) présentant un autre exemple de système modulaire. Nous avons choisi de présenter ce travail en annexe afin de ne pas rompre la progression du corps principal du mémoire: théories fondamentales (cf. § 2, § 3) systèmes algorithmiques (cf. § 4, § 5), enfin expériences de reconnaissance de la parole (cf. § 6).

Il s'agit ici de faire coopérer un perceptron multicouches et une carte topologique de Kohonen. Un apprentissage coopératif des deux modules conduit imparablement au phénomène de codage ponctuel (cf. § 4.42.3.ii). La solution que nous employons ici est celle de l'ajout d'une contrainte d'intégrité dans la fonction de coût MLP-TMap. Nous utilisons comme contrainte d'intégrité le maintient de la variance des codages à plus de un. Nous décrivons l'algorithme adaptatif qui en résulte.

A modular system which improves the topological maps

H. Schwenk, P. Gallinari, X. Driancourt

Université de Paris Sud • CNRS URA 410

Laboratoire de Recherches en Informatique • Batiment 490

91405 Orsay Cedex • France

A2.0 Abstract

Recently, complex neural net systems built from several modules which cooperate to a global task have raised much interest. Although very powerful, these systems may perform poorly in practice due to generic problems which must be studied from a theoretical point of view. We propose here as an example a system which makes cooperate a linear network with a topological map (TM). It allows to drastically reduce the computing time of TM. In order to train the hybrid architecture, new cost functions and algorithms are proposed.

A2.1 Introduction

Topological maps (TM) [Kohonen 88] are one of the most popular unsupervised Neural Net (NN) technique. They have been used in several applications ranging from combinatorial optimization to speech or image processing. Their applicability is however limited because their computational complexity drastically increases with the input space dimension and the complexity of the data. Several attempts have been performed in order to reduce this computational burden, e.g. [Lampinen &Oja 90] have proposed a fast algorithm for the detection of the most active output cell.

We propose here a different approach to the problem. Complexity of TM is proportional to the input and output layer size. Most computation time is spent on distances between input and reference vectors. Output dimension depends both on the data structure and on the desired precision of the analysis. On the other hand, it is well known that the intrinsic dimensionality of most high dimensional data is much smaller than the input space dimension. Computational complexity of the TM could be diminished if we were able to optimaly reduce the input dimension of the data through a projection module whose output would then be sent to a TM.
More generally, complex tasks are often decomposed into simpler subtasks which are solved sequentially through different modules. These systems present two drawbacks: they are not adaptive and the assumed decomposition is not guarantied to be optimal. We propose here a hybrid NN architecture and adaptive algorithms which allow to train both modules (projection and TM) simultaneously. It is thus a fully adaptive and optimal system. It offers both a considerable speed up and an increase of the performances compared to the original TM.

This architecture is a particular case of hybrid or multi-module architectures which have raised much interest recently [Bollivier et al.90, Bottou & Gallinari 90, Bourlard 90, Driancourt et al.91, Jacobs & Jordan 90]. These systems present generic problems (training,convergence) for which there is now no general solution. Putting our system in this general framework allows

to investigate some of these problems and to propose solutions which are expected to be general.

In the following, we first show that TM may be expressed as an adaptive gradient algorithm for the minimization of a cost function. We then propose a new system and training algorithms. We discuss convergence issues which allow to raise generic problems in the general framework of multi-modules architectures and propose solutions. Finally we present some tests which illustrate the behaviour and performances of the algorithms.

A2.2 Algorithm

A2.2.1 Topological maps as an adaptive gradient algorithm

Most unsupervised learning algorithms can be shown to minimize some cost function whose general form is [Bottou & Gallinari 91]:

$$C(\zeta) = \sum_k \int_{Z_k} \theta_k(\mathbf{z},\zeta)\, F_k(\mathbf{z},\zeta)\, p(\mathbf{z})\, d\mathbf{z} = E\left(\sum_k \theta_k(\mathbf{z},\zeta)\, F_k(\mathbf{z},\zeta)\right) \qquad \text{(A2-1)}$$

where p is the mixture probability of input data $\mathbf{z}$, ζ are the parameters of the model, and the local cost function upon presentation of sample $\mathbf{z}$ is $F_k(\mathbf{z},\zeta)$ if $\mathbf{z}$ is classified into region Z_k. θ_m is the characteristic function of region Z_m as it is defined by the current parameters of the system i.e.:

$$\theta_m(\mathbf{z},\zeta) = \begin{cases} 1 \text{ if } \mathbf{z} \in Z_m \text{ , i.e. } m = \underset{k}{\text{Argmin}}\, F_k(\mathbf{z},\zeta) \\ 0 \text{ otherwise} \end{cases} \qquad \text{(A2-2)}$$

It should be noted that θ_m depends on the current partition $(Z_m)_{1\le m\le M}$ of input space Z and not on the actual class of $\mathbf{z}$. The simplest adaptive learning rule for these algorithms is then:

$$\zeta := \zeta - \varepsilon \nabla_\zeta F_k(\mathbf{z}, \zeta) \qquad \text{if } \mathbf{z} \text{ has been classified in region } Z_k \qquad \text{(A2-3)}$$

where ∇_ζ is the gradient with respect to ζ. The original TM algorithm [Kohonen 88] does not enter directly this formalism. However it is easy to infer variations of TM which fit the formalism. For example, the algorithm proposed by [Angeniol et al. 88] makes use of a continuous neighbourhood function on the output layer which measures how modifications in one cell will influence the modification in others. From this algorithm, one can infer the following local cost function:

$$F_i(\mathbf{z},\mathbf{C}) = \sum_k \beta_{ik} \|\mathbf{z}-\mathbf{c}_k\|^2 \qquad \text{(A2-4)}$$

where $\mathbf{C}$ is the weight matrix, $\mathbf{c}_k$ is the weight vector to the k^{th} cell and β_{ik} is a decreasing function of the distance d_{ik} in the output layer between position of cells i and k in the topology of this layer, e.g. $\beta_{ik} = 1/d_{ik}$. Cost function (1) becomes:

$$C(\mathbf{C}) = E\Big(\sum_i \theta_i(\mathbf{z},\mathbf{C}) \sum_k \beta_{ik} \|\mathbf{z}-\mathbf{c}_k\|^2 \Big) \qquad \text{(A2-5)}$$

The learning algorithm for weight vector $\mathbf{c}_k$ can be written:

$$\mathbf{c}_k := \mathbf{c}_k + \varepsilon\beta_{ik}(\mathbf{z}-\mathbf{c}_k) \qquad \text{(A2-6)}$$

where i is the winning output cell. Factors ε and β_{ik} are often combined into one function, e.g.: $\varepsilon(t,d) = \varepsilon_0 e^{-(d/T^2)}$ where T decreases during learning. Original TM makes use of a discrete neighbourhood function.

A similar derivation of (6) from (5) has been proposed by [Ritter & Schulten 88]. Theoretical results about the behaviour and convergence of this algorithm have been shown for particular cases [Kohonen 88, Ritter & Schulten 88]. The cost function presents several local minima, however the algorithm performs well in practice.

A2.2.2 A modular architecture

During training or recognition, most of the time is spent computing distances between input and reference vectors. Considerable improvement could be achieved if we were able to reduce the dimension of the data processed by the map. In order to do this, we will first project the input data **z** onto a smaller dimensional space, the projection **x** being then sent to a TM. This could be implemented using the network in figure 1.

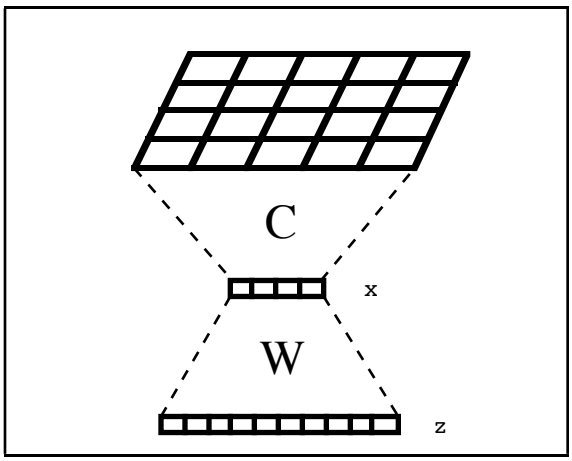


Figure A2.1
Structure of the hybrid network for a two dimensional output map

W and C denote respectively the weight matrices of the projection module and of the TM module. The respective dimensions of the three layers are from input to output: di, dh, do.

The propagation of the information through this system is :

$$\mathbf{x} = \mathbf{W}\mathbf{z} \qquad \text{(intermediate projection)}$$

$$i = \underset{k}{\operatorname{argmin}} \|x-c_k\|^2 \qquad \text{(output cell selection)} \qquad \text{(A2-7)}$$

The problem is now to train this network in an optimal way while preserving the information of interest during the projection. Usually such projections are defined so as to optimize ad hoc criteria such as the variance or the discriminant information. Of course, it would be far better to *optimize the projection according to the global goal of the system* which in this case is the minimization of (5). This is exactly what our system does.

A2.2.3 Training algorithms

Because the TM algorithm has been shown to optimize global cost function (5), there is a straightforward extension of this algorithm to the new architecture. Using the notations of figure 1, the local cost function (4) may now be written

$$F_i(\mathbf{z},\mathbf{W},\mathbf{C}) = \sum_k \beta_{ik} \|\mathbf{W}\mathbf{z}-\mathbf{c}_k\|^2 \qquad \text{(A2-8)}$$

The corresponding adaptive algorithm is then:

$$\mathbf{c}_k := \mathbf{c}_k + \varepsilon_1 \beta_{ik} (\mathbf{x}-\mathbf{c}_k) \qquad k \in 1..d_o \text{ and i is the winning output cell} \qquad \text{(A2-9)}$$

$$\mathbf{W} := \mathbf{W} - \varepsilon_2 \sum_k \beta_{ik} (\mathbf{x}-\mathbf{c}_k)\, \mathbf{z}^t$$

This simple algorithm presents several problems:

- ε_1 and ε_2 should be carefully tuned to prevent the weights from exploding,
- it converges towards strong minima of (5), e.g. $\mathbf{w} = 0$, which are uninteresting.

The same phenomenon has been observed for many modular hybrid systems [Bollivier et al.90] and many algorithms which were expected to work well actually perform poorly in practice. It is thus of interest to develop general solutions to these drawbacks.
These problems may be handled by using constraints which may be explicitly incorporated into the cost function or implicitly introduced by modifying the model (e.g. architecture or transition functions). These constraints may also be used to incorporate some *a priori* knowledge about the nature of the solution. In our case:

- weights may be constrained to a limited range (constraint 1);
- convergence towards zero may be prevented by imposing a fixed distortion onto the hidden cells space (constraint 2).

We are now faced with the problem of deriving adaptive algorithms in the presence of constraints.

A2.2.4 Learning in the presence of constraints

Although there is an abundant literature about classical constrained optimization, the problem of applying adaptive algorithms to constraint problems has received little attention. Most classical algorithms cannot be implemented adaptively. In the following, we will consider for simplicity equality constraints of the form $\mathbf{g}(\zeta) = E(\mathbf{h}(\mathbf{z},\zeta)) = 0$ where $\mathbf{g}$ and $\mathbf{h}$ are m-valued functions of respectively ζ and $(\mathbf{z},\zeta)$. Minimizing (5) under such constraints is not a classical optimization problem. We will mention two methods which are based respectively on the use of a penalty function and on Lagrange multipliers.

The method based on the Lagrange function is discussed extensively in [Tsypkin 71]. We will consider as the new goal of learning the minimisation of:

$$C(\zeta,\lambda) := C(\zeta) + \lambda^t \mathbf{g}(\zeta)$$

where λ are Lagrange multipliers. In this case, adaptive algorithm (3) becomes

$$\zeta := \zeta - \varepsilon_1(\nabla_\zeta F_k(\mathbf{z},\zeta) + \nabla_\zeta \mathbf{h}\, \lambda) \text{ if } \mathbf{z} \text{ has been classified in region } Z_k \qquad \text{(A2-11)}$$

$$\lambda := \lambda + \varepsilon_2 \mathbf{h}(\mathbf{z})$$

where: $\nabla_\zeta \mathbf{h} = \left[\frac{\partial h_i}{\partial c_j}\right]_{i=1..m,j=1..n}$ is a n * m matrix

Note that in (11) λ is an adaptive indicator and its update rule is not a gradient descent on (10) [Tsypkin 71].
In the following, we will denote:

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}_1^t \\ \cdot\cdot \\ \mathbf{w}_p^t \end{pmatrix} \ , \ \Lambda = \begin{pmatrix} \lambda_1 & 0 \\ & \cdot\cdot \\ 0 & \lambda_p \end{pmatrix} ,$$

$\mathfrak{A}$ the operator which transforms a vector into a diagonal matrix. (A2-12)

Constraint 1 may be given the explicit form $g_j(\mathbf{W}) \overset{\Delta}{=} \|\mathbf{w}_j\|^2 - 1 = 0$, $j = 1..d_h$, where $\mathbf{w}_k$ is the weight vector to hidden unit k. The corresponding algorithm is thus:

$\mathbf{c}_k := \mathbf{c}_k + \varepsilon_1 \beta_{ik} (\mathbf{x} - \mathbf{c}_k)$ $k = 1..d_o$ and i is the winning output cell

$$\mathbf{W} := \mathbf{W} - \varepsilon_2 \Big(\sum_k \beta_{ik} (\mathbf{x} - \mathbf{c}_k) \mathbf{z}^t + \Lambda \mathbf{W} \Big) \qquad \text{(A2-13)}$$

$$\Lambda := \Lambda + \varepsilon_3 \mathfrak{A}\mathbf{h} \quad \text{with } \mathbf{h} := \begin{pmatrix} \|w_1\|^2 - 1 \\ \cdots \\ \|w_p\|^2 - 1 \end{pmatrix}$$

Constraint 2 may be written $g_j(\mathbf{W}) = E((x_j - E(x_j))^2 - 1) = 0$, $j = 1..d_h$, which leads to:

$\mathbf{c}_k := \mathbf{c}_k + \varepsilon_1 \beta_{ik} (\mathbf{x} - \mathbf{c}_k)$ $k \in 1..d_o$ and i is the winning output cell

$$\mathbf{W} := \mathbf{W} - \varepsilon_2 \Big(\sum_k \beta_{ik} (\mathbf{x} - \mathbf{c}_k) \mathbf{z}^t + \Lambda (\mathbf{x} - \mathbf{m_x})(\mathbf{z} - \mathbf{m_z})^t \qquad \text{(A2-14)}$$

$$\Lambda := \Lambda + \varepsilon_3 \mathfrak{A}\mathbf{h}$$
$$\mathbf{m_x} := \mathbf{m_x} - \varepsilon_4 (\mathbf{x} - \mathbf{m_x}) \quad \text{with } \mathbf{h} := \begin{pmatrix} \|x_1 - m_1\|^2 - 1 \\ \cdots \\ \|x_p - m_p\|^2 - 1 \end{pmatrix}$$

where $\mathbf{m_x}$ is an adaptive estimation of the mean value of vector $\mathbf{x}$ and $\mathbf{m_z}$ is the mean value of input data z.

Penalty methods [Scales 85] consist in adding a term to the cost function so as to softly enforce the constraints. The new cost function will be:

$$J(\zeta) := C(\zeta) + \alpha^t \mathbf{F}(\mathbf{g}(\zeta)) = E\Big(\sum_k \theta_k(\mathbf{z},\zeta) \, F_k(\mathbf{z},\zeta) + \alpha^t \mathbf{F}(\mathbf{h}(\mathbf{z},\zeta)) \Big) \ . \qquad \text{(A2-15)}$$

$C(\zeta)$ has been defined in (1), $\mathbf{F}(\mathbf{g}(\zeta))$ the penalty function is usually $g^2(\zeta)$ i.e; $(g_1^2(\zeta),\ldots, g_I^2(\zeta))^t$, α is a vector of coefficients. Algorithm (3) may be used for the minimization of J and writes for $\mathbf{F}(\mathbf{g}) = g^2$:

$\zeta := \zeta - \varepsilon(\nabla_\zeta F_k(\mathbf{z},\zeta) + (\nabla_\zeta \mathbf{h}) \mathfrak{A}\alpha \, \mathbf{h})$ if $\mathbf{z}$ has been classified in region Z_k (A2-16)

The algorithm for constraint 2 is thus:

$$\mathbf{c}_k := \mathbf{c}_k + \varepsilon_1 \beta_{ik}(\mathbf{x}-\mathbf{c}_k) \quad k \in 1..d_o \text{ and i is the winning output cell} \qquad \text{(A2-17)}$$

$$\mathbf{W} := \mathbf{W} - \varepsilon_2 \Big(\sum_k \beta_{ik} (\mathbf{x}-\mathbf{c}_k)\, \mathbf{z}^t + \sum \alpha \sum \mathbf{h}(\mathbf{x}-\mathbf{m}_\mathbf{x})(\mathbf{z}-\mathbf{m}_\mathbf{z})^t \Big)$$

$$\mathbf{m}_\mathbf{x} := \mathbf{m}_\mathbf{x} - \varepsilon_3(\mathbf{x}-\mathbf{m}_\mathbf{x})$$

Where notations are as above. This method is easy to use, but the constraints are not exactly fulfilled and the coefficients in α may be difficult to set when there is a large number of constraints.

A2.3 Experiments

A2.3.1 Data sets

We have performed experiments using two databases: the irises of Fisher [Fisher 65] which is may be the best known data set in data analysis and an artificial data set of waveforms [Breiman et al. 84]. Data in the later set come from a three class classification problem. Each class consists of a random convex combination of two waveforms chosen among three (see figure 2), plus a random term, i.e. : $x[i] = u\, h_m[i] + (1-u)\, h_n[i] + \varepsilon_i$ $(m,n \in \{1,2,3\}, m \neq n)$. Where x[i] is the ith component of **x** (i=0…20), u is a uniform random number in [0,1], h_m and h_n are two waveforms (figure 2) and ε_i comes from a normal distribution with mean 0 and variance 1.

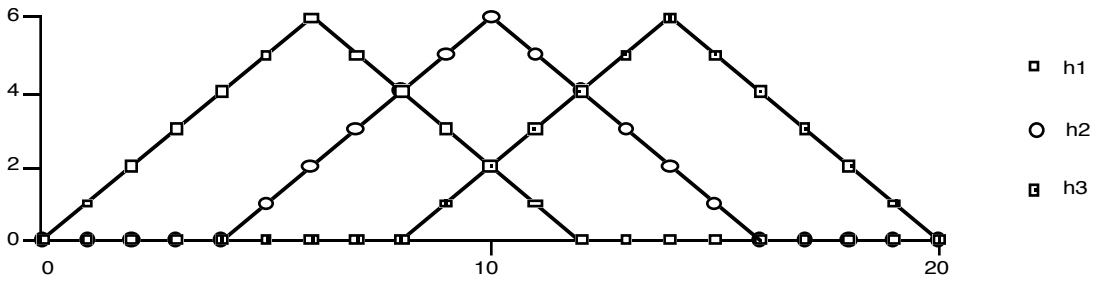


Figure A2.2: the three waveforms used in the test

The net was trained with 100, 85 and 115 examples of respective class 1, 2 and 3 and tested on 5000 additional vectors.

A2.3.2 Performances

A2.3.2.1 Behaviour

Algorithms (13), (14) or (17) with an intermediate layer of 2 cells perform very well on the Iris data set which is well structured. However for the waveforms which are much less structured, a good starting point in the parameter space is needed to converge towards an interesting local minimum. Convergence and stability of constrained gradient techniques are more delicate to achieve than for unconstrained problems. Good convergence results have been obtained by initializing W with the vectors of a principal component analysis (PCA) of the data z.

Quantitative estimation and comparison of unsupervised algorithms is difficult especially when the criterion is complicated as it is the case here because of the presence of the influence function on the output layer. When dealing with classification problems, one way to estimate the quality of the algorithm is to label the output units with the class of the data after unsupervised training. After that the algorithm may be used for classifying new data according to the class of the most active unit. Classification performances should be related to the algorithm ability to discover the data structure. Algorithms (14) and (17) have a similar behaviour on the waveforms dataset. We give below performances of algorithm (17).

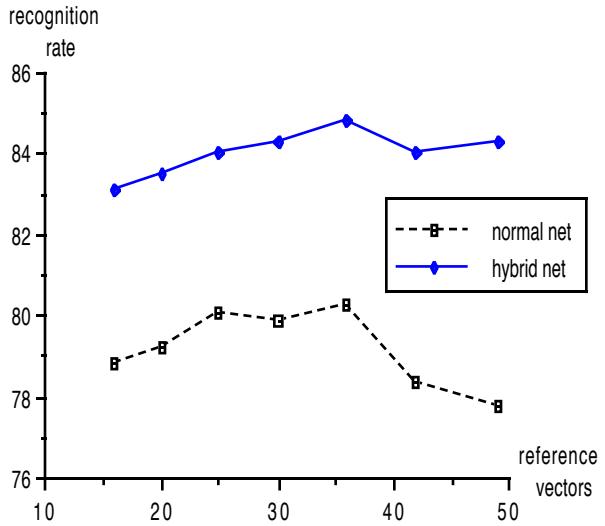


Figure A2.3 classification accuracy for the TM and a 2 hidden cells hybrid network (5000 test patterns)

In figure 3 the classification accuracy obtained by the classical TM (lower line) and the hybrid net trained with a fixed projection matrix set to PCA values (upper line) are compared for different numbers of TM reference vectors. The hybrid net allows to improve performances of about 5%. The dimensionality reduction allows the TM to concentrate upon interesting

information. Note that input dimension has been divided by 10 in this example.

The initial PCA value of W may be improved by training algorithms from 2.4 as may be seen in figure 4 for algorithm (17). In this experiment, the gain is up to 1.3 % for the training set compared to a PCA W matrix. On the test set, the improvement is smaller (0.5 %). However it should be noticed that the fully adaptive approach allows to reach maximum performances with less reference vectors which results in computational savings.

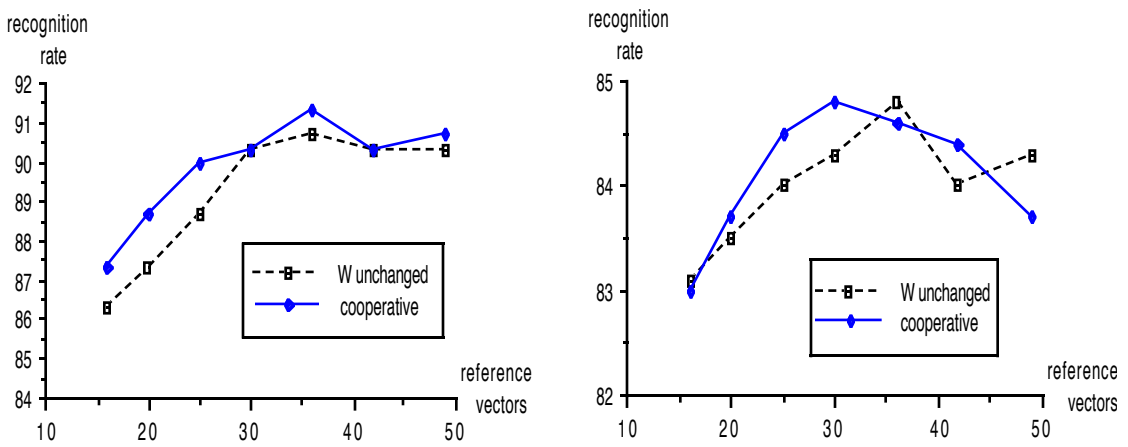


Figure 4: classification accuracy of the 2 hidden cells hybrid network on the learning set (300 vectors, left) and the test set (5000 vectors, right).

W unchanged means W set at PCA values, cooperative means W trained with algorithm (17).

A2.3.2.2 Speed comparison

For a fixed architecture, it is easy to compute for the test phase the theoretical speed increase of the algorithm compared to the original TM. For the learning step this is more difficult because we have to take into account the convergence time. To illustrate the improvement, we have plotted on figure 5 the processing time of the different algorithms on a sparc station 2 for the waveform problem. The hybrid net has a hidden layer with two cells.

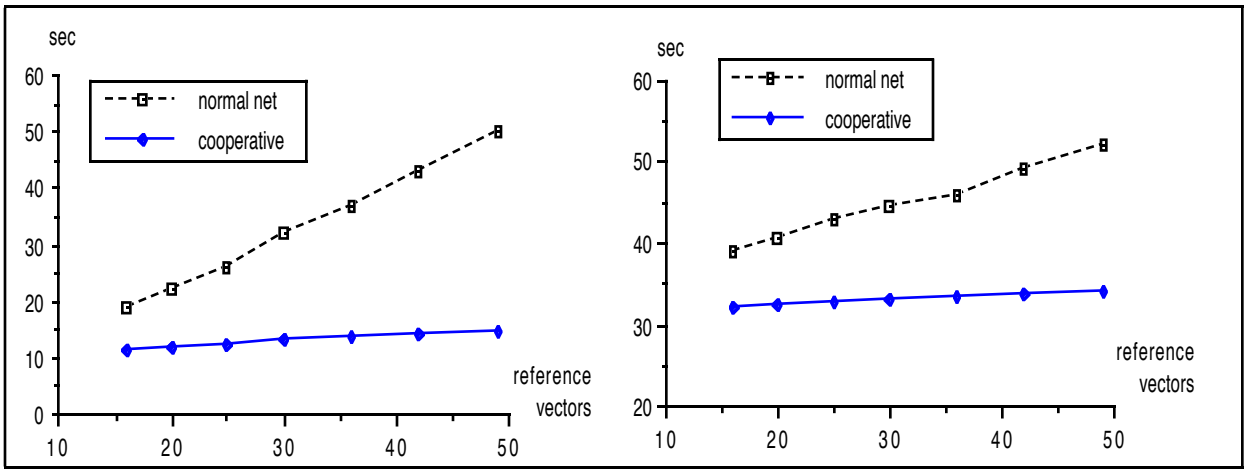


Figure 5: execution time for learning the waveforms (300 vectors, left) and test (5000 vectors, right).

Normal net is the classical TM and cooperative is the hybrid net trained with (17).

These results clearly illustrate the speed gain of the new architecture. This gain should increase together with the input dimension.

A2.3 Discussion

We have presented new algorithms which increase the performances of Topological Maps and make them suitable for high dimensionality problems. These algorithms take into account ad-hoc constraints and allow to solve generic problems of modular architectures. Several issues of these algorithm like speed of convergence and stability are critical to their implementation and are currently being investigated.

A2.4 References

Angeniol B, de La Croix Vaubois G., Le Texier J.Y.: *Self organizing feature maps and the travelling salesman problem*, Neural Networks, Vol. 1, n° 4, 289-293 (1988)

Bollivier M., Gallinari P., Thiria S.: *Cooperation of Neural Nets for Robust Classification* - IJCNN 90 San Diego vol 1, pp 113-120 (1990)

Bottou L., Gallinari P.: *A Framework for the Cooperation of Learning Algorithms* - Proceedings of NIPS 90 (1990)

Bottou L., Gallinari P.: *A formalism for neural net algorithms*, Neuro-Nîmes 91 (1991)

Bourlard H., Wellekens C.J.: *Links between Markov Models and Multilayer perceptrons*, Procedings of IEEE Conference on Neural Information Processing Systems (NIPS'88), pp502-510, Morgan Kaufman. Denver (1988)

Bourlard H.,Wellekens C.J.: *How Connectionist Models could Improve Markov Models for Speech Recognition* - Proc. Int. Symp. on Neural Networks for Sensory and Motor systems, Dusseldorf, Ed. Eckmiller (1990)

Breiman L., Friedmann J., Olshen R. Stone C.: *Classification and Regression Trees*, Wadsworth (1984)

Driancourt X., Bottou L., Gallinari P.: *Multi Layer Perceptron, Learning Vector Quantization and Dynamic Programming: Comparison and Cooperation* - Proceedings of IEEE-IJCNN 91, Seattle (1991)

Driancourt X., Gallinari P.: *A speech recognizer optimaly combining learning vector quantization dynamic programming and multi-layer Perceptron* - to appear in Proceedings of IEEE-ICASSP 92 (San Fransisco) (1991)

Jacobs R. A., Jordan M.I.: *A competitive modular architecture*, NIPS3, Morgan Kaufmann, 767-774 (1991)

Kohonen T.: *Self-Organization and Associative Memory*, Springer Verlag (1988)

Lampinen J., Oja E.:*Fast computations of Kohonen self organization. in Neurocomputing, Nato ASI series vol 68, 65-74.* (1990)

Ritter H., Schulten K.: *Convergence properties of Kohonen's topology conserving maps: fluctuations, stability and dimension selection.* Biological Cybernetics 60, 59-71 (1988)

Scales L.E: *Introduction to non linear optimization*, Macmillan (1985)

Tsypkin Ya: *Adaptation and learning in automatic systems*, Mathematics in science and engineering vol 73, Academic Press (1971)

Références bibliographiques

1. **Ackley D.H., Hinton G.E., Sejnowski T.J.:** *A learning algorithm for Bolzman machines* - Cognitive Science 9: 147-169 (1985)

2. **Acton F.S.:** *Numerical Methods that Work* - Harper and Row, pp 467-468 (1970)

3. **Alessandro (d') C. :** La Représentation du Signal de Parole par une Somme de Fonctions Elémentaires - Thèse de Doctorat, Université Paris VI (Avril 1989)

4. **Alessandro (d') C. :** *La Représentation du Signal de Parole par une Somme de Fonctions Elémentaires* - Thèse de Doctorat, Université Paris VI (1989)

5. **Amari S.I.:** *Neural theory ofasociation and concept-formation* Biological Cybernetics 26: 175-185 (1977)

6. **Baum L. E., Petrie T., Soules G., Weiss N.:** *A maximisation technique occuring in the stastistical analysis of probabilistic functions of Markov Chains* - Annals of mathematic statistics, vol 41, n°1, pp 164-171 (1970)

7. **Bengio Y., Cardin R., De Mori R., Normandin Y.:** *A hybrid coder for hidden Markov models using a recurrent network* - Proceeding of ICASSP 90, pp 537-540 (1990)

8. **Bengio Y., De Mori R., Flammia G. and Kompe R.:** *Gloabal Optimization of a Neural Network-Hidden Markov Model Hybrid* - Proceedings of IJCNN (1991)

9. **Bennani Y.:** *Approches connectionistes pour la reconnaissance automatique du locuteur: modelisation et identification* - Thèse de doctorat, pp 210, Université de Paris XI (1992)

10. **Bennani Y., Gallinari P.:** *On the Use of TDNN-Extracted Features Information In Talker Identification* - ICASSP S6.5, pp 385-388, Toronto (1991)

11. **Berger J.O.:** *Stastistical décision theory and Bayesian alalysis* - Spichoper (1985)

12. **Bollivier M. de:** *Incorporation de la modularité dans les réseaux connexionnistes par décomposition de tâche* - Thèse de doctorat, Université Paris XI, Orsay , pp 73-74 (1992)

13. **Bottou L. :** *Reconnaissance de la parole par réseaux multi-couches* - Rapport de stage, DEA d'Informatique du LRI, Université de Paris XI, Orsay (1988)

14. **Bottou L., Cortes C., Vapnik V. N.:** *On the mesure of the Effective VC Dimension* - Submited to Neural Computation

15. **Bottou L., Gallinari P.:** *A Framework for the cooperation of Learning Algorithms* - Advances in Neural Information Processing Systems 3, pp 781-788, Morgan-Kaufman. Denver (1990)

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

17. **Bottou L., Vapnik V. N.:** *Local Learning Algorithms* - Neural Computation Vol 3, Nb 6, pp 888-900 (1992)

18. **Bottou L.:** *A Framework for the Cooperation of Learning Algorithms* - Proceeding of NIPS (1990)

19. **Bottou L.:** *Une approche théorique de l'Apprentissage Connexionniste: Applications à la reconnaissance de la Parole* -Thèse de doctorat, Université Paris XI, Orsay , pp 217 (1991)

20. **Bottou. L.:** *Reconnaissance de la parole par réseaux multi-couches* - Neuro-Nîmes pp 197-218 EC2 (1988)

21. **Bourlard H., Morgan N., Wooters Ch. :** *Connectionist aproaches to the Use of Markov Models for Speech Recognition* - Advances in Neural Information Processing System 3, Morgan Kaufman (1991)

22. **Bourlard H., Morgan N., Woters Ch., Renals S. :** *CDNN: A context-dependent neural network for continuous speech recognition* - ICASSP, vol II-349, San Francisco (1992)

23. **Bourlard H., Morgan N.:** *A continuous Speech Recognition System Embedding MLP into HMM* - Proceedings of NIPS 2 pp. 502-510, Morgan Kaufman (1991)

24. **Bourlard H., Wellekens C. :** *Links between Markov Models and Multilayer perceptrons* - Advance in Neural Information Processing Systems, Vol 1, pp. 502-510, Morgan Kaufman, Denver (1989)

25. **Braverman E.M., Litvakov B.M.:** Convergence of algorithms of learning and adaptation - congress IFAC 4th, Warsawa, (1969)

26. **Breiman L., Friedman J. H., Olshen R. A., Stone C. J.:** *Classification and Regression Trees* - Wadsworth International Group (1984)

27. **Brent R.P.:** *Algorithms for minimization without Derivatives* - Prentice-Hall, Englewood Cliffs, N.J. (1973)

28. **Chistovich I.A., Granstrem M.P., Kozhevnikov V.A., Lesogor L.W., Shupljakov V.S., Taljasin P.A., Tjulkov W.A. :** *A functional model of signal processing in the périférical aauditory system* - Acustica, Vol. 31, (1974)

29. **Csizar I., Tusnady G.:** *Information geometry and alternating minimization procedure* - Preprint n° 35, Math-Inst. Hungar, Acad. Sci., Budapest (1983)

30. **Devijer P. A., Kitter J.:** *Pattern Recognition: A statistical Approach* - Prentice Hall International (1982)

31. **Devilers L. :** *Reconnaissance monolocuteur des phonèmes français au moyen de réseaux à masques temporels* - Actes des XVIIIèmes Journées d'Etudes sur la Parole - Montréal, (1990)

32. **Devillers L., Dugast C.:** *Comparison of Continuous Mixture Densities and TDNN in a Viterbi-framework: Experiment on Speaker Dependant Darpa RM1* - Eurospeech, Genova, Italy, 24-26 (1991)

33. **Devyaterikev, Kaplinsky, Tsypkin:** *On the convergence of the algorithms of learning* - Automat. i Telemeh. n° 10 (1969)

34. **Driancourt X.:** *Extraction de Primitives Caractéristiques du Signal par Réseaux Neuronaux pour la Reconnaisance de la Parole* - Rapport DRET CJC-89/1591 (1991)

35. **Driancourt X., Bottou L., Gallinari P.:** *Comparison and Cooperation of Several Classifiers* - Proceedings of ICANN (1991)

36. **Driancourt X., Bottou L., Gallinari P.:** *Multi-Layer Percpetron, Learning Vector Quantization and Dynamic Programming: Comparison and Cooperation* - Proceedings of IJCNN (1991)

37. **Driancourt X., Bottou L.:** *TDNN Extracted Features* - Actes de Neuronimes, EC2 (1990)

38. **Driancourt X., Gallinari P.:** *A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer Perceptron* - Proceedings of ICASSP (1992)

39. **Driancourt X., Gallinari P.:** *An empirical risk optimizer for speech recognition* - Proceedings of IJCNN (1992)

40. **Driancourt X., Gallinari P.:** *Empirical risk optimisation: neural networks and dynamic programming* - Proceedings of NNSP (1992)

41. **Driancourt X.:** *Extraction de primitives caractéristiques du signal par réseaux neuronaux pour la reconnaisance de la parole* - Rapport de recherche DRET, CJC 89/1591 (1991)

42. **Drucker H., Le Cun Y.:** *Improving generalization performance using double backpropagation* - IEEE Transaction on Neural Networks (1992)

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

44. **Ephraim Y., Rabiner L.R.:** *On the Relation Between Modeling Approaches for Information Sources* - S1.2 pp 24-27, proceedings of IEEE-ICASSP (1988)

45. **Ermolyev Y. M., Nekrilova Z. V. :** *Method of stochastic gradient and its application* - First seminary on the theory of optimal decisions, Kiev (1967)

46. **Fabian V. :** *Pregled deterministichych a stochastilhych approximacnich method pro minimalizaci funkci* - Kybernetica (Kiev) 1,6 (1965)

47. **Fisk D.L.:** *Quasi-martingales* - Tans. Amer. Math. Soc, nb 120, pp 359-388 (1965)

48. **Fraisse P. :** La Psychologie Expérimentale - Presse Universitaires de France, Paris (1966)

49. **Franzini M., Lee K. F., Waibel A.:** *Connectionnist Viterbi Training: A new hybrid method for continuous speech recognition* - ICASSP pp 245 (1990)

50. **Funahashi K. I.:** *On the approxmate realization of continuous mapping by neural networks* - Neural Networks Vol 2. pp 183-192, (1989)

51. **Gauvain J. L.:** *Reconnaissance de mots enchaînés et détection de mots dans la parole continue* - Thèse de 3ème cycle, LIMSI, Orsay (1982)

52. **Gribenski A.:** L'audition - Presse Universitaires de France, Paris (1951)

53. **Grossberg S.:** *Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors* - Biological cybernetics 23: 121-134 (1976)

54. **Guyon I., Vapnick V. N, Boser B. E., Bottou L., Solla S. A.:** *Structural risk minimization for character recognition* - Advances in Neural Processing Systems 4, Morgan Kaufman (1992)

55. **Haffner P., Fanzini M., Waibel A.: Integrating** *Time alignment and neural networks for high performance continuous speech recognition* - Proceeding of ICASSP, S2.17, pp 105-108 (1991)

56. **Hebb D.O.:** *The Organization of Behavior* - New York Wiley (1947)

57. **Hopfield J.J.:** *Neural netwoks and physical systems with emergent collective computational abilities* - Proceedings of the National Academy of Sciences 70: 2554-2558 (1982)

58. **Hornik K., Stinchcombe H. W.:** *Multilayer feedforward networks are universal aproximators* - Neural Networks Vol 2. pp 359-366, (1989)

59. **Huynh C., Bourrely J., Bottou L.:** *Sensor Model Inversion and Regularization with Neural Network* - Submited to NIPS 93

60. **Iso K., Watanabe T.:** *Large vocabulary speech recognition using neural prediction model* - pp 61-64 ICASSP (1991)

61. **James W.:** *Psychology (Briefer Course)* - New York, Holt (1890)

62. **Katagari S., Lee C.H., Huang B.H.:** *New discriminative training algorithms based on the generalized probabilistic decscent method* - NNSP, pp 299-308 (1991)

63. **Keitel W.D, Neff W.D. :** *Auditory System* - in Handbook of sensory physiology, Volume V, Springer-Verlag, Berlin (1975)

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

65. **Kohonen T.:** *Self-organized formation of topologically correct feature maps* - Biological Cybernetics 43, pp 59-69 (1982)

66. **Kolmogorov A. N.:** *On the representation of continuous functions functions of may variables by superposition of continuous functions of one varible and addition* - Doklady Akademii Nauk SSR 144, pp. 679-681 (1957)

67. **Lashley K.S.:** *In search of the engram* - in *Society of Experimental Biology Symposium*, No. 4: *Psychological Mechanisms in Animal Behavior*, Cambridge University Press, pp. 454-455, 477-480 (1950)

68. **Le Cun Y., Denker J., Solla S.:** *Optimal Brain Damage* - Advances in Neurla Information Processing Systems 2 Morgan Kaufman (1990)

69. **Le Cun Y.:** *Generalization and Network Design Strategies* - Connectionnism in Perspective, Zurich, Switzerland Elsiever (1989)

70. **Le Cun Y.:** *Modèles conexionnistes de l'Apprentissage* - Thèse de doctorat de l'université de Paris 6 (1987)

71. **Liénard, J.S. :** Les Processus de la Communication Parlée - Masson (1977).

72. **Macchi O.:** *Advances in adaptive filtering* - pp 41-57 in E. Biglieri and Prati G. (eds) Digital Communications, North Holland, Amsterdam (1986)

73. **Mallat S.G. :** *A theory for Multiresolution Signal Decomposition: the Wavelet Representation* - Technical report MS-CIS-87-22, University of Pensylvania,Philadelphie, (1987)

74. **Marcquardt D. W.:** in Journal of Society for Industry Applied Mathematics, vol 11, pp 431-441 (1963)

75. **Marcos S., Vignat C., Macchi O.:** *L'algorithme de rétropropagation du gradient en réseaux de neurones comparés aux alorithmes adaptatifs du traitement du signal* - Rapport Interne., Laboratoire des signaux et Systèmes, Gif sur Yvette (September 1989)

76. **Marcos S., Macchi O., Vignat C., Dreyfus G., Personnaz L., Roussel-Ragot P.:** *A unified framework for gradient algorithms used for filter adaptation and neural network training* - International journal of circuit theory and applications,Vol 20, 159-200 (1992)

77. **Mc Dermott E., Katagari S.:** *Shift-invariant multi-category phonem recognition unsing Kohonen's LVQ2* - Proceedings of ICASSP, pp 81-84, Galsgow U.K. (1989)

78. **Mellouk A., Gallinari P.:** *A Discriminative Neural Prediction System for Speech Recognition System* - ICASSP (1993)

79. **Mellouk A., Gallinari P.:** *Discriminative Training for Improved Neural Prediction Systems* - ICASSP (1994)

80. **Metivier M.:** *Martingales et convergence presque sûre d'algorithmes stochatiques, in "Outils et modèles mathématiques pour l'automatique, lanalyse de système et le traitement du signal"* - Edition du CNRS, vol 1, partie 3, "Techniques probabilistes en automatique et traitement du signal", pp 529-552 (1981)

81. **Minsky M., Papert S.:** *perceptrons*, Cambridge, MA: MIT Press (1969)

82. **Nelder J.A., Mead R.:** *Computer Journal* - Vol 7, pp 308, (1965)

83. **Neveu J.:** *Martingales à temps discret* - Masson (1972)

84. **Nillson N.J.:** *Learning Machines: Foundation of trainable Patter-Clasiciation Systems* - MacGtraw Hill (1965)

85. **Pitts W., McCulloch W.S.:** *A logical calculus of the ideas immanent in nervous activity* - Bulletin of Mathematical Biophysics 5:115-133

86. **Poggio T., Girosi F.:** *A theory of Networks for approximation and learning* - IEEE vol. 78, no 9, pp 1481-1496 (1990)

87. **Polack E.:** *Computational Methods in Optimization* - Academic Press, New York (1971)

88. **Polack E.:** *Computational Methods in Optimization* - Academic Press, New York (1971)

89. **Press W.H., Flannery B.P. Teukolsky S.A., Vetterling W.T.:** *Numerical recipes: the art of scientific computing* - Cambridge University Press (1986)

90. **Quinlan J. R.:** *Induction of Decision Trees* - Machine Learning, vol 1, n°1, pp 81-106 (1986)

91. **Rosenblatt F.:** *The Perceptron: a probabilistic model for information storage and organisation of the brain* - Psychological review 65: 386-408 (1958)

92. **Rumelhart D.E., Hinton G.E., Williams R.J.:** *Learning representations by back-propagating errors* - Nature 323:533-536 (1986)

93. **Sakoe H., Isotani R., Yoshida K. Iso K. Watanabe T.:** *Speaker Independant Word Recognition Using Dynamic Programming Neural Networks* - Proceeding of ICASSP, pp 107-110 (1988)

94. **Schwenk H., Gallinari P., Driancourt X. :** *a modular system which improves the topological maps* - IEEE IJCNN 92, Baltimore (1992)

95. **Seiker A., Hanson B.A. :** *Improved 1-Bark bankwidth auditory filter* - in JASA, Vol. 75 No 6 (Jully 1984)

96. **Simard P., Le Cun Y., Denker J.:** *Efficient pattern recognition using a new transformation distance* - Advances in Neural Information Processing Systems 5 (1993)

97. **Simard P.:** *Tangent prop: a formalism for specifying selected invariances in adaptive networks* - Advances in Neural Information Processing Systems 4 (1992)

98. **Soules G.W.:** *A gradient derivation of PTAH* - IDA-CRD Tech Report N°80328, (1977)

99. **Sprecher D. A.:** *On the structure of continuous functions of several variables* - Transactions of the American Mathematical Society, 115, pp 340-355 (1965)

100. **Stevenin A., Gallinari P.:** *Concept Prediction for Spontaneous Speech Retranscription* - WCNN (1994)

101. **Suddhart S., Holden A.:** *symbolic neural systems and the use of hints for developping complex systems* - Int. J. Machine Studies 35 pp 291 (1991)

102. **Thodbey H.:** *Ace of Bayes: application of neural networks with pruning* - TR n° 1132E Danish meet. research institute (1993)

103. **Tsypkin Y.:** *Adaptation and learning in Automatic systems* - Mathematics in science and ingineering, vol 73, Academic Press (1971)

104. **Tsypkin Y.:** *Foundation of the Theory of Learning Systems* - Mathematics in science and ingineering, vol 101, Academic Press (1973)

105. **Vapnik V.N., Chernovskys A.Y.:** *On the uniform convergence of relative frequencies of events to their probabilities* - Theory of Probability and its Applications, vol 16, No 2, pp. 264-280 (1971)

106. **von Neumann J.:** *The Computer and the Brain* - New Haven: Yale University Press (1958)

107. **Waibel A., Hanazawa T., Hinton G., Lang K. :** *Phoneme recognition: neural networks vs. Hidden Markov Models* - Proceedings of ICASSP 88, S-Vol 1, pp 107-110, (1988)

108. **Watrous R. L., Lokendra S.:** *Learning phonetic features using connectionnist networks : an experiment in speech recognition* - pp 381-388, IEEE vol 4, san Diego California (1987)

109. **Widrow B., Hoff M.E.:** *Adaptive switching circuits* - 1960 IRE WESCON Convention Record, New York: IRE, pp. 96-104 (1960)

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